Contributions à l'analyse de données fonctionnelles multivariées, application à l'étude de la locomotion du cheval de sport

Amandine Schmutz

N^o d'ordre NNT : 2019LYSE1241

THESE de DOCTORAT DE L'UNIVERSITE DE LYON

opérée au sein de

l'Université Claude Bernard Lyon 1

Ecole Doctorale N^o ED512
(Infomaths)

Spécialité de doctorat : **Mathématiques Appliquées**

Soutenue publiquement le 15/11/2019 par :

Amandine SCHMUTZ

Contributions à l'analyse de données fonctionnelles multivariées, application à l'étude de la locomotion du cheval de sport.

Devant le jury composé de :

Pr. Julien JACQUES	Directeur de thèse
Pr. Laurence CHÈZE	Directrice de thèse
Dr. Pauline MARTIN	Encadrante en entreprise
Pr. Sophie LAMBERT LACROIX	Rapporteur
Pr. Cristian PREDA	Rapporteur
Dr. Vincent BRAULT	Examinateur
Pr. Henri CHATEAU	Examinateur
Pr. Véronique MAUME DESCHAMPS	Examinateur

Résumé

Avec l'essor des objets connectés pour fournir un suivi systématique, objectif et fiable aux sportifs et à leur entraineur, de plus en plus de paramètres sont collectés pour un même individu. Une alternative aux méthodes d'évaluation en laboratoire est l'utilisation de capteurs inertiels qui permettent de suivre la performance sans l'entraver, sans limite d'espace et sans procédure d'initialisation fastidieuse. Les données collectées par ces capteurs peuvent être vues comme des données fonctionnelles multivariées : se sont des entités quantitatives évoluant au cours du temps de façon simultanée pour un même individu statistique.

Cette thèse a pour objectif de chercher des paramètres d'analyse de la locomotion du cheval athlète à l'aide d'un capteur positionné dans la selle. Cet objet connecté (centrale inertielle, IMU) pour le secteur équestre permet de collecter l'accélération et la vitesse angulaire au cours du temps, dans les trois directions de l'espace et selon une fréquence d'échantillonnage de 100 Hz. Une base de données a ainsi été constituée rassemblant 3221 foulées de galop, collectées en ligne droite et en courbe et issues de 58 chevaux de sauts d'obstacles de niveaux et d'âges variés.

Nous avons restreint notre travail à la prédiction de trois paramètres : la vitesse par foulée, la longueur de foulée et la qualité de saut. Pour répondre aux deux premiers objectifs nous avons développé une méthode de clustering fonctionnelle multivariée permettant de diviser notre base de données en sous-groupes plus homogènes du point de vue des signaux collectés. Cette méthode permet de caractériser chaque groupe par son profil moyen, facilitant leur compréhension et leur interprétation. Mais, contre toute attente, ce modèle de clustering n'a pas permis d'améliorer les résultats de prédiction de vitesse, les Support Vector Machine (SVM) restant le modèle ayant le pourcentage d'erreur inférieur à 0.6 m/s le plus faible. Il en est de même pour la longueur de foulée où une précision de 20 cm est atteinte grâce aux SVM. Ces résultats peuvent s'expliquer par le fait que notre base de données est composée uniquement de 58 chevaux, ce qui est un nombre d'individus très faible pour du clustering.

Nous avons ensuite étendu cette méthode au co-clustering de courbes fonctionnelles multivariées afin de faciliter la fouille des données collectées pour un même cheval au cours du temps. Cette méthode pourrait permettre de détecter et prévenir d'éventuels troubles locomoteurs, principale source d'arrêt du cheval de saut d'obstacle.

Pour finir, nous avons investigué les liens entre qualité du saut et signaux collectés par l'IMU. Nos premiers résultats montrent que les signaux collectés par la selle seuls ne suffisent pas à différencier finement la qualité du saut d'obstacle. Un apport d'information supplémentaire sera nécessaire, à l'aide d'autres capteurs complémentaires par exemple ou encore en étoffant la base de données de façon à avoir un panel de chevaux et de profils de sauts plus variés.

Remerciements

Ce travail de thèse a été financé par le LabCom "CWD-VetLab" qui est soutenu financièrement par l'Agence Nationale de la Recherche (contrat ANR 16-LCV2-0002-01).

Je tiens tout d'abord à remercier Pauline Martin pour m'avoir donné l'opportunité d'allier mes deux passions, l'équitation et l'analyse de données, dans mon travail quotidien et cela dans un cadre de vie agréable (quoi qu'un peu isolé) qui m'a permis de multiplier par trois la taille de ma ménagerie. Je remercie aussi mes directeurs de thèse Stéphane Bonnevay, grâce à qui j'ai pu m'inscrire en thèse en mathématiques, Julien Jacques pour sa disponibilité, son soutien et sa patience dans mon apprentissage des mathématiques "théoriques" et Laurence Chèze pour sa disponibilité et son regard extérieur par rapport à nos travaux mathématiques. Vous m'avez permis de gagner en rigueur dans la formalisation des méthodes statistiques et de me découvrir des capacités qui m'étaient jusqu'alors inconnues en lecture d'équations mathématiques.

Je remercie les personnes de mon comité de suivi de thèse, Sophie Lambert Lacroix et Julien Velcin qui ont veillé à la bonne progression de mon travail.

Je suis aussi reconnaissante envers l'équipe du laboratoire ERIC, qui m'a toujours fait sentir la bienvenue lors de mes cours passages sur Lyon et pour la bonne ambiance au sein de notre bureau des doctorants. Je remercie particulièrement Margot, Antoine, Erwan, Hussein, Jairo, Habiba, Julien V., Mohammed et Adrien.

Je remercie l'équipe de Lim France pour ces 3 ans et demi passés avec eux et les nombreux barbecues. Camille pour m'avoir appris les bases de lecture des signaux d'IMUs et pour sa participation à quasi toutes mes campagnes de mesure. Marie, pour son assistance sans faille dans les sessions de mesures iJump. Jeanne, pour ton cours passage au sein de l'équipe plein de bonne humeur et ton aide dans la rédaction de ma publication biomécanique. Emilie pour nos pauses thé et ton écoute dans les moments difficiles. Et bien sûr Simon, Mathieu, Zachary, Denis, Guillaume, Caroline, Raphael, Franck, Christophe, Jean-charles, David, Richard pour votre bonne humeur et votre sens de l'humour.

Enfin je remercie ma famille, mon conjoint et mes proches pour leurs encouragements et leur confiance. Particulièrement Manue, Emilie et Mathilde pour avoir testé mon premier package me permettant de résoudre les multiples bugs et pour votre aide sur des points précis de programmation R. Marine D., Marine O. et Aurélie pour nos partages de moments de solitudes de notre périple de futurs docteurs. Et tous les autres pour avoir été là pour moi quand j'en ai eu besoin.

Table des matières

Chapitre 1

Introduction

D'après l'article 234 du règlement FEI pour le saut d'obstacle, la vitesse des chevaux pour les compétitions internationales doit être comprise entre 350 m/min et 400 m/min, avec des exceptions selon le type d'épreuve (FEI, FEI Jumping Rules, 26ème édition, 2019). La vitesse est donc un paramètre clef de succès dans les compétitions de saut d'obstacle et un facteur important à évaluer lors des entrainements.

Le système de capture de mouvement optique (3D-mocap) est actuellement l'outil de référence pour l'analyse de la locomotion du cheval et peut être utilisé pour mesurer la vitesse et la longueur de foulée [Pfau et al., 2005]. Néanmoins, l'utilisation de cet outil contraint la taille de la zone de mesure à un petit volume (fonction du nombre de caméras disponibles) et la mise en place de ce système est très chronophage. De plus, le traitement des données peut être compliqué lorsque l'on sort des modèles standards humains définis par le logiciel de traitement, le suivi des marqueurs 3D devient semi-automatisé et nécessite donc un suivi manuel des marqueurs par l'opérateur [der Kruk and Reijine, 2018]. Cette étape de traitement des données peut aussi être longue si le nombre de caméras est insuffisant par rapport au volume d'étude ou au nombre de marqueurs suivis, surtout si ceux-ci sont très proches les uns des autres. Ces deux contraintes rendent son utilisation quotidienne ou lors de championnats impossible.

Avec l'essor des nouvelles technologies, de nouvelles techniques d'analyse de la locomotion ont émergé et ont rendu possible le développement d'outils portatifs fournissant des paramètres objectifs de la locomotion du cheval [Bosch et al., 2018] ou permettant de détecter la boiterie [Pfau et al., 2016]. Ces outils peuvent permettre un suivi systématique, objectif et fiable de la performance. Ils s'appuient sur des centrales inertielles (IMU) à bas coût composées de 3 capteurs : un accéléromètre tri-axes, un gyroscope tri-axes et un magnétomètre tri-axes. Coupler les IMUs avec un GPS permet d'améliorer la précision des mesures, limiter l'erreur liée au biais inertiel et améliorer l'estimation de paramètres de locomotion comme la vitesse [Tan et al., 2008, Zihajehzadeh et al., 2016]. Mais le GPS le plus précis abordable pour le grand public a une précision satisfaisante pour l'estimation de la vitesse moyenne mais une faible précision de 8 m [Zandbergen, 2009] pour l'estimation de distance, et ces estimations peuvent être impactées négativement par la présence d'obstacles [Wing et al., 2005]. Le GPS présente aussi la contrainte majeure de ne pas pouvoir être utilisé à l'intérieur à cause de la perte de signal lors de la présence d'un toit [Johansson, 2009], alors que les compétitions de saut d'obstacles peuvent aussi bien se dérouler en intérieur qu'en extérieur.

Il existe trois grandes familles de méthodes développées pour calculer des paramètres caractéristiques du mouvement à partir de signaux collectés par les IMUs. La première catégorie regroupe les méthodes basées sur un modèle physique, comme le modèle du pendule pour l'estimation de la vitesse d'un homme au cours de la marche par exemple. Ces modèles simplifient des comportements biomécaniques complexes et incorporent des informations spécifiques au sujet comme la longueur de sa jambe [Brandes et al., 2006, Murphy et al., 2010]. Ce type de modèle physique simplifé n'existe pas pour l'allure dissymétrique du cheval au galop et cette catégorie de méthodes ne pourra pas être appliquée ici. La deuxième catégorie correspond aux méthodes basées sur l'intégration des signaux collectés, elles utilisent des prétraitements des données comme des filtres de Butterworth pour limiter les dérives liées à l'intégration mathématique [Brzostowski, 2018]. Ces méthodes nécessitent de formuler des hypothèses réalistes pour corriger la dérive des capteurs et définir les constantes d'intégration. Comme par exemple dans le cas de la méthode proposée par [Pfau et al., 2005] qui estime le déplacement du cheval sur tapis roulant à partir d'une IMU positionnée sur le garrot. Dans ces conditions, pendant une

foulée, le capteur fixé sur le garrot du cheval est supposé décrire une trajectoire cyclique en revenant à son point de départ, ce qui permet de définir les constantes d'intégration nécessaires au calcul de sa vitesse et de son déplacement. Cette hypothèse n'est pas applicable au cheval qui se déplace dans un espace non contraint. La troisième catégorie correspond aux méthodes dites de machine learning, qui sont utilisées chez l'humain pour l'estimation de la vitesse de la marche [Mannini and Sabatini, 2014,Sabatini and Mannini, 2016] ou l'évaluation de la récupération de la locomotion suite à une blessure [Kampakis, 2013] à partir de données IMUs, mais n'ont pas encore été appliquées sur des modèles animaux.

Cette thèse a pour premier objectif de mettre en place un modèle de prédiction permettant l'estimation de la vitesse du cheval à chaque foulée. Cette prédiction doit être réalisée à partir des signaux mesurés par une seule centrale inertielle ne comportant pas de magnétomètre, avec une précision de 0.6 m/s (36 m/min) pour répondre aux attentes des professionnels du saut d'obstacles. Une fois intégré dans l'objet connecté, le modèle doit permettre de dépasser les limites des systèmes GPS et 3D-mocap. Le second objectif de la thèse est de fournir un modèle permettant l'estimation de la longueur de foulée avec une précision de 20 cm. Pour finir, le dernier objectif de la thèse est de caractériser la qualité du saut d'obstacles à l'aide des signaux collectés.

1 Récolte des données

Pour la réalisation de chacun des objectifs, il est nécessaire de constituer une base de données d'entrainement afin de construire des modèles mathématiques qui permettront par la suite de calculer automatiquement un résultat à partir de nouvelles données mesurées.

Figure 1.1: Orientation des axes de nos capteurs

Pour répondre aux deux premiers objectifs, 3221 foulées de galop ont été récoltées sur 58 chevaux de saut d'obstacle montés de race, taille (129 - 176cm), âge (5-18 ans) et niveau de compétition (amateur ou professionnel) différents. Ces données sont récoltées à l'aide d'une IMU (LSM6DSL, STMicroelectronics) composée d'un accéléromètre (8g) et d'un gyroscope (2000dps) positionnés dans la selle proche du garrot des chevaux. La

fréquence d'échantillonnage de ces capteurs est de 100 Hz, ils collectent les données selon les trois axes du mouvement : dorso-ventral (X), medio-latatéral (Y) et cranio-caudal (Z) (cf. Figure 1.1). Les données collectées par les IMU sont envoyées vers le smartphone (iPhone X, Apple Inc.) de l'utilisateur grâce à une antenne Bluetooth et sont stockées sur un serveur.

Deux protocoles de mesures différents ont été utilisés : le premier pour mesurer la vitesse en ligne droite et le second pour mesurer la vitesse en courbe.

La première base de données a été constituée en couplant les mesures de l'IMU avec celles d'une méthode de référence pour la mesure de la vitesse : l'acquisition de vidéos rapides 2D. Pour cela, les données de l'IMU sont synchronisées à un système de 4 caméras de tracking 2D (UI-5240CP-M-GL, IDS) sur un champ d'étude rectiligne de 26 mètres (cf. Figure 1.2). Cette méthode permet d'évaluer la vitesse à chaque foulée dans le champ d'étude rectiligne à l'aide des 4 caméras, 2 ordinateurs et de marqueurs réfléchissants positionnés sur les membres du cheval, le tapis, la selle et le champ d'étude (cf. Figure 1.3). La précision de ce système est de 1.4% de la distance mesurée [Martin, 2015], ce qui correspond à +/- 2.8cm pour une distance mesurée de 2 mètres par exemple.

Des données sont collectées pour différentes vitesses choisies par le cavalier (normale, lente et rapide), avec et sans sauts avant et après le champ des caméras et avec et sans barres au sol espacées de 2.5m à 4.5m dans le champ de mesure. Ces conditions variées sont choisies pour avoir une grande variété de données dans le jeu de données d'entrainement et afin d'être au plus près des conditions rencontrées lors de l'entrainement quotidien des chevaux de saut.

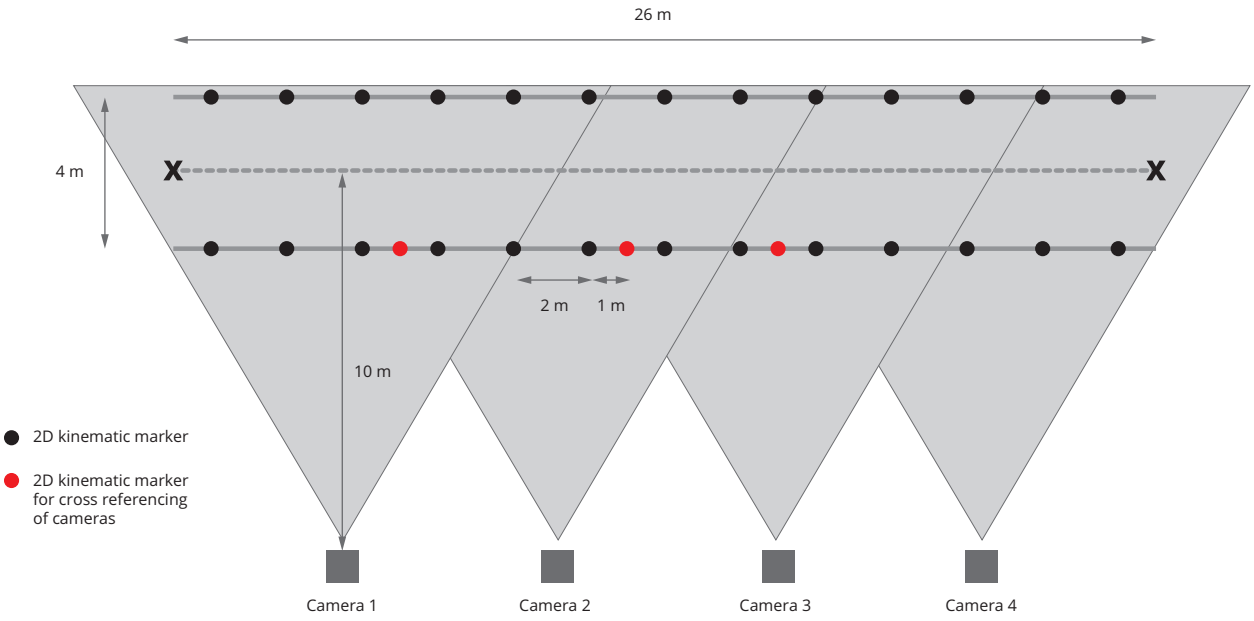


Figure 1.2: Montage du champ de mesure en ligne droite

Le second protocole de mesure a été mis en place pour collecter la vitesse en virage. En effet, le système de tracking avec caméras 2D a une précision satisfaisante uniquement quand le cheval se déplace de façon perpendiculaire au champ des caméras, ce qui n'est pas possible dans le cas de lignes courbes avec des caméras fixes. Il a donc fallu réfléchir à un protocole de mesure spécifique aux courbes. Pour cela, des couloirs circulaires de périmètre connu ont été définis à l'aide de cônes et d'une largeur suffisamment faible pour

Figure 1.3: Cheval équipé avec les marqueurs réfléchissants (à gauche) et champ d'étude vidéo (à droite). Les marqueurs positionnés à la hanche, jarret et croupe ont servi à la mesure de la taille des membres de chaque cheval.

limiter au maximum le déplacement du cheval de part et d'autre de la trajectoire courbe matérialisée (cf. Figure 1.4). La distance parcourue est alors calculée avec $distance = 2\pi r$ et r le rayon du demi-cercle.

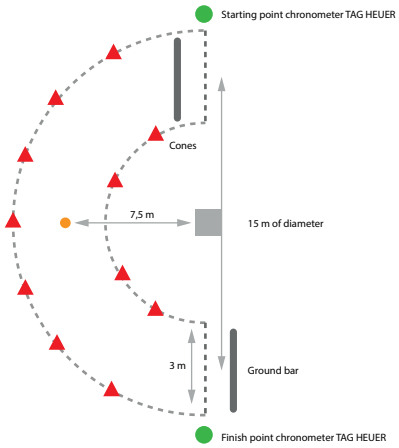


Figure 1.4: Montage du champ de mesure en virage

Le temps passé dans la courbe est mesuré à l'aide d'un chronomètre automatique (CP 520, Tag Heuer) qui se déclenche à l'entrée et la sortie du cheval. La vitesse moyenne du cheval est alors calculée avec $vitesse = \frac{distance}{temps}$. A chaque foulée du cheval galopant sur cette courbe, est alors associée cette vitesse moyenne. Par exemple, si la vitesse moyenne dans la courbe était de 6 m/s et que le cheval a fait 5 foulées, alors à chacune de ces 5 foulées a été associée la vitesse de 6 m/s.

Dans le cas de notre travail, la seule allure étudiée chez le cheval est le galop, nous définissons une foulée par la distance parcourue entre deux posés successifs de l'antérieur directeur (cf. Figure 1.5). En ligne droite, la longueur de foulée a été mesurée à l'aide du système 2D, la foulée est alors définie à partir des images où le sabot de l'antérieur directeur touchait le sol. En ligne courbe, la mesure a été faite à la main à l'aide d'un

odomètre laser-roll pilot (cf. Figure 1.6), à partir des empreintes au sol laissées par les sabots.

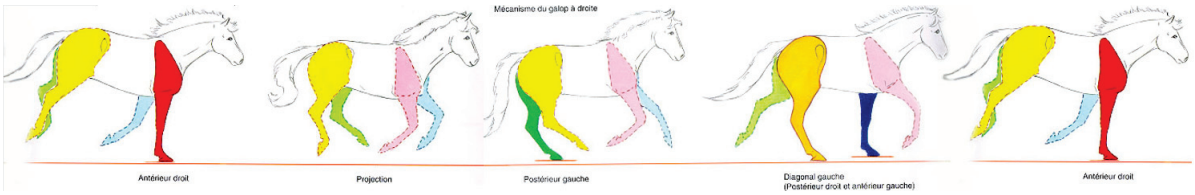


Figure 1.5: Décomposition du galop à main droite, la distance parcourue entre les deux posés d'antérieur droit correspond à la longueur de foulée

Figure 1.6: Odomètre permettant la mesure de la longueur de foulée entre deux posés d'antérieur directeur successifs

Ces mesures serviront de référence sur laquelle s'appuyer pour construire des modèles de prédiction en les mettant en rapport avec les signaux collectés par l'IMU sur ces mêmes foulées. Sept campagnes de mesures ont ainsi été réalisées au cours de ma thèse, permettant de collecter des données pour 3221 foulées de galop dont 2906 foulées en ligne droite et 315 foulées en virage conduisant à une base finale de 3221 lignes et 6×101 colonnes (101 valeurs correspondant à une seconde de mesure selon les accélérations et vitesses angulaires tri-axes). En effet, comme cela sera expliqué plus loin, une "foulée" en tant qu'individu statistique sera systématiquement définie par une seconde d'enregistrement à partir du début de la foulée.

Suite à ces journées de prises de mesure, il faut ensuite prévoir un temps d'analyse d'images dans le cas des lignes droites, appelé aussi tracking, pour récupérer les données de vitesse. Pour cela, une interface clic-bouton a été développée sur Matlab par l'école vétérinaire de Maison Alfort (cf. Figure 1.7). Pour chaque passage d'un cheval les marqueurs réfléchissants doivent être numérotés sur chaque image afin de suivre leur position d'une image à l'autre (numérotés de 1 à 12). Un programme permet ensuite de suivre tous les posés de l'antérieur directeur et le déplacement du pommeau de la selle en nombre de pixels. Grâce à la structure de calibration présente sur le tapis de selle et visible à chaque image, d'une taille connue de 15cm (marqueurs 9 et 10 Figure 1.7), le nombre de pixels est converti en une distance parcourue en centimètres pour chaque foulée. Le temps écoulé entre deux posés d'antérieur peut être obtenu en comptant le nombre d'images et en le divisant par la fréquence d'échantillonnage de la caméra (75 Hz). On peut alors en déduire la vitesse en divisant cette distance par le temps écoulé entre deux posés d'antérieur directeur.

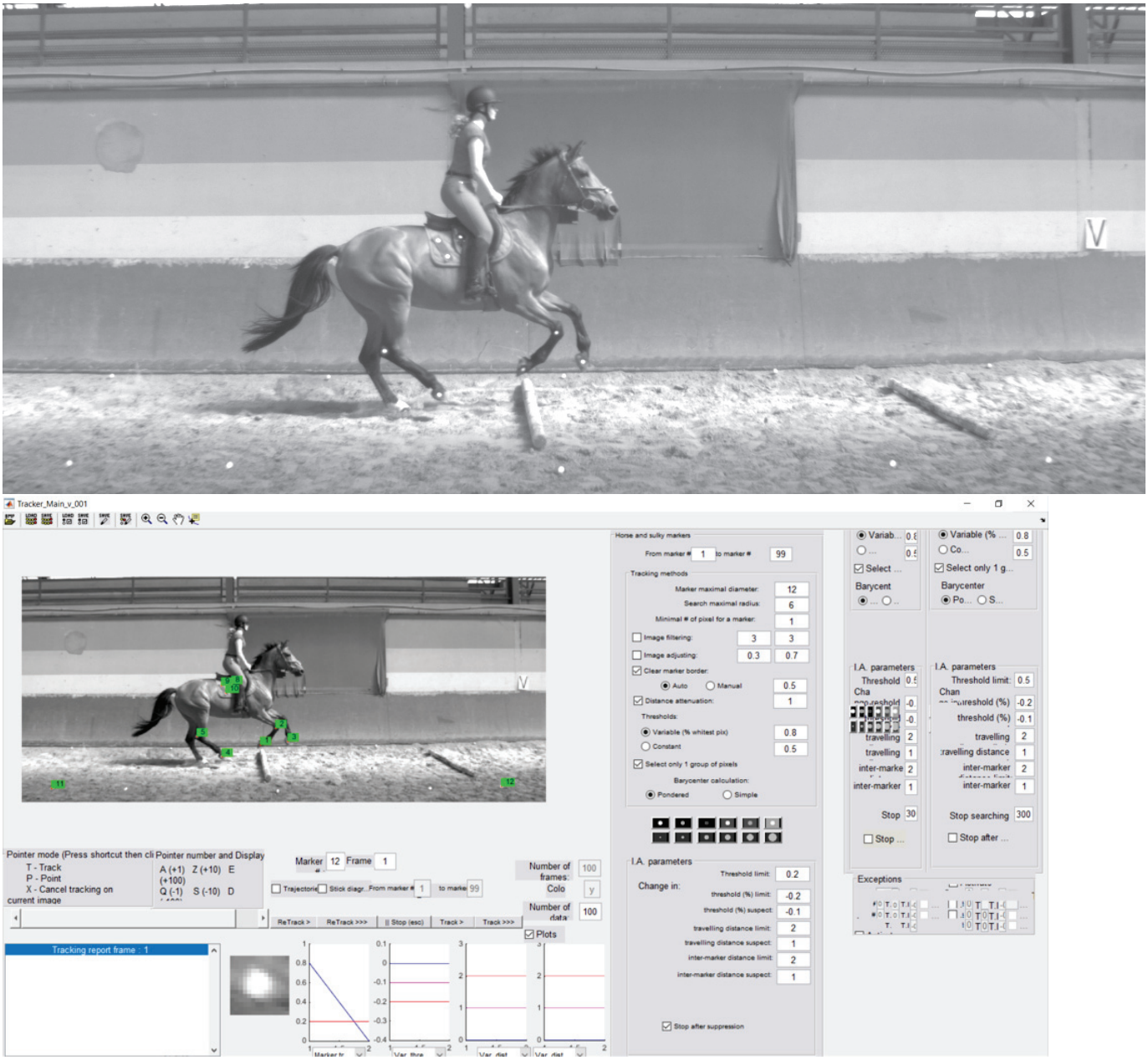


Figure 1.7: Image avant traitement dans le logiciel de tracking (en haut) et suivi manuel des marqueurs réfléchissants dans l'interface de Matlab (en bas)

Lorsque l'on observe le signal d'un foulée (cf. Figure 1.8) on peut noter que le nombre de points de la courbe est vitesse dépendant puisque la fréquence d'échantillonage de l'IMU est constante (100 Hz). En effet, pour un cheval qui galope rapidement, le nombre de valeurs enregistrées sur une foulée sera plus petit que pour un cheval qui galope lentement. Or, en statistique, il est difficile de gérer des individus, ici une foulée, qui n'ont pas la même longueur. Pour s'affranchir de ce problème de nombre de points variable pour chaque signal en fonction de la vitesse, les données d'une foulée sont découpées à partir du pic maximal détecté sur l'accélération selon Z jusque les 100 points suivants (cf. Figure 1.9) de façon automatique. Ce découpage peut donc attribuer à une "foulée" les données de plusieurs cycles de galop mais permet de conserver de façon indirecte la notion de

durée de la foulée : un cheval qui galope lentement aura moins de cycles qu'un cheval qui galope rapidement avec ce découpage.

Ainsi, il existe une dépendance temporelle entre les foulées d'un même parcours. Comme première approche, nous proposons dans cette thèse d'omettre cette dépendance dans le but de simplifier les approches méthodologiques. De plus, un même cavalier peut aussi réaliser plusieurs parcours et donc se retrouver plusieurs fois dans la base de données finale. Nous ne prendrons pas en compte l'effet cavalier dans les modèles que nous développerons par la suite, car, lors de l'extension de notre modèle à l'application en conditions réelles, nous n'aurons pas de moyen systématique d'associer un parcours à un cavalier si ce dernier ne le renseigne pas.

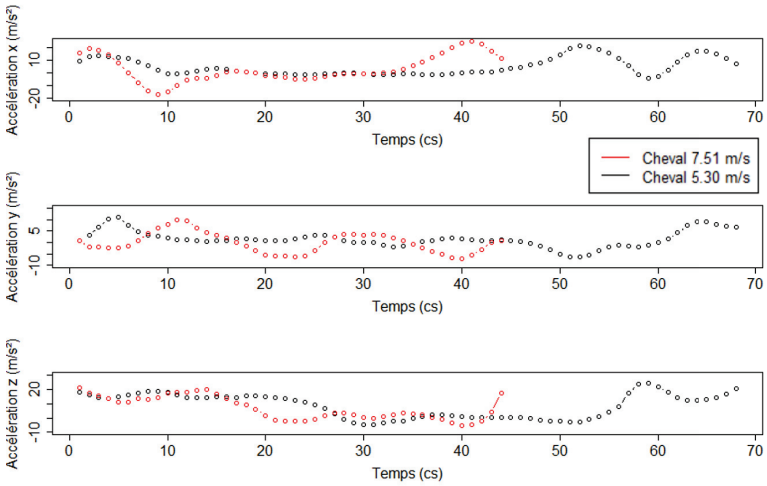


Figure 1.8: Différence de longueur des signaux d'accélération correspondant à une foulée pour un cheval qui galope rapidement (rouge) et un cheval qui galope lentement (noir)

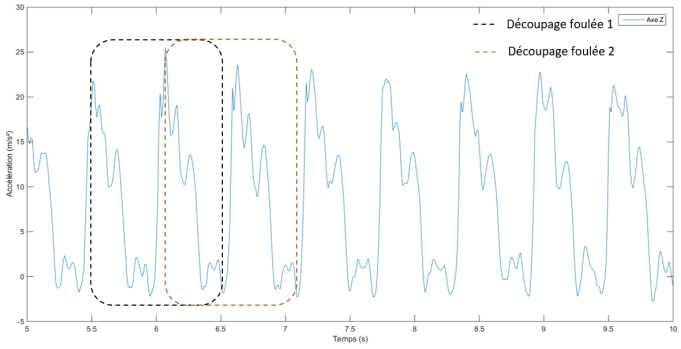


Figure 1.9: Découpage du signal selon l'axe Z par fenêtre glissante à partir du pic de posé de l'antérieur pour définir les "foulées"

Pour finir, le dernier objectif de la thèse est de caractériser la qualité du saut d'obstacle à l'aide des signaux collectés. Afin de répondre à cet objectif, une autre base de données a été constituée à partir de 20 vidéos de parcours d'obstacles qui ont été notées par 10 juges. Ce panel de juges amateurs a été constitué au sein de l'entreprise par des personnes ayant de l'expérience dans la discipline du saut d'obstacles. Chaque juge s'est vu attribuer

10 vidéos selon un plan d'expérience permettant de limiter les biais de notation dont le biais dit "précédent". Ce biais correspond au fait que les notes d'une vidéo peuvent être influencées par la qualité des sauts observés à la vidéo précédente. Les juges ont eu à noter les trois phases de saut (abord, planer, réception) sur une échelle de 1 à 5. La base de données finale correspond aux signaux d'accélération et de vitesse angulaire collectés lors du saut et la note associée à chacun des 699 sauts.

2 Modélisation mathématique

2.1 Les variables descriptives

Nous avons donc pour chaque foulée et chaque signal collecté par l'IMU une série de 101 valeurs :

$$x_{ij}(t_1), ..., x_{ij}(t_{101}) \tag{1.1}$$

où $i \in \{1, ..., 3221\}$ indique l'indice de la "foulée", $j \in \{1, ..., 6\}$ l'indice de la variable mesurée. Pour ce dernier indice, tout au long de ce manuscrit, nous allons adopter les notations suivantes : la première variable mesurée est l'accélération selon l'axe X (Ax), la seconde l'accélération selon l'axe Y (Ay), la troisième l'accélération selon Z (Az), la quatrième la vitesse angulaire selon l'axe X (Gx), la cinquième la vitesse angulaire selon l'axe Y (Gy) et la sixième la vitesse angulaire selon l'axe Z (Gz).

Il existe différentes façons de considérer cette série de données.

- La première façon, appelée statistique multidimensionnelle, est de considérer ces données comme un vecteur en dimension 101. La principale limite de cette approche est que l'on néglige la corrélation temporelle des données pour un individu i : en effet il existe une corrélation naturelle entre $x_{ij}(t)$ et $x_{ij}(t+1)$ qui n'est pas prise en compte si on le considère comme un vecteur multivarié quelconque. Néanmoins, cette approche a l'avantage de permettre d'utiliser toute la panoplie des modèles classiques d'apprentissage statistique.

- La seconde approche possible et de prendre en compte l'aspect temporel en considérant ces données comme séries temporelles à temps discret. La dépendance temporelle est alors prise en compte, mais nous devons nous concentrer sur des techniques d'apprentissage spécifiques aux séries temporelles.

- La dernière façon d'appréhender ces données est l'approche des données fonctionnelles. Cette approche implique qu'un individu, dans notre cas une "foulée", est représenté par un objet de dimension infini : une courbe.

  On appelera variable aléatoire fonctionnelle tout processus stochastique $X = \{X(t), t \in [0, T]\}$ à valeur dans un espace de dimension infini, que l'on supposera généralement être $L_2([0, T])$. Une telle variable fonctionnelle peut être par exemple l'accélération mesurée sur l'intervalle de temps $[0, T]$. On peut également rencontrer des surfaces aléatoires comme par exemple les niveaux de gris d'une image, dans ce cas $T \subset \mathbb{R}^2$. Dans le cadre de ce manuscrit, nous ne traiterons que le cas des données fonctionnelles s'exprimant dans un domaine à une dimension.

  Une réalisation x de X est appelée donnée fonctionnelle. Comme précédemment, on définit $x = \{x(t), t \in [0, T]\}$ [Ferraty and Vieu, 2006a]. A noter qu'en pratique,

on n'observe jamais totalement $x(t)$ mais uniquement ses valeurs en un nombre de points temporels finis : $x = \{x(t_j), j = 1, ..., q, q < \infty\}$.

Les avantages de cette approche sont : la conservation de la dynamique temporelle mais aussi la gestion de pas de temps irréguliers, le lissage des données discrétisées dans une base de fonctions permettant de débruiter les données et le calcul de dérivée permettant de mettre en évidence des variations subtiles dans les données. C'est pourquoi nous avons choisi de poursuivre avec cette approche pour cette thèse.

Ainsi, avec l'analyse fonctionnelle, on distingue les données univariées, où un individu est égal à une courbe, $X_i(t) = X_i^1(t)$, des données multivariées où un individu est égal à plusieurs courbes collectées simultanément. C'est le cas des données de notre étude où une "foulée" i est caractérisée par 6 variables fonctionnelles $\boldsymbol{X}_i(t) = (X_i^1(t), ..., X_i^6(t))' \in \mathbb{R}^6$, correspondant aux trois accélérations et aux trois vitesses angulaires collectées par l'IMU au cours du temps.

2.2 Les variables à prédire

Dans un premier temps, on cherche à construire un modèle de prédiction du scalaire $y \in \mathbb{R}$ correspondant respectivement à la vitesse ou la longueur de foulée.

Dans un second temps, on cherche un modèle de prédiction du scalaire y représentant la qualité du saut. C'est une variable ordinale prenant ses valeurs sur une échelle à 5 niveaux indiquant si le saut est très mauvais (1), moyen (3) ou très bon (5).

Chapitre 2

Etat de l'art en analyse de données fonctionnelles

Dans ce chapitre, nous allons présenter les différentes méthodes de représentation des données fonctionnelles et les pré-traitements que l'on peut réaliser sur ces données avant de les analyser. Puis, nous verrons les méthodes d'exploration classiques des données fonctionnelles que sont l'analyse en composantes principales fonctionnelle, permettant une représentation des principales sources de variabilité présentes dans les données, et les méthodes de clustering fonctionnel, qui permettent de découper une base de données en des sous-groupes homogènes, facilitant son interprétation. Puis nous présenterons les méthodes de régression pour prédire une réponse scalaire à partir de prédicteurs fonctionnels. Et enfin, nous étudierons le modèle de régression le plus classique de prédiction d'une variable catégorielle : la régression logistique fonctionnelle.

Les codes R ayant permis la réalisation des exemples de ce chapitre sont disponibles en Annexe A.

1 Représentation des données fonctionnelles et analyses descriptives

1.1 Représentation des données

Comme présenté dans l'introduction, l'expression fonctionnelle des courbes $x_i(t)$ n'est pas connue en pratique et nous avons uniquement accès aux observations discrètes à des temps précis : $x_i(t_1), ..., x_i(t_j), i \in [1, N], j = 1, ..., q$, $q < \infty$. Notons que, dans le cas général, le pas de temps entre deux mesures n'est pas nécessairement constant et peut même différer entre deux individus.

Il est nécessaire de reconstruire la forme fonctionnelle des données à partir des observations discrètes, pour cela il est classique d'utiliser une combinaison linéaire de fonctions de base dont les coefficients sont estimés par interpolation ou par lissage [Ramsay and Silverman, 2005]. Une seconde approche, basée sur l'expression des données à l'aide de semi-métriques [Ferraty and Vieu, 2006a], est aussi possible. Ces deux approches seront développées ci-dessous.

1.1.1 Représentation dans une base de fonctions

Le système de fonctions de base est un ensemble de fonctions connues ϕ_r qui ont la propriété de pouvoir approximer arbitrairement toutes les fonctions en prenant une combinaison linéaire d'un nombre R suffisamment grand de ces bases. L'expression dans une base de fonction se construit en deux étapes :

- On définit un ensemble de blocs de construction appelés fonctions de base, noté $\phi_r(t)$, avec $1 \leq r \leq R$ le nombre de fonctions de base choisies par l'utilisateur.
- On construit un vecteur de coefficients pour définir la courbe comme une combinaison linéaire des fonctions de base, noté c_r.

On fait ainsi l'hypothèse qu'une variable fonctionnelle est une combinaison linéaire d'un système de fonctions de base et d'un ensemble de coefficients :

$$X_i(t) = \sum_{r=1}^{R} c_{ir}\phi_r(t). \tag{2.1}$$

avec ϕ_r la matrice de fonctions de base, $c = (c_{ir})_{1 \leq i \leq N, 1 \leq r \leq R}$ la matrice de coefficients organisée de la façon suivante :

- si R fonctions de base et un seul individu (dans le cas d'une seule courbe, N=1) on a une matrice d'une colonne et R lignes.
- si R fonctions de base et n individus on a une matrice de n colonnes et R lignes.

Ces coefficients peuvent être estimés par différentes méthodes.

La plus simple est par interpolation. Elle peut être réalisée quand $R = n$ dans le sens où nous pouvons choisir les coefficients c_r pour obtenir $x(t_j) = y_j, \forall j$. L'interpolation est appliquée lorsque l'on considère que les mesures sont faites sans erreur. C'est la méthode la plus basique pour estimer la valeur prise par une fonction continue entre deux points. Elle consiste à utiliser une fonction affine passant par les deux points déterminés. Cette approche est peu recommandée en pratique. [Ramsay et al., 2009, Ramsay and Silverman, 2005]

Une seconde méthode d'estimation est la méthode de lissage par les moindres carrés [Ramsay and Silverman, 2005, Bouveyron et al., 2015]. Le lissage permet de prendre en compte le fait qu'une donnée comprend une erreur de mesure qui nécessite d'être retirée. L'objectif est de modéliser les observations discètes h_i en utilisant le modèle $h_i = x(t_i) + \epsilon_i$. Cette méthode est utilisée lorsque l'on fait l'hypothèse que les erreurs ϵ associées au modèle sont indépendantes, identiquement distribuées, de moyenne zéro et de variance constante. Le lissage linéaire est obtenu si l'on détermine les coefficients c_r en minimisant le critère des moindres carrés :

$$SMSSE(h|c) = \sum_{i=1}^{n} [h_i - \sum_{r=1}^{R} c_r \phi_r(t_i)]^2$$

ou d'un point de vue matriciel :

$$SMSSE(h|c) = {}^t(h - \phi c)(h - \phi c).$$

Si l'on calcule la dérivée du critère $SMSEE(h|c)$ par rapport à c, on obtient l'équation :

$$2\phi^t \phi c - 2\phi h = 0.$$

La résolution de cette équation pour c nous donne l'estimateur $\hat{c}$ qui minimise la solution :

$$\hat{c} = ({}^t\phi\phi)^{-1t}\phi h.$$

Quand les hypothèses du modèle précédent ne semblent pas réalistes, dans le cas par exemple d'erreurs non stationnaires ou auto-corrélées, il faut amener une pondération différentielle des résidus. Cette approche est détaillée dans [Ramsay and Silverman, 2005].

Le choix du nombre de fonctions de base R est plus empirique. Plus le nombre de fonctions de base est important, plus la courbe de lissage va être proche des données, mais risque aussi de modéliser du bruit. Un nombre de bases trop faible pourrait lisser des aspects importants de la courbe que l'on cherche à estimer.

Ces fonctions de base peuvent être réparties en deux catégories : périodiques et non périodiques. Le système de bases de Fourier est le choix classique pour les fonctions périodiques, le système de bases splines (et B-splines) tend à bien fonctionner pour les

fonctions non périodiques. Ces systèmes peuvent être complétés par les systèmes de bases constantes (ex : 1), monomiales (ex : $1, t, t^2, t^3$...) ou encore à ondelettes. Ces dernières peuvent être utilisées quand on souhaite conserver de l'information à plus petite échelle qu'avec les splines. Dans la suite de ce document, nous allons détailler le cas des bases de Fourier et des bases splines car se sont les deux systèmes de bases les plus communéments utilisés et qui illustrent d'un point de vue calculatoire le cas d'une base de fonction orthogonale et d'une base non orthogonale.

Les séries de Fourier sont définies par $\phi_1(t) = 1$, $\phi_2(t) = sin(\omega t)$, $\phi_3(t) = cos(\omega t)$, $\phi_4(t) = sin(2\omega t)$, ... où $\omega = \frac{2\pi}{T}$. Les deux seuls éléments nécessaires à la définition d'un système de bases de Fourier sont le nombre de fonctions de base R et la période T. Si le pas de temps est constant, cette base est dite orthogonale car le produit des matrices ${}^t\boldsymbol{\phi}\boldsymbol{\phi}$ donne une matrice diagonale.

Le système de bases splines est construit en divisant l'intervalle des observations en sous-intervalles avec comme frontières les points appelés "points de rupture". Sur chaque sous intervalle, la fonction spline est un polynôme de degré fixé, mais la nature du polynôme change lors du passage au sous-intervalle suivant. Il y a une contrainte de continuité sur les bords des intervalles, il faut que les premières dérivées soient continues et égales aux bords des intervalles. L'ordre du polynôme correspond à son degré augmenté de 1, le degré faisant référence à la plus grande puissance du polynôme. L'ordre du polynôme est le même pour chaque sous-intervalle.

Pour choisir l'ordre des splines, il est souvent recommandé de choisir le plus grand ordre de dérivée que l'on va étudier augmenté de deux. Par contre, si l'on ne souhaite pas regarder les dérivées d'une variable fonctionnelle, le lissage à l'aide d'une base de splines cubiques est la plus classiquement utilisée.

1.1.2 Représentation à l'aide de semi-métriques

Soit d une semi-métrique de l'espace F tel que :

- $\forall x \in F, d(x, x) = 0$
- $\forall (x, y, z) \in F \times F \times F, d(x, y) \leq d(x, z) + d(z, y)$

Une semi-métrique d est une métrique dont $d(x, y) = 0 \not\Rightarrow x = y$.

Les semi-métriques peuvent être vues comme des outils exploratoires. Une grande partie de ces outils consistent à représenter les données dans un espace de dimension réduite. On peut choisir par exemple de mesurer la proximité entre sujets à l'aide d'une Analyse en composantes principales (ACP), cette proximité est alors calculée à l'aide de la métrique L_2. Comme notre travail porte sur les données fonctionnelles en tant que courbes, nous allons détailler ici uniquement les semi-métriques bien adaptées aux courbes. Nous présenterons les semi-métriques ACP et PLS qui sont bien adaptées pour les courbes de forme irrégulière et la semi-métrique basée sur les dérivées dans le cas des courbes à aspect lisse.

L'ACP fonctionnelle est un bon outil pour calculer la proximité entre courbes dans un espace de dimension réduite quand les courbes sont observées sur un même nombre de points et si la grille de mesure est suffisamment fine. Cette distance peut être approximée empiriquement par :

$$d_l^{PCA}(x_i, x_{i'}) = \sqrt{\sum_{k=1}^{l} (\sum_{j=1}^{q} w_j (x_i(t_j) - x_{i'}(t_j))[v_k]_j))^2}$$

avec x_i et $x_{i'}$ deux courbes discrétisées, l le nombre de dimensions du sous-espace, v_k les vecteurs propres W-orthonormaux associés à la matrice de covariance, $W = diag(w_1, ..., w_q)$ et $w_j = t_j - t_{j-1}$.

Une autre famille de semi-métrique possible a été développée en adaptant la régression des moindres carrés partiels multivariée (MPLSR). La MPLSR a été développée pour prédire une réponse multivariée à partir de variables indépendantes quand il y a une forte colinéarité parmi les prédicteurs ou quand le nombre de prédicteurs est grand par rapport au nombre d'observations. Contrairement à l'ACP où seuls les prédicteurs sont représentés, l'approche PLS construit des composantes en lien avec la réponse multivariée. Soit $v_1^l, ..., v_n^l$ les vecteurs de $\mathbb{R}^q$ calculés par MPLSR où l correspond au nombre de facteurs et n au nombre de réponses scalaires. On peut alors définir la semi-métrique :

$$d_l^{PLS}(x_i, x_{i'}) = \sqrt{\sum_{k=1}^{n}(\sum_{j=2}^{q} w_j(x_i(t_j) - x_{i'}(t_j))[v_k^l]_j)^2}$$

avec w_j des pondérations quadratiques.

La dernière semi-métrique que nous présenterons ici est la distance par rapport à une des dérivées de la courbe. Pour s'absoudre du coût computationnel que représente le calcul de dérivées successives, les données discrétisées sont exprimées dans une base B-spline avant le calcul des dérivées. Comme les données sont lissées dans une base de fonctions, cette métrique peut être utilisée dans le cas de données où le nombre de points de mesure diffère d'une courbe à l'autre ou lorque les temps de mesure diffèrent d'une courbe à l'autre. Mais cette dernière est peu adaptée dans le cas de courbes de forme très irrégulières.

Ces lissages seront utilisés lorsque nous testerons la fonction de régression non paramétrique proposée par Ferraty et Vieu.

Exemple Prenons les données d'accélération selon l'axe Z, collectées par la centrale inertielle placée sur le garrot d'un cheval au galop. Le motif pour une foulée n'étant pas régulier (contrairement à une fonction sinus ou cosinus par exemple), nous avons choisi d'appliquer un lissage avec des splines cubiques. La Figure 2.1 présente un extrait des données brutes, puis ces données lissées avec 15 splines cubiques et avec 50 splines cubiques.

On observe ainsi que la forme des courbes est plus grossière dans le cas d'un lissage avec 15 splines, et plus fin dans le cas d'un lissage avec 50 splines.

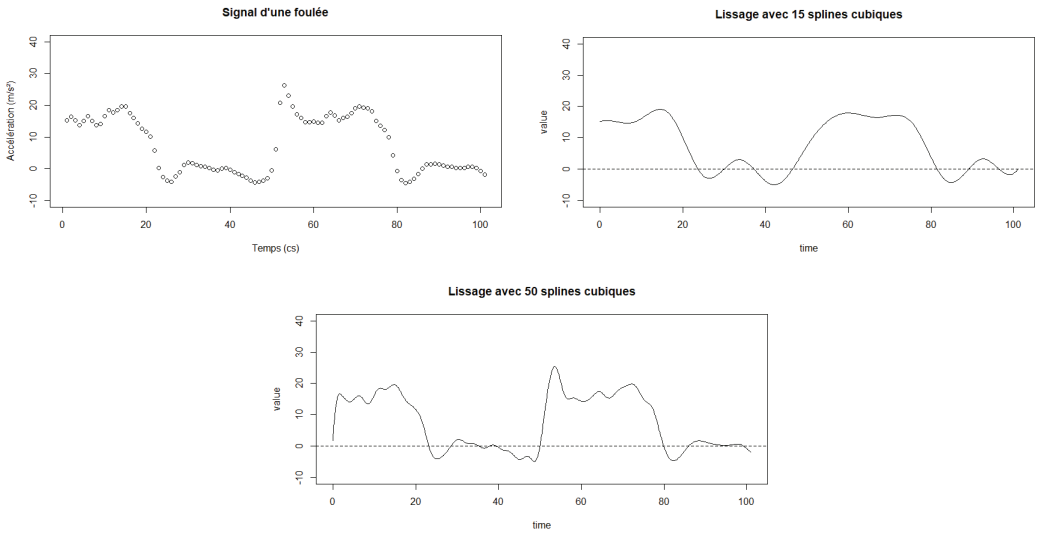


Figure 2.1: Signal brut d'une foulée d'accélération selon Z (en haut à gauche), signal après lissage avec 15 bases splines (en haut à droite) et 50 bases splines (en bas)

1.2 Alignement des données

L'alignement consiste à transformer l'échelle de temps pour chaque courbe observée afin de comparer des évènements identiques. Prenons par exemple les courbes de croissances d'enfants âgés de 4 à 18 ans. L'apparition du pic de croissance n'a pas lieu au même âge pour tous les enfants. Ne pas en tenir compte avant le calcul de la fonction moyenne va mener à une courbe moyenne qui ne ressemblera à aucune des courbes individuelles observées [Ramsay and Silverman, 2005]. Dans ce type de situation, une étape d'alignement des données par rapport à ce pic de croissance avant calcul de la moyenne est important.

Lorsque l'on analyse des courbes, on distingue deux grands types de variation :

- la variation d'amplitude qui est due au fait que deux courbes $x_1(t)$ et $x_2(t)$ diffèrent aux temps t où on les compare mais ont le même aspect,

- la variation de phase où les courbes $x_1(t)$ et $x_2(t)$ ne peuvent pas être comparées pour des temps t précis car elles ne présentent pas le même comportement. Pour pouvoir les comparer, il faut que l'échelle de temps soit transformée.

Il est donc intéressant d'aligner les données quand la métrique du temps n'est pas directement pertinente pour expliquer la dynamique interne de systèmes réels. Ce calibrage des données doit être effectué après l'étape de lissage pour limiter au maximum l'impact du bruit dans l'alignement des courbes.

Dans la suite de cette section, nous allons présenter les méthodes d'alignement global par décalage de l'échelle de temps et la méthode d'alignement par rapport à des points de repère.

La plupart des problèmes d'alignement peuvent être résolus par un décalage de l'échelle de temps. Soit $\mathcal{T} = [T_1, T_2]$ l'intervalle sur lequel les courbes doivent être alignées, cet intervalle ne correspond pas forcément à la totalité de la période de mesure. On suppose que chaque courbe x_i est disponible au delà de cet intervalle, et l'on s'intéresse aux valeurs $x_i^*(t) = x_i(t+\delta_i)$ où $x_i^*(t)$ correspond aux valeurs de l'individu i après alignement et $\delta_i \in \mathbb{R}$

correspond au paramètre de décalage qui est choisi de façon à aligner convenablement les courbes. Ce δ_i peut être une valeur fixée de façon à ce que toutes les courbes soient alignées par rapport à une caractéristique comme par exemple le pic de croissance. Mais il peut être difficile de déterminer avec précision la valeur de δ_i si la position de la caractéristique d'alignement est ambigue pour certaines courbes. Dans ce cas, on définit un critère d'alignement global de la façon suivante :

- calcul de la fonction moyenne $\hat{\mu}(t)$ pour chaque $t \in [T_1, T_2]$
- on définit le critère d'alignement à minimiser comme étant :

$$REGSSE = \sum_{i=1}^{N} \int_{\mathcal{T}} [x_i(t + \delta_i) - \hat{\mu}(t)]^2 ds$$

On procède ensuite de façon itérative : après avoir ré-aligné les courbes, on recalcule la fonction moyenne puis on choisit de nouvelles valeurs de δ_i pour minimiser *REGSSE* et ce jusqu'à convergence.

La seconde méthode est l'alignement à partir de points de repère. les points de repère sont un ensemble de points caractéristiques d'une courbe comme par exemple la position de maximas, minimas ou de points d'inflexion qui peuvent être identifiés au niveau des dérivées des courbes par exemple. Cette méthode vise à calculer une transformation non linéaire h_i du temps t qui met en correspondance des points caractéristiques qui sont communs à un ensemble de signaux devant être alignés. Les points de repère qui caractérisent les mêmes structures se trouvent ainsi à la même position après déformation.

Ce processus nécessite donc que, pour chaque courbe x_i, on identifie les valeurs t_{if}, $f = 1, ..., F$ associées aux points de repère puis que l'on construise une transformation h_i pour chaque courbe telle que $x_i^*(t) = x_i[h(t)]$. Pour finir, il faut calculer les valeurs des fonctions déformées. Ce calcul se fait en deux étapes :

- estimer la fonction inverse de la déformation temporelle $h^{-1}(t)$ d'après la propriété $h^{-1}[h(t)] = t$. La fonction $h^{-1}(t)$ est calculée par interpolation à partir de la relation entre $h(t)$ et t,
- interpoler la relation entre $h^{-1}(t)$ et $x(t)$.

Exemple La figure 2.2 représente graphiquement le processus d'alignement global d'une courbe d'accélération d'une foulée selon Z par rapport à la courbe moyenne de ces foulées. Suite à cette procédure, le premier pic de la courbe de la foulée concorde avec le premier pic de la courbe moyenne apparaissant à -14.

1.3 Statistiques descriptives

Dans toute analyse de donnée, le premier réflexe est d'estimer la moyenne et l'écart type. Pour les données fonctionnelles, celles-ci peuvent être calculées à partir des formules ci-dessous.

La fonction moyenne empirique de la variable x s'exprime :

$$\bar{x}(t) = \frac{1}{n} \sum_{i=1}^{n} x_i(t)$$

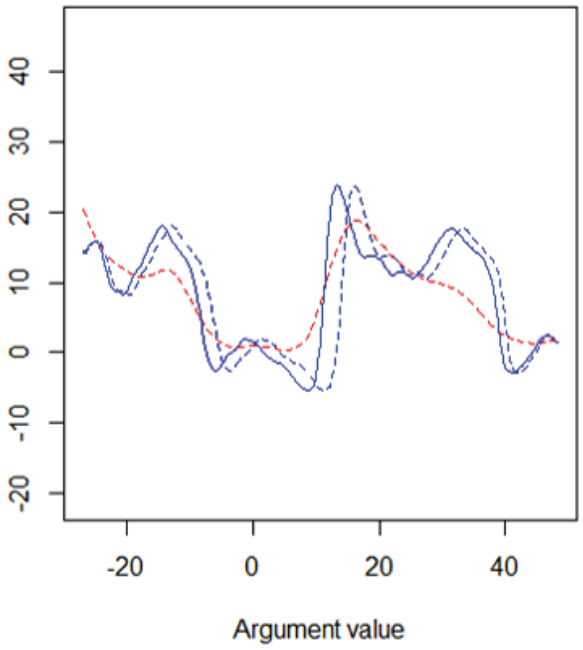


Figure 2.2: Courbe d'une foulée d'accélération selon Z avant (bleu pointillé) et après (bleu trait plein) réalignement par rapport à la courbe moyenne de toutes les foulées (rouge)

Dans le cadre de nos données, l'individu i correspond à une "foulée".

Et la fonction de variance empirique s'écrit :

$$s_x^2(t) = \frac{1}{n-1}\sum_{i=1}^{n}[x_i(t) - \bar{x}(t)]^2$$

2 L'Analyse en Composantes Principales Fonctionnelle (FPCA)

L'analyse en composantes principales fournit un moyen d'observer la structure de covariance des données et donc les principales sources de variation observées dans les données [Ramsay and Silverman, 2005, Jacques and Preda, 2014b]. Dans le cas des données fonctionnelles, elle s'applique sur les données lissées dans une base de fonctions. On suppose donc que nos courbes univariées peuvent être décomposées dans un espace de dimension finie par le biais d'une base de fonctions telles que définie dans l'équation 2.1.

Soit $X_1, ..., X_n$ un échantillon i.i.d. de X et X un processus stochastique continu L_2. La transformation de Karhunen-Loève de X s'écrit :

$$X(t) = \mu(t) + \sum_{l\geq 1} \nu_l f_l(t). \tag{2.2}$$

où f_l est la base orthonormale de fonctions propres. Ces fonctions propres sont les solu-

tions de la décomposition spectrale de l'opérateur de covariance ν :

$$\begin{aligned} \nu : L_2([0,T]) &\to L_2([0,T]) \\ f &\xrightarrow{\nu} \nu f \\ \nu f_l = \lambda_l f_l, \forall l &\geq 1 \end{aligned}$$

avec λ_l un ensemble de valeurs propres positives. Les composantes principales, C_l, sont des variables aléatoires de moyenne zéro et de variance λ_l, correspondant à la projection de X sur les fonctions propres de ν :

$$C_l = \int_0^T \langle X(t) - \mu(t), f_l(t) \rangle dt$$

Pour résoudre cette équation, on doit généralement approximer chaque courbe dans une base de fonctions selon l'équation 2.1. Suite à cette approximation, on peut écrire l'estimateur de l'opérateur de covariance comme étant :

$$\hat{\nu}(s,t) = \frac{1}{n-1} X'(s) X(t).$$

On suppose que chaque fonction propre f_l appartient à l'espace linéaire exprimé avec la matrice ϕ contenant les bases de fonction $\{\phi_1, ..., \phi_R\}$:

$$f_l(t) = \phi(t)^t b_l$$

où b_l peut être approximé par :

$$b_l = W^{1/2} f_l$$

et $W = \int_0^T {}^t\phi(t)\phi(t)$. Les individus peuvent alors être identifiés dans le sous-espace de dimension réduite par leurs scores obtenus à partir du produit scalaire :

$$\delta_{il} = \int_0^T \langle c_i, b_l \rangle dt.$$

Exemple Nous appliquons l'ACPF sur la base de données d'accélération selon Z pour chaque foulée (cf. Figure 2.3).
Les fonctions propres représentent les principaux modes de variation des données (cf. Figure 2.4). Dans le cas de notre exemple, le pourcentage de variance associé à la première fonction propre (appelée aussi harmonique) est 38% et 17% pour la seconde. Il peut être intéressant de représenter ces principales sources de variation directement par rapport à la fonction moyenne. Ces variations se calculent selon la formule :

$$\bar{x}(t) \pm \sqrt{\lambda_k} f_k(t)$$

avec λ_k la k-ième valeur propre et f_k la k-ième fonction propre [Ramsay and Silverman, 2005].

Les trois premiers graphes de variation de la fonction moyenne (cf. Figure 2.5) montrent que les principales sources de variabilité des données sont la hauteur des pics. Alors que le quatrième graphe montre qu'il peut y avoir un léger décalage de phase entre les courbes. Notre découpage en "foulée" comprenant plusieurs cycles de galop, le premier

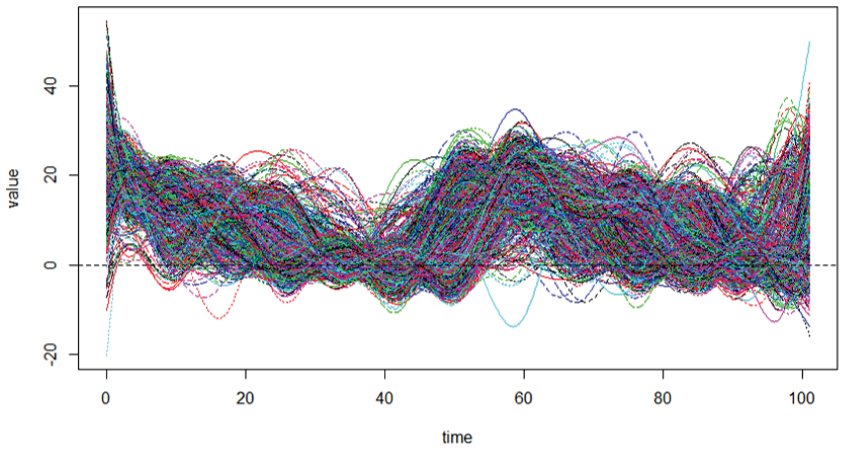


Figure 2.3: Représentation de l'ensemble des données lissées, une courbe correspondant à une "foulée" d'accélération selon Z

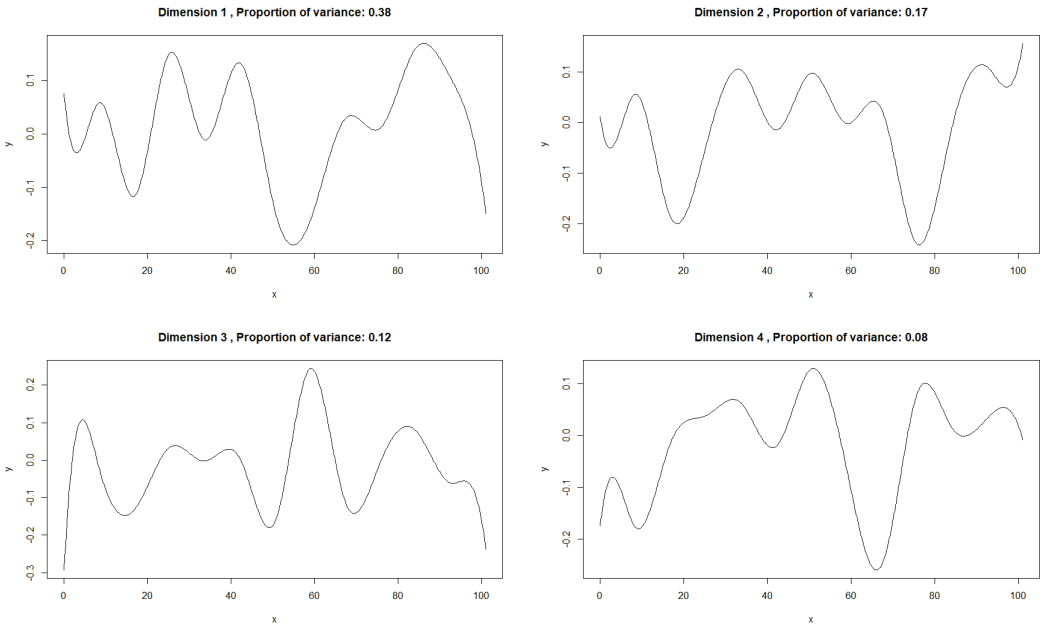


Figure 2.4: Représentation des 4 premières fonctions propres

pic (au temps 0) correspond au moment où l'antérieur directeur touche le sol, le second pic (au temps 20) correspond à la phase de suspension où tous les membres sauf l'antérieur directeur ont quitté le sol et où tout le poids est sur l'antérieur directeur et enfin la phase plane (entre les temps 35 et 45) correspond à la phase de projection du galop (cf. Figure 2.6). Puis une nouvelle foulée commence. La taille des pics est cheval dépendante et semble aussi être impactée par l'allure. En effet, nous avons observé que, selon si nous demandions au cavalier un galop allongé ou un galop rassemblé, l'amplitude des pics variait, mais aucune étude approfondie n'a été menée sur ce sujet.

On peut ensuite étudier la forme du nuage des scores. Ces scores sont les coordonnées des individus sur les harmoniques. L'étude des scores permet parfois de distinguer des groupes d'individus ou des individus atypiques. Dans le cas de ces données, on n'observe

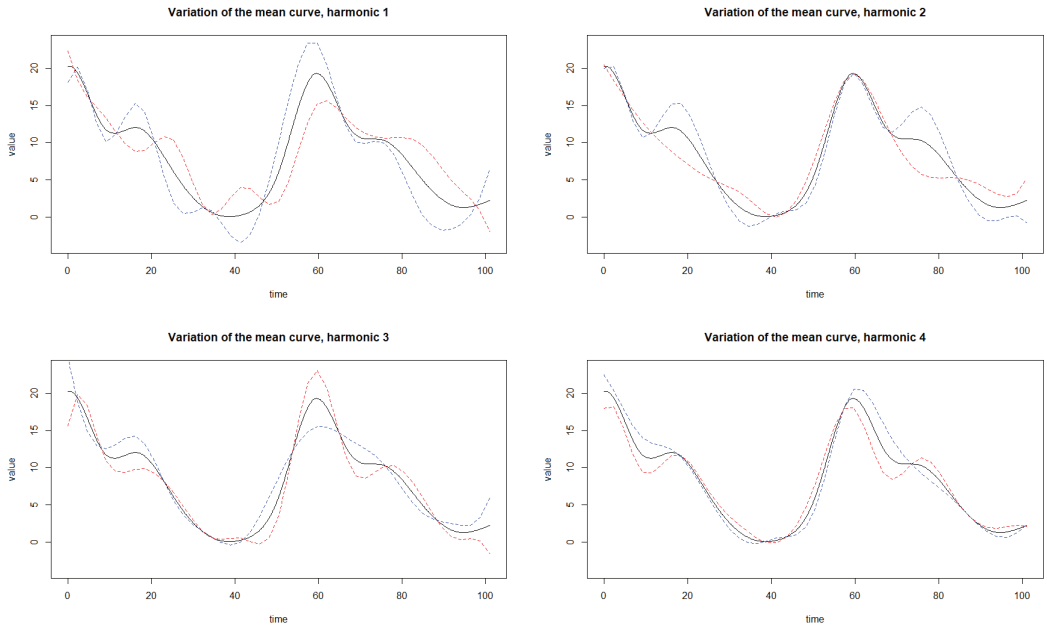


Figure 2.5: Variations de la fonction moyenne (noire) pour les 4 premières harmoniques. Les courbes bleues et rouges représentent les amplitudes de variation de la courbe moyenne.

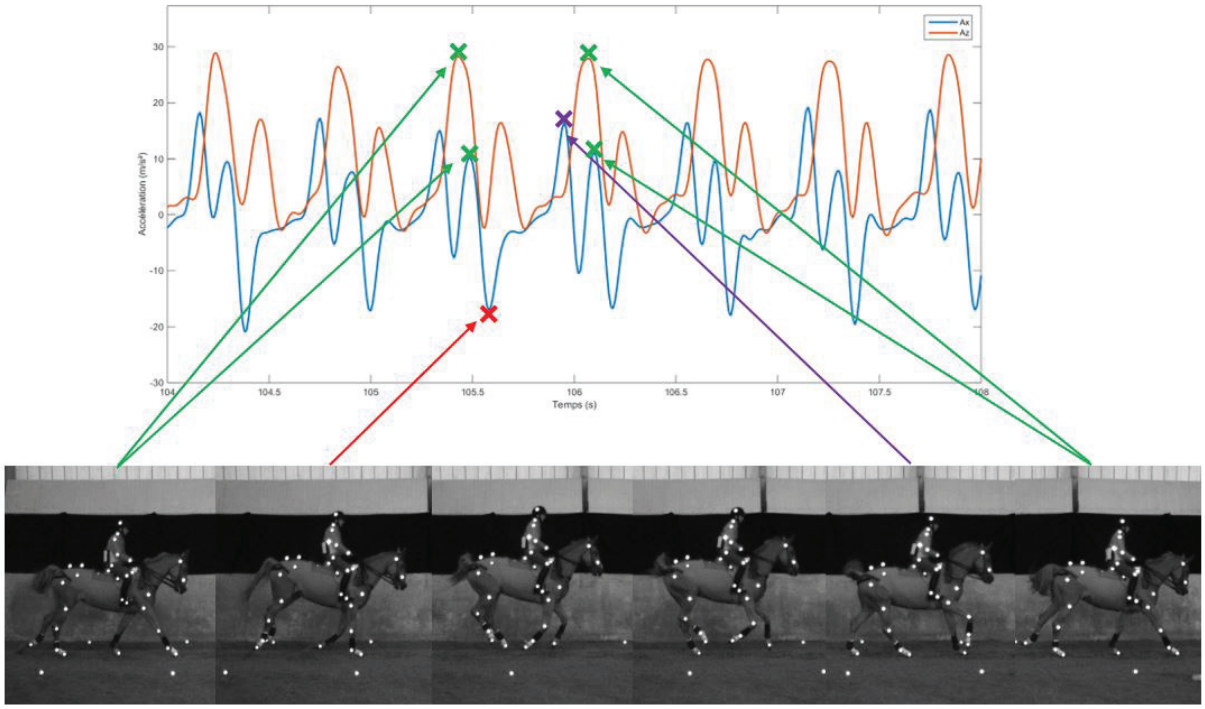


Figure 2.6: Décomposition des courbes d'accélération Ax (en bleu) et Az (en orange) à l'aide des images collectées par le système de capture 2D

pas de groupes nets, mais un noyau central et quelques "foulées" atypiques comme les foulées 2629 à la 2636 (cf. Figure 2.7). Ces foulées correspondent au passage d'un même cheval qui était agacé par le dispositif mis en place et contraint par son cavalier pour qu'il

ne change pas de pied entre les deux obstacles installés, ce qui pourrait expliquer une forme atypique de ces foulées où il montait la croupe de façon marquée par rapport à ses autres passages.

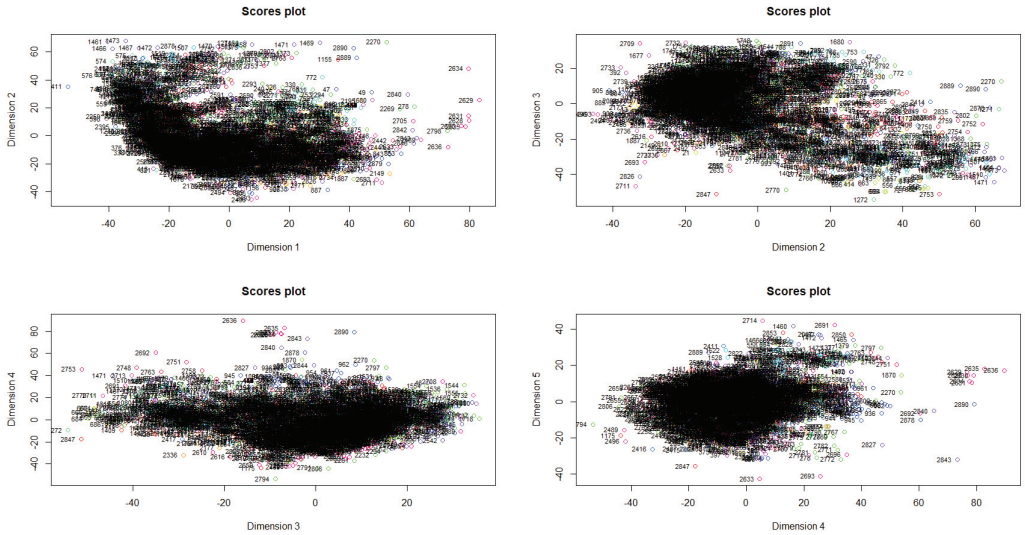


Figure 2.7: Représentation des scores pour les 4 premières harmoniques, chaque point correspondant à une foulée et colorés par cheval

3 Clustering fonctionnel

Le but du clustering est d'identifier des groupes d'observations homogènes, ces observations étant la représentation de l'effet d'une variable aléatoire X. [Jacques and Preda, 2014a] distinguent quatre grands groupes de méthodes de clustering fonctionnel :

- Les méthodes qui travaillent directement sur les points discrétisés. [Boullé, 2012, Bouveyron and Brunet, 2013]
- Les méthodes de filtrage où il y a une première étape de lissage des données dans une base de fonctions et une seconde étape de classification sur les coefficients de la base. [Abraham et al., 2003, Peng and Müller, 2008, Kayano et al., 2010]
- Les méthodes adaptatives qui réalisent simultanément une réduction de la dimension et une classification en s'appuyant sur des modèles de type probabilistes. [James and Sugar, 2003, Chiou and Li, 2007, Bouveyron and Jacques, 2011, Jacques and Preda, 2014b]
- Les méthodes basées sur un calcul de distance pour la classification. [Ieva et al., 2011, Tarpey and Kinateder, 2003, Ferraty and Vieu, 2006a]

Nous allons détailler ici les méthodes adaptatives dont nous nous sommes inspirés pour développer notre modèle de clustering qui sera présenté au Chapitre 3. Dans [James and Sugar, 2003] les données sont exprimées dans une base de fonctions splines. Les coefficients dans cette base sont supposés être distribués selon un mélange de Gaussiennes. Ils suggèrent que leur méthode peut être particulièrement utilisée dans le cas où l'on a peu de points de mesure.

[Bouveyron and Jacques, 2011] et [Jacques and Preda, 2014b] proposent un modèle avec réduction de la dimension basé sur un modèle de mélange latent fonctionnel qui répartit les données dans des sous-espaces fonctionnels spécifiques aux groupes. Pour cela, les données sont exprimées dans une base de fonctions puis une ACP fonctionnelle est réalisée pour chaque groupe en pondérant les courbes par leur probabilité d'appartenance au groupe. Les scores obtenus sont considérés comme des variables aléatoires dont la distribution de probabilité est spécifique au groupe. Le nombre de dimensions du sous-espace fonctionnel est choisi à l'aide du scree-test de Cattell [Cattell, 1966]. La principale différence entre ces deux méthodes est que dans l'algorithme funClust [Jacques and Preda, 2014b] seules les premières composantes principales sont conservées dans l'analyse alors que dans la méthode funHDDC de [Bouveyron and Jacques, 2011], toute l'information est conservée, car les dernières valeurs propres, considérées comme étant associées à du bruit, sont moyennées et conservées dans l'analyse. Ceci permet d'avoir un modèle parcimonieux en terme de nombre de paramètres à estimer, tout en conservant un maximum d'information. De plus, dans ce dernier travail, plusieurs sous-modèles plus parcimonieux sont proposés selon les hypothèses que l'on veut faire sur les données.

Enfin, [Chiou and Li, 2007] présentent une méthode de clustering fonctionnel non paramétrique, basée sur un algorithme k-means, qui utilise la moyenne et le mode des différences de variation entre les groupes pour prédire le groupe d'appartenance. Cette méthode est un cas particulier de [Bouveyron and Jacques, 2011] et [Jacques and Preda, 2014b] où l'on suppose que la variance des composantes principales est égale au sein d'un groupe et entre les groupes.

Plus récemment, [Bouveyron et al., 2015] présentent un modèle de mélange discriminant fonctionnel, appelé funFEM, qui modélise les données dans un unique sous-espace

fonctionnel discriminant. Pour cela, les données sont exprimées dans une base de fonctions et l'on considère que les coefficients suivent un mélange de gaussiennes. Le modèle d'estimation diffère de celui des deux méthodes précédentes. La première étape consiste à calculer la matrice d'orientation du sous-espace latent discriminant qui permet de séparer le plus les groupes. Ce sous-espace est choisi de façon à ce que la variance intra-groupe soit minimale et la variance inter-groupes soit maximale.

Exemple Nous allons comparer ici les résultats obtenus par les algorithmes funFEM [Bouveyron et al., 2015] et funHDDC [Bouveyron and Jacques, 2011] sur les données d'accélération Az car ces deux algorithmes sont disponibles sur le CRAN. Chaque algorithme est lancé pour un nombre de partitions K allant de 2 à 20. Pour sélectionner la meilleure partition des données, nous utiliserons le Bayesian Information Criterion (BIC, [Schwarz, 1978]) que l'on cherche à maximiser. Le tableau 2.1 résume les résultats de BIC pour chaque modèle. On peut voir que dans le cas des données d'accélération Az, l'algorithme funHDDC a du mal à distinguer des clusters d'individus, contrairement à funFEM dont la meilleure partition est celle en 14 groupes, funFEM a l'air de favoriser le nombre maximal de classes jusqu'à ce que l'algorithme n'arrive plus à converger.

Model	K	BIC
funFEM AkjBk	2	-155459
funFEM AkjBk	3	-150827.2
funFEM AkjBk	4	-148585.2
funFEM AkjBk	5	-147082.7
funFEM AkjBk	6	-145455.8
funFEM AkjBk	7	-144120.2
funFEM AkjBk	8	-143605.5
funFEM AkjBk	9	-142232.3
funFEM AkjBk	10	-141695.3
funFEM AkjBk	11	-141180.9
funFEM AkjBk	12	-140930.4
funFEM AkjBk	13	-140186.6
funFEM AkjBk	**14**	-139547.1
funFEM AkjBk	15	No convergence
funHDDC AkjBkQkDk	**2**	-441458.59
funHDDC AkjBkQkDk	3	NA
funHDDC AkjBkQkDk	4	NA
funHDDC AkjBkQkDk	5	NA
funHDDC AkjBkQkDk	6	NA

Table 2.1: Valeurs de BIC pour la sélection du nombre de groupes funHDDC et funFEM

La Figure 2.8 représente les fonctions moyennes de chaque cluster. On observe que funHDDC distingue les foulées dont le pic de la phase de suspension est marqué, des foulées dont ce pic est plus "plat" et plus rapproché du pic de posé de l'antérieur directeur. Tandis que funFEM distingue 14 groupes en fonction de leurs hauteurs de pic mais aussi en fonction de l'apparition décalée dans le temps de ces pics.

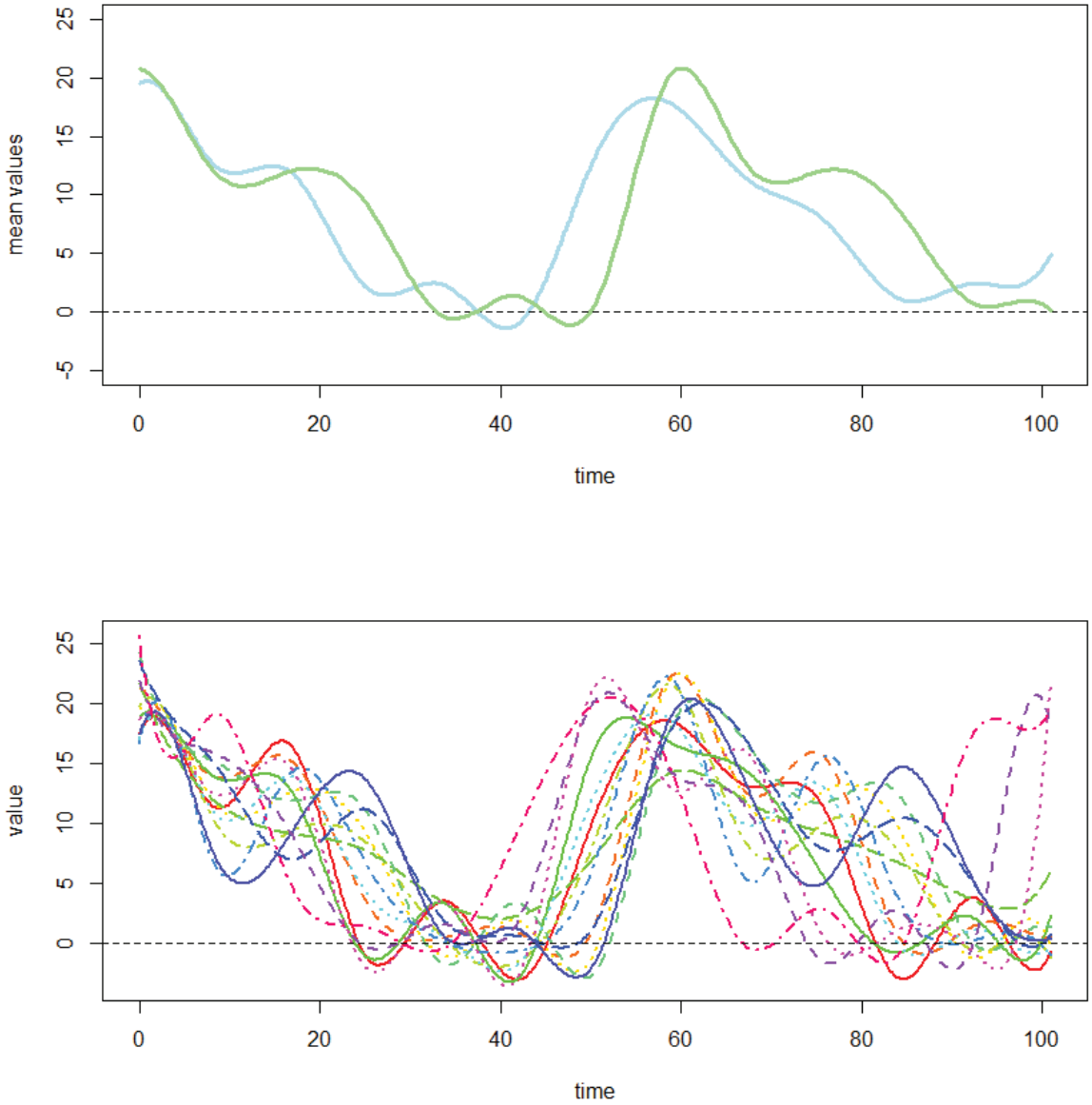


Figure 2.8: Représentation des fonctions moyenne de chaque cluster pour l'algorithme funHDDC (en haut) et funFEM (en bas)

4 Régression pour une réponse scalaire et des prédicteurs fonctionnels

4.1 Modèle paramétrique

Le modèle de régression le plus classique avec un prédicteur fonctionnel peut s'écrire :

$$y_i = \alpha + \int_0^T x_i(t)\beta(t)dt + \epsilon_i.$$

avec α l'intercept qui permet l'ajustement par rapport à l'origine de la variable X, $\beta(t)$ la fonction de coefficient de régression et ϵ_i la fonction résiduelle. [Ramsay and Silverman, 2005] proposent deux méthodes d'estimation par régularisation du paramètre $\beta(t)$: la première utilise des fonctions de base restreintes et la seconde s'appuie sur une méthode de pénalisation irrégulière. Ces méthodes permettent d'éviter des fluctuations locales importantes de la fonction estimée, que l'on aurait en utilisant la méthode des moindres carrés résiduelle pour l'estimation de $\beta(t)$.

Exemple Nous allons étudier l'impact du nombre de bases de fonction utilisées pour le lissage des données sur la qualité de prédiction de ce modèle de régression. Pour cela, nous allons chercher à prédire la vitesse d'une foulée de galop à partir de l'accélération mesurée selon Z. Puis ces valeurs seront comparées aux valeurs de vitesse mesurées par les méthodes de référence (c'est à dire mesurées à partir du système de caméras 2D ou le chronomètre à déclenchement automatique) pour calculer une erreur de prédiction. Nous obtenons les résultats présentés dans le Tableau 2.2. On observe qu'un lissage des données avec un grand nombre de fonctions de base ne permet pas un gain en précision de prédiction important. En effet, un cheval sur un concours de saut d'obstacles galope à une vitesse de 6 m/s. Un gain de précision inférieur à 0.1 m/s est trop faible pour être vraiment perçu et interprété par les professionnels du secteur équestre.

	Scénario 1	Scénario 2
Nombre de splines	15	50
Moyenne	0.69	0.67
SD	0.58	0.58

Table 2.2: Moyenne (en m/s) et SD d'erreur de prédiction en fonction du lissage des données

4.2 Modèles non paramétriques

Les méthodes non paramétriques sont utilisées quand on souhaite faire le moins d'hypothèses possibles sur la forme du lien entre la variable à prédire (ici un scalaire noté Y) et la variable prédictrice (ici une variable fonctionnelle notée X). "Non paramétrique" fait référence au fait que l'on utilise un paramètre de modélisation libre pour l'estimation des opérateurs non linéaires, ici les méthodes à noyaux seront utilisées.

Les méthodes à noyaux sont bien connues de la communauté non-paramétrique car elles sont utiles pour réaliser une pondération locale. Pour rappel dans le cas univarié, la pondération par noyau local est basée sur des fonctions à noyau et sur un paramètre

de lissage, appelé bande passante. L'idée principale de la pondération locale autour d'un nombre x est d'attribuer à chaque variable un poids qui prend en compte la distance entre x et cette variable, plus la variable est éloignée de x, plus le poids est faible. Il existe plusieurs fonctions à noyau, les plus connues sont les fonctions positives et symétriques comme les noyaux Box, Triangle, Quadratique et Gaussien. Dans le cas multivarié, cette approche est étendue, on considère alors un noyau multivarié. Dans ce cas, la pondération par noyau local multivarié consiste à transformer les n vecteurs aléatoires en n variables qui sont des transformations pondérées des variables prenant en compte la proximité avec x et la taille de la variable. L'extension de ce concept aux données fonctionnelles implique l'utilisation de la distribution de probabilité de la variable aléatoire fonctionnelle.

Trois modèles de prédiction sont proposés dans [Ferraty and Vieu, 2006b] : soit $(X_i, Y_i)_{i=1,\dots,n}$ n paires indépendantes, identiquement distribuées pouvant s'écrire (X, Y) et prenant leurs valeurs dans $E \times \mathbb{R}$, où (E, d) est un espace de semi-métrique et d une semi-métrique. Soit x (resp. y) un élément fixé de E (resp. $\mathbb{R}$)

- prédiction via l'espérance conditionnelle : $\hat{y} = \hat{r}(x)$ avec $\hat{r}(x)$ un estimateur de $r(x) = \mathbb{E}(Y|X = x)$,

- prédiction via la médiane fonctionnelle conditionnelle : $\hat{y} = \hat{m}(x)$, avec $\hat{m}(x)$ un estimateur de la médiane conditionnelle fonctionnelle,

- prédiction via le mode fonctionnel conditionnel : $\hat{y} = \hat{\theta}(x)$, avec $\hat{\theta}(x)$ un estimateur du mode conditionnel fonctionnel.

Chacune de ces trois méthodes est basée sur l'estimation d'un opérateur non linéaire : un opérateur de régression, une fonction de densité conditionnelle ou une fonction de densité pour le dernier modèle. La première étape de la modélisation statistique consiste à introduire un ensemble de contraintes qui agissent sur l'un des opérateurs précédents. Ces objets mathématiques sont estimés à l'aide d'estimateurs à noyaux qui combinent les avantages d'avoir une expression simple et d'être faciles à implémenter.

Exemple Nous allons comparer la performance de prédiction du modèle de régression fonctionnel univarié par rapport à un modèle fonctionnel bivarié. Pour cela nous allons appliquer le modèle de prédiction via l'espérance conditionnelle pour prédire la vitesse par foulée à partir de l'accélération Az. Puis nous verrons l'impact de l'ajout d'une variable supplémentaire, la vitesse angulaire Gy sur la prédiction de vitesse. Les résultats obtenus sont présentés dans le Tableau 2.3. On observe que la prise en compte de la vitesse angulaire Gy permet de réduire l'erreur moyenne de prédiction et donc d'affiner les prédictions du modèle. On observe de plus que le passage en non paramétrique a permis de gagner 0.28 m/s sur la précision de la prédiction et de réduire l'écart type associé à l'erreur moyenne par rapport à la régression paramétrique (cf. Table 2.2).

	Régression Univariée	Régression Bivariée
Moyenne	0.41	0.38
SD	0.37	0.34

Table 2.3: Moyenne (en m/s) et SD d'erreur de prédiction en fonction du type de régression

5 Régression logistique fonctionnelle

La régression logistique permet de prédire une variable catégorielle à partir de prédicteurs numériques. Pour les variables catégorielles il est inaproprié d'utiliser une régression linéaire car les termes d'erreur ne sont pas distribués normalement. Dans le cas d'une régression linéaire, le lien entre la variable réponse et la variable explicative est une combinaison linéaire. Dans le cas de la régression logistique, ce lien est une probabilité exprimée par la transformée logit correspondant au logarithme du résultat par rapport à la variable explicative.

Ce modèle pourra servir à la prédiction de la qualité du saut car aucun modèle de régression ordinal n'existe à notre connaissance pour les données fonctionnelles.

Soit Y la variable réponse binaire et X l'ensemble des covariables, on souhaite modéliser la probabilité conditionnelle $\pi(x) = P(Y = 1|X = x)$ comme une fonction de x qui décrit l'effet de X sur la distribution de la réponse Y.

Dans le cas fonctionnel logistique, le prédicteur $X : T \to \mathbb{R}$ est supposé être une fonction double intégrable dans un domaine compact $T \subset \mathbb{R}$ et la probabilité de succès conditionnelle est supposée être de la forme :

$$\pi(x) = \frac{e^{\alpha+\int_T \beta(t)x(t)dt}}{1 + e^{\alpha+\int_T \beta(t)x(t)dt}} \tag{2.3}$$

où α est le paramètre de moyenne et $\beta : T \to \mathbb{R}$ est une fonction dont les coefficients sont supposés double-intégrables sur T.

Avec la transformation logit le modèle peut s'écrire :

$$\eta(x) = logit(\pi(x)) = log(\frac{\pi(x)}{1 - \pi(x)}) = \alpha + \int_T \beta(t)x(t)dt \tag{2.4}$$

La fonction coefficient β décrit la relation entre Y et X. De grandes valeurs de $|\beta(t)|$ sur certaines parties du domaine indiquent que les différences de X pour ces régions ont une grande puissance prédictive pour Y.

Soient (x_i, y_i) les paires de données issues de n sujets. Ces paires sont considérées comme des réalisations des variables aléatoires indépendantes $(X_i, Y_i), i = 1, ..., n$ avec la distribution conditionnelle $Y_i|X_i = x$ donnée par les équations 2.3 et 2.4. La probabilité de succès conditionnelle pour le sujet i est $\pi_i = \pi(x_i)$ et le prédicteur linéaire correspondant est noté η_i.

Le problème majeur de la régression logistique fonctionnelle est l'estimation de β. Pour résoudre le problème de l'équation 2.3 il faut faire des restrictions sur x_i et β. L'approche standard est de représenter x_i et β dans une base de fonctions. Soit ϕ et ω les bases de fonction, on peut écrire :

$$x_i(t) = \sum_{r=1}^{R_x} c_{ir}\phi_r(t) = c_i^T\phi(t)$$

$$\beta(t) = \sum_{l=1}^{R_\beta} b_l\omega_l(t) = \omega^T(t)b$$

avec $\phi_r(t)$ et $\omega_l(t)$ les r -ièmes et l -ièmes bases de fonctions calculées au temps t, $\phi(t)$ et $\omega(t)$ les vecteurs contenant ces valeurs, c_{ir} et b_l les coefficients d'expression dans ces bases

et c_i et b les vecteurs contenant les coefficients concaténnés.

Tous les coefficients correspondant aux fonctions des covariables sont rassemblés dans la matrice C de dimension $n \times R_x$. On note $J_{\phi\omega}$ la matrice de dimension $R_x \times R_\beta$ telle que $J_{\phi\omega} = \int_T \phi(t)\omega^T(t)dt$. Soit $\eta = (\eta_1, ..., \eta_n)^T$ le vecteur de prédicteurs linéaires et $\mathbf{1} = (1, ..., 1)^T$ un vecteur de dimension n. En substituant l'équation 2.4 par les expressions matricielles, on obtient :

$$\eta = \alpha\mathbf{1} + \int_T C\phi(t)\omega^T(t)bdt = \alpha\mathbf{1} + CJ_{\phi\omega}b$$

D'après l'hypothèse d'indépendance, la fonction de vraisemblance conditionnelle est donnée par :

$$\begin{aligned} L(\alpha, \beta) &= \prod_{i=1}^{n} \pi^{y_i}(1-\pi_i)^{1-y_i} \\ &= \prod_{i=1}^{n} \frac{e^{y_i(\alpha+\int_T x_i(t)\beta(t)dt)}}{1+e^{\alpha+\int_T x_i(t)\beta(t)dt}} \\ &= \prod_{i=1}^{n} \frac{e^{y_i(\alpha+c_i^T J_{\phi\omega}b)}}{1+e^{\alpha+c_i^T J_{\phi\omega}b}} \end{aligned}$$

et l'estimateur du maximum de vraisemblance peut être trouvé par des méthodes d'optimisation numérique.

Or, on sait que les estimations des paramètres dans les modèles de régression logistique multiple ne sont pas fiables quand il y a un haut degré de dépendance entre les covariables. La nature fonctionnelle des données implique une corrélation importante entre les colonnes de la matrice $CJ_{\phi\omega}$. Plusieurs auteurs ont donc proposé des méthodes pour contourner ce problème. Trois de ces méthodes, la régression logistique fonctionnelle utilisant les bases à ondelettes et la pénalisation Lasso, la régression logistique par composante principale fonctionnelle et la régression logistique fonctionnelle pénalisée, sont comparées dans [Mousavi and Sørensen, 2018] sur données simulées et sur des données réelles de détection de boiteries de chevaux à partir de signaux d'accélération. Mais aucune de ces méthodes n'est disponible sous la forme d'un package R.

Part I

Prédiction de la vitesse et de la longueur de foulée

Les deux premiers objectifs de la thèse ont pour but la prédiction d'une variable quantitative Y, correspondant respectivement à la vitesse et à la longueur de foulée, par les signaux collectés par l'IMU. Comme vu en introduction, ces signaux peuvent être étudiés d'un point de vue statistique multidimensionnelle et d'un point de vue fonctionnel. Dans un premier temps, nous allons faire un rapide état des lieux des méthodes statistiques multidimensionnelles permettant de prédire une variable quantitative et leur application sur les données. Puis, nous aborderons la méthode de clustering fonctionnelle multivariée que nous avons développé dans le but d'améliorer la précision des prédictions de vitesse et de longueur de foulée et son application. Pour finir, nous présenterons l'extension de ce modèle au co-clustering de données fonctionnelles multivariées, méthode qui devrait permettre une fouille des bases de données de l'entreprise.

Chapitre 3

Etat des lieux des méthodes de régression d'une variable quantitative

1 Régression linéaire multiple

La régression linéaire multiple consiste à prédire une variable quantitative Y par p variables explicatives quantitatives $X_1, ..., X_p$. Ces variables sont liées par le modèle suivant :

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip} + \epsilon_i, \ i = 1, ..., n \tag{3.1}$$

avec y_i les mesures de la variable réponse Y, x_{ij} les valeurs des variables explicatives X_{ij}, β_j les paramètres du modèle à estimer, β_0 la constante du modèle et ϵ_i les erreurs de mesure [Cornillon et al., 2008].

A partir des observations, nous estimons les paramètres inconnus du modèle en minimisant le critère des moindres carrés :

$$\hat{\beta} = \text{argmin}_{\beta_0,...,\beta_p} \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2,$$

ou de façon équivalente par maximisation de la vraisemblance sous l'hypothèse de normalité des résidus. Sous ce modèle $Y \sim \mathcal{N}(X\beta, \sigma^2)$.

Le modèle de régression linéaire multiple est le modèle le plus simple (après la régression linéaire simple) de prédiction d'une variable quantitative.

2 Generalized Additive Model (GAM)

Cette catégorie de modèles permet d'utiliser des fonctions non linéaires de chacune des variables $X_1, ..., X_p$ en conservant l'additivité. Ce modèle s'écrit alors :

$$y_i = \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + ... + f_j(x_{ij}) + \epsilon_i, i = 1, ..., n, 1 \leq j \leq p,$$

avec f_j une fonction non linéaire. Les fonctions que l'on peut utiliser sont, par exemple, les splines, des polynômes ou encore des fonctions Loess [James et al., 2014]. Une fonction Loess est une méthode de régression non paramétrique par ajustement local permettant

de produire des courbes lissées. Ce lissage est plus ou moins important selon la taille de la fenêtre pour l'ajustement choisie.

L'avantage majeur de cette méthode est que l'on peut automatiquement modéliser des relations non linéaires qu'une régression linéaire aurait manqué. Mais le modèle GAM est restreint à l'additivité, certaines interactions peuvent alors être ignorées. Malgré tout, cette catégorie de modèles est un bon compromis entre la régression linéaire et les approches intégralement non paramétriques.

3 Modèles de mélange de régressions

Dans le cadre de ce travail, nous avons choisi de présenter ici le modèle *FlexMix* [Leisch, 2004] dont le package est disponible sous R.

On définit un modèle de mélange de régression à p composantes par :

$$h(y|x,\psi) = \sum_{j=1}^{p} \pi_j f(y|x,\theta_j), \tag{3.2}$$

$$\pi_j \geq 0, \sum_{j=1}^{p} \pi_j = 1,$$

où y est la variable à prédire de densité conditionnelle h, x est un vecteur de variables explicatives indépendantes, π_j est la probabilité a priori de la composante j, θ_j est le vecteur de paramètres spécifiques à la composante pour la fonction de densité f et $\psi = (\pi_1, ..., \pi_p, \theta'_1, ..., \theta'_p)$ est le vecteur contenant tous les paramètres.

Par exemple, si f est une fonction de densité normale univariée de moyenne $\beta'_j x$ et de variance σ_j^2, spécifiques aux composantes, on peut alors noter $\theta_j = (\beta'_j, \sigma_j^2)'$ et l'équation 3.2 décrit alors un mélange de modèles de régressions linéaires.

La probabilité postérieure que l'observation (x, y) appartienne à la classe k est donnée par :

$$P(j|x,y,\psi) = \frac{\pi_k f(y|x,\theta_k)}{\sum_j \pi_j f(y|x,\theta_j)}.$$

Les probabilités postérieures peuvent être utilisées pour segmenter les données en assignant chaque observation à la classe pour laquelle cette probabilité est maximale.

L'estimation des paramètres du modèle se fait pas estimation du maximum de vraisemblance à l'aide de l'algorithme Expectation Maximisation [Dempster et al., 1977].

L'avantage de ce type de modèle est qu'il peut détecter, au sein d'une base de données, des sous-populations suivant des lois différentes.

4 Méthodes d'analyse de données en grande dimension

Ces méthodes consistent à prédire une variable quantitative Y par p variables quantitatives quand le nombre p de variables est très grand voire plus grand que le nombre n d'individus à analyser. En effet, dans ce dernier cas de figure, il est impossible d'utiliser la méthode

des moindres carrés pour estimer les paramètres du modèle. Les présentations de ces méthodes sont issues de [James et al., 2014, Hastie et al., 2001].

4.0.1 Régression Ridge

Cette régression consiste à estimer le modèle de régression 3.1 en minimisant la quantité suivante :

$$\text{RSS } + \lambda \sum_{j=1}^{p} \beta_j^2, \lambda \geq 0,$$

avec $RSS = \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2$, p le nombre de variables, et $\lambda \sum_{j=1}^{p} \beta_j^2$ la pénalité de réduction des coefficients de régression.

Cette méthode réduit les coefficients à des valeurs proches de 0 pour contrôler la variance. La valeur des coefficients de régression est très dépendante de la réduction ou non des données d'entrée. Il est donc recommandé pour des variables d'échelle différente de les réduire afin qu'elles soient toutes exprimées sur la même échelle. L'avantage de la régression Ridge par rapport à la méthode des moindres carrés est le compromis biais-variance : quand λ augmente, la souplesse de la courbe de régression diminue, il y a une diminution de la variance mais une augmentation du biais. Le désavantage de cette méthode est qu'elle va toujours conserver toutes les variables dans le modèle, ce qui peut en rendre l'interprétation difficile.

4.1 Régression Lasso

La régression Lasso surmonte le désavantage de la méthode Ridge, car elle réduit les coefficients à des valeurs pouvant être égales à 0 quand λ est suffisamment grand. Cette méthode permet donc une sélection de variables. Les coefficients Lasso minimisent la quantité :

$$\text{RSS} + \lambda \sum_{j=1}^{p} |\beta_j|.$$

La sélection de lambda se fait par validation croisée.

4.1.1 Elastic net

Cette méthode propose un compromis différent situé entre Ridge et Lasso. Les coefficients de l'équation 3.1 sont estimés en minimisant :

$$\lambda \sum_{j=1}^{p} (\alpha|\beta_j| + (1-\alpha)\beta_j^2).$$

Le second terme encourage les traits caractéristiques très corrélés à être moyennés et le premier terme de l'équation encourage une solution avec peu de coefficients pour ces traits moyennés.

4.2 Régression sur composantes principales (PCR)

Cette méthode fait partie des méthodes de régression par réduction de la dimension qui visent à transformer les variables explicatives puis à estimer le modèle à l'aide des composantes obtenues par un modèle des moindres carrés.

Soit $Z_1, ..., Z_M$ la représentation des $M < p$ combinaisons linéaires de nos p variables explicatives, appelées composantes principales et pouvant s'écrire :

$$Z_m = \sum_{j=1}^{p} \phi_{jm} X_j, \ m = 1, ..., M$$

avec $X_1, ..., X_p$ les prédicteurs originaux et $\phi_{1m}, ..., \phi_{pm}$ les paramètres de la combinaison linéaire. On peut alors écrire le modèle de régression linéaire suivant :

$$y_i = \theta_0 + \sum_{m=1}^{M} \theta_m z_{im} + \epsilon_i, \ i = 1, ..., n.$$

que l'on va chercher à estimer par la méthode des moindres carrés.

Ainsi, la première composante principale est la direction selon laquelle les observations varient le plus. Elle est choisie de façon à ce que les observations projetées soient les plus proches des observations originales. La deuxième composante principale est une combinaison linéaire de variables qui n'est pas corrélée à la première composante et dont la variance est la plus grande par rapport à cette contrainte. La PCR implique donc de construire les M premières composantes principales et d'utiliser ces composantes comme des prédicteurs dans un modèle de régression linéaire qui est modélisé à l'aide de la méthode des moindres carrés.

Il est recommandé de réduire les données avant d'appliquer la PCR car en l'absence de cette réduction, les variables avec une grande variance auront plus de poids dans la construction des composantes principales que les autres.

Le désavantage de cette méthode est que les combinaisons linéaires des prédicteurs sont choisies de façon non supervisée puisque la variable réponse Y n'intervient pas dans la détermination des composantes principales.

4.3 Partial Least Square (PLS)

Cette méthode est une alternative supervisée à la méthode PCR, qui fait donc partie des méthodes de réduction de la dimension. En effet, elle utilise la réponse Y pour identifier les nouvelles composantes qui sont des combinaisons linéaires des variables de départ et qui doivent être liées à la réponse Y.

La première direction de la PLS est calculée de la façon suivante : il y a d'abord une première étape de réduction des p prédicteurs. Puis, on calcule la première direction Z_1 en imposant chaque paramètre de la combinaison linéaire à être égal au coefficient issu de la régression linéaire simple de Y sur X. Ces coefficients sont proportionnels à la corrélation entre X et Y. Pour identifier la 2ème direction de la PLS, on ajuste chacune des variables sur Z_1 à l'aide d'une régression et on conserve les résidus. On calcule Z_2 en faisant une régression des résidus de chaque variable sur Y et on conserve les coefficients issus de cette régression. On applique ensuite cette procédure itérative pour identifier les M directions suivantes. Pour finir, on utilise la méthode des moindres carrés pour prédire y à partir des M composantes.

En résumé, la régression PLS recherche de façon itérative une suite de composantes orthogonales entre elles. Ces composantes sont choisies de façon à maximiser la covariance avec la variable réponse Y. Le choix du nombre de composantes se fait par validation croisée ou à l'aide d'un échantillon de test et un échantillon de validation afin d'éviter le surajustement.

5 Méthodes de Machine Learning

Dans cette section nous allons présenter rapidement les méthodes les plus connues de Machine Learning pour la régression en s'appuyant sur les ouvrages [Bishop, 2006, Hastie et al., 2001].

5.1 Support Vector Machine (SVM)

Le but des SVM dans le cas de la classification est de trouver l'hyper plan de séparation optimal qui maximise la marge des données d'entrainement. Cette marge correspond à l'intervalle autour de l'hyper-plan dans lequel aucun point n'est présent, sa taille correspond à la distance entre l'hyper plan et le point le plus proche.

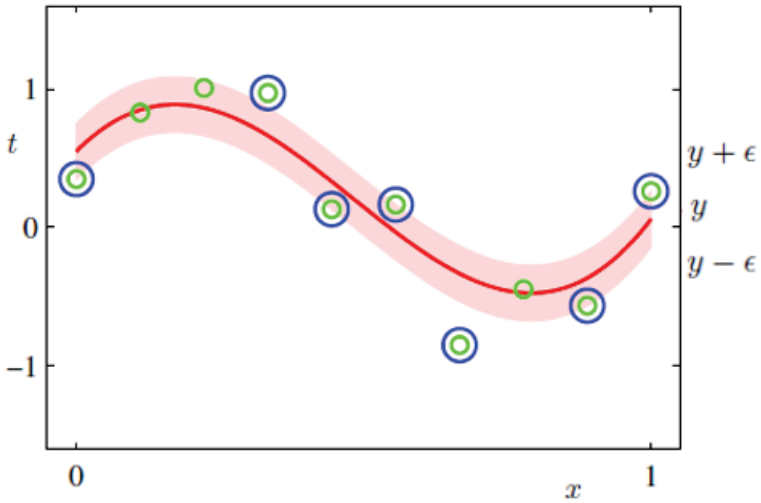


Figure 3.1: Graphe permettant d'illustrer la régression SVM ϵ et la régression SVM ν [Bishop, 2006]

Dans le cas des SVM pour la régression, deux cas sont possibles. Le premier est le modèle ϵ-insensible dont le but est de trouver la fonction $f(x_i)$ qui a au plus une déviation ϵ par rapport à la cible y_i sur les données d'entrainement. Cette régression consiste donc à fixer la largeur ϵ de la région insensible utilisée pour coller le modèle aux données d'entrainement. Cette valeur induit le nombre de vecteurs supports utilisés pour construire la fonction de régression : plus ϵ est grand, moins on a de vecteurs supports.

La seconde alternative est la régression ν où le paramètre ν limite la fraction de points situés dans la zone insensible.

Ces deux régressions sont illustrées à la Figure 3.1. $f(x_i)$ est représentée par le trait rouge, les points verts représentent les données x_i, la bande rouge claire représente la zone ϵ insensible et les points entourés de bleu les vecteurs supports. Dans le cas de la régression-ν, c'est le nombre de points situés dans la zone rouge claire qui est limité.

5.2 Forêt aléatoire

La méthode des forêts aléatoires est une méthode non paramétrique qui peut gérer la non linéarité, les interactions, les prédicteurs corrélés et l'hétérogénéité des données. Une forêt aléatoire combine plusieurs arbres de décision individuels pour fournir une prédiction finale qui correspond à la moyenne des prédictions fournies par les arbres. Chaque arbre est construit à partir d'un échantillon aléatoire de données, des variables de segmentation sont choisies aléatoirement et l'arbre est découpé en plusieurs noeuds en suivant la meilleure segmentation basée sur un critère d'impureté du noeud. La précision de chaque arbre est estimée sur l'échantillon de données restant, n'ayant pas servi à sa construction. Le nombre d'arbres utilisés dans le modèle est un paramètre à choisir en fonction du problème traité.

5.3 Réseau de neurones

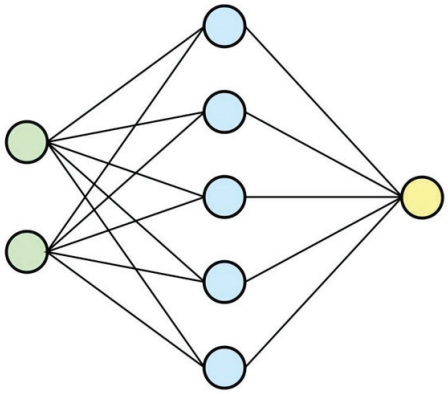


Figure 3.2: Schéma d'un réseau de neurone avec deux variables d'entrée et une couche composée de 5 unités cachées

L'idée principale des réseaux de neurones dans le cas de la régression est d'extraire des combinaisons linéaires des variables d'entrée (points verts sur la Figure 3.2), appelées caractéristiques dérivées ou unités cachées (points bleus) et ensuite de modéliser la variable cible (point jaune) comme une fonction non linéaire de ces unités cachées. La fonction linéaire utilisée peut être un modèle linéaire standard ou un modèle multilogit. Il est nécessaire d'estimer les poids de chaque variable d'entrée permettant de se rapprocher au mieux de la variable cible, ce qui nécessite un jeu de données d'entrainement. Pour entrainer un réseau de neurones, les valeurs des poids sont choisies au départ de façon aléatoire proches de 0, puis, par itérations successives, les poids évoluent, le modèle commence donc presque linéaire et devient non linéaire au fur et à mesure que les poids augmentent. Pour la construction du modèle, il faut aussi renseigner le nombre d'unités cachées et le nombre de couches du réseau. Le nombre d'unités cachées se situe communément entre 5 et 100 et dépend du nombre de variables d'entrées et du nombre de cas d'entrainement. Le nombre optimal peut être estimé par validation croisée. Le nombre de couches cachées est guidé par la connaissance a priori.

6 Résultats obtenus pour la prédiction de la vitesse

Dans un premier temps, nous avons travaillé sur une base de données résumée des signaux collectés par la selle, composée de 22 variables calculées comme par exemple la moyenne d'accélération selon l'axe X, la fréquence de galop, la hauteur de certains pics caractéristiques des signaux, les énergies du mouvement, etc. La majorité de ces variables correspondaient à des éléments supposés comme étant des caractéristiques biomécaniques d'une foulée que l'on pensait différenciantes [Lopez et al., 2011, Moe-Nilssen and Helbostad, 2004, Gastin et al., 2008]. Sur ces données, l'efficacité de trois méthodes ont été comparées : la régression multiple, GAM et Flexmix.

Les différentes méthodes sont comparées par rapport au pourcentage d'erreur de prédiction supérieur à 0.6 m/s. Ce pourcentage est calculé pour chaque simulation où le modèle est construit sur la base de données d'entrainement, composée d'un tirage aléatoire de 80% des données, et testé sur la base de données de test, composée des 20% de données restantes. Les valeurs médiane, minimale et maximale de pourcentage d'erreurs pour chaque méthode calculées sur 500 simulations sont résumées en Table 3.1.

Méthode	Base de données	Médiane	Min	Max
Régression multiple	Base de données intégrale	26.8	35.8	46.8
GAM	Base de données intégrale	38.9	29.5	49.5
Flexmix	Groupe 1	16.7	0	41.3
Flexmix	Groupe 2	22.2	0	40.4
Flexmix	Groupe 3	4.1	0	41.1
Flexmix	Groupe 4	0	0	19.6

Table 3.1: Médiane, valeur minimale et maximale de pourcentage d'erreur supérieur à 0.6 m/s sur la base de 1906 foulées

On observe que lorsque la base de données est découpée en sous-groupes plus homogènes du point de vue des variables mesurées, le pourcentage d'erreur médian supérieur à 0.6 m/s diminue. Or, l'objectif fixé par l'entreprise est d'arriver à passer sous la barre des 10% d'erreur. Seuls deux groupes obtenus à l'aide de la régression Flexmix atteignent ce seuil. Nous avons donc choisi d'étudier la base de données brutes, comme présentée dans l'introduction, en espérant gagner en précision, plutôt qu'un résumé des signaux collectés par la centrale inertielle.

Nous avons donc testé dans un second temps les méthodes d'analyse de données en grande dimension et les méthodes d'analyse de données fonctionnelles sur les données brutes collectées par le capteur de la selle comme décrites en introduction. On observe à la Table 3.2 que les méthodes d'analyse de données fonctionnelles présentent de meilleurs résultats que les méthodes d'analyse de données en grande dimension d'un point de vue du pourcentage d'erreur moyen.

Méthode	Base de données	Moyenne	Min	Max
Ridge	Base de données brutes	27.7	24.2	30.7
Ridge	Coefficients splines	36.0	33.0	40.8
Lasso	Base de données brutes	29.1	25.9	31.8
Lasso	Coefficients splines	36.1	31.8	41.1
PCR	Base de données brutes	28.5	25.5	31.1
PCR	Coefficients splines	35.2	31.1	39.9
PLS	Base de données brutes	28.8	25.0	32.2
PLS	Coefficients splines	35.2	31.1	39.9
Elastic net $\alpha = 0.3$	Base de données brutes	27.9	24.6	30.9
Elastic net $\alpha = 0.3$	Coefficients splines	35.7	32.4	40.2
Régression multiple	Base de données brutes	34.0	30.4	37.8
Régression multiple	Coefficients splines	35.2	31.1	39.9
Régression fonctionnelle non paramétrique	Base de données brutes	16.7	13.0	20.1
Régression fonctionnelle non paramétrique	Coefficients splines	26.2	21.0	29.8
Régression fonctionnelle paramétrique	Base de données brutes	26.9	23.6	31.7
Régression fonctionnelle paramétrique	Coefficients splines	25.0	21.0	28.9

Table 3.2: Moyenne, valeur minimale et maximale de pourcentage d'erreur supérieur à 0.6 m/s sur la base de 1906 foulées

On a pu voir avec la méthode Flexmix que le découpage de la base de données en sous-groupes plus homogènes avant l'application d'un modèle de régression permettait de diminuer le pourcentage d'erreur de prédiction. Nous avons donc souhaité tester l'apport de la séparation de la base de données en plusieurs sous-ensembles, à l'aide d'une méthode de clustering fonctionnelle univariée, avant l'application du meilleur modèle de prédiction. Dans ce cas, c'est la moyenne pondérée des résultats des différents groupes qui est présentée à la Table 3.3.

Méthode	Base de données	Moyenne	Min	Max
Régression fonctionnelle non paramétrique	Base de données brutes	12.9	9.5	17.3
Funclust 2 classes + Régression fonctionnelle non paramétrique	Base de données brutes	14.4	8.6	19.6
Funclust 4 classes + Régression fonctionnelle non paramétrique	Base de données brutes	15.1	7.3	22.8
FunFEM 6 classes + Régression fonctionnelle non paramétrique	Base de données brutes	13.9	5.4	25.6
funHDDC 2 classes + Régression fonctionnelle non paramétrique	Base de données brutes	13.4	8.2	18.6
funHDDC 3 classes + Régression fonctionnelle non paramétrique	Base de données brutes	14.5	8.0	23.5

Table 3.3: Moyenne pondérée, valeur minimale et maximale de pourcentage d'erreur supérieur à 0.6 m/s sur la base de 2685 foulées

On peut remarquer que l'application des modèles de prédiction sur des sous-groupes issus du clustering fonctionnel permet de réduire le minimum de pourcentage d'erreur de prédiction. Or, le modèle de clustering appliqué est un modèle univarié, les meilleurs résultats de prédiction sont obtenus avec un découpage de la base selon l'accélération Az. Afin de pouvoir réaliser un découpage plus fin, à partir de plusieurs variables fonctionnelles et non pas qu'une seule variable, nous avons décidé de développer un modèle de clustering fonctionnel multivarié s'appuyant sur le même principe que funHDDC. Ce modèle est présenté dans la partie suivante.

Chapitre 4

Clustering de données fonctionnelles multivariées

Nous avons pu voir précédement que la division de la base de donnée générale en sous-ensembles plus homogènes avant l'application d'un modèle de prédiction semble impacter le pourcentage d'erreur de prédiction. Nous allons présenter ici la méthode de clustering fonctionnelle multivariée que nous avons développé de façon à tenir compte de plusieurs signaux collectés dans la division de notre base en sous-ensembles homogènes. Le travail présenté dans cette partie est un article actuellement en cours de révision.

1 Article

Computational Statistics manuscript No.
(will be inserted by the editor)

Clustering multivariate functional data in group-specific functional subspaces

Amandine Schmutz · Julien Jacques · Charles Bouveyron · Laurence Chèze · Pauline Martin

Received: date / Accepted: date

Abstract With the emergence of numerical sensors in many aspects of everyday life, there is an increasing need in analyzing multivariate functional data. This work focuses on the clustering of such functional data, in order to ease their modeling and understanding. To this end, a novel clustering technique for multivariate functional data is presented. This method is based on a functional latent mixture model which fits the data into group-specific functional subspaces through a multivariate functional principal component analysis. A family of parsimonious models is obtained by constraining model parameters within and between groups. An Expectation Maximization algorithm is proposed for model inference and the choice of hyper-parameters is addressed through model selection. Numerical experiments on simulated datasets highlight the good performance of the proposed methodology compared to existing works. This algorithm is then applied to the analysis of the pollution in French cities for one year.

Keywords Multivariate functional curves · multivariate functional principal component analysis · model-based clustering · EM algorithm

We would like to thank the LabCom 'CWD-VetLab' for its financial support. The LabCom 'CWD-VetLab' is financially supported by the Agence Nationale de la Recherche (contract ANR 16-LCV2-0002-01)

Amandine Schmutz · Pauline Martin
Lim France, Chemin Fontaine de Fanny, Nontron, France & CWD-VetLab, Ecole Nationale Vétérinaire d'Alfort, Maisons-Alfort, F-94700, France. Tel.: +336-85861287,
Orcid: 0000-0003-2523-0411 / 0000-0002-3571-4244, E-mail: aschmutz@lim-group.com

Julien Jacques
Université de Lyon, Lyon 2, ERIC EA3083, Lyon, France. Orcid: 0000-0003-4808-2781

Charles Bouveyron
Université Côte d'Azur, Inria, CNRS, LJAD, Maasai team, France. Orcid: 0000-0002-6956-4491

Laurence Chèze
Université de Lyon, Lyon 1, LBMC UMR T9406, Lyon, France. Orcid: 0000-0003-2265-9781

1 Introduction

The modern technologies ease the collection of high frequency data which is of interest to model and understand the studied phenomenon for further analyses. For example in sports, athletes wear devices that collect data during their training to improve their performances and follow their physical constants in order to prevent injuries. This kind of data can be classified as functional data: a quantitative entity evolving along time. For instance in the univariate case, a functional data X is represented by a single curve, $X(t) \in \mathbb{R}$, $\forall t \in [0, T]$. With the growth of smart device market, more and more data are collected for the same individual, such as runner heartbeat and the altitude of his travel. An individual is then represented by several curves. The corresponding multivariate functional data can be written: $\mathbf{X} = \mathbf{X}(t)_{t\in[0,T]}$ with $\mathbf{X}(t) = (X^1(t), ..., X^p(t))' \in \mathbb{R}^p, p \geq 2$. We refer to Ramsay and Silverman (2005) for univariate and bivariate examples.
Because of this amount of collected data, there is an increasing need for methods able to identify homogeneous subgroups of data, to make better individualized predictions for example. The clustering of functional data can be addressed with different methods, that can be split into 4 categories according to Jacques and Preda (2014a): the raw data methods that consists of clustering directly the curves on their finite set of points ; the filtering methods that need a first step of smoothing curves into a basis of functions and a second step of clustering the obtained expansion coefficients ; the adaptive methods where clustering and expression of the curves into a finite dimensional space are performed simultaneously ; and the distance-based methods where usual clustering algorithms are applied with specific distances for functional data. Among these categories, there exist numerous works for the clustering of univariate functional data as for instance James and Sugar (2003), Tarpey and Kinateder (2003), Chiou and Li (2007), Bouveyron and Jacques (2011), Jacques and Preda (2013), Bouveyron et al. (2015) and Bongiorno and Goia (2016).

Conversely, only a few exists for clustering multivariate functional data. Singhal and Seborg (2005) and Ieva et al. (2013) use a k-means algorithm based on specific distances between multivariate functional data. Kayano et al. (2010) consider Self-Organizing Maps based on the coefficients of multivariate curves into an orthonormalized Gaussian basis expansions. Tokushige et al. (2007) extend crisp and fuzzy k-means algorithms for multivariate functional data by considering a specific distance between functions. Those methods cluster data by considering that they lie in the same subspace. A new method has been recently published based on a hypothesis testing k-means (Dias et al., 2018). At each step of the k-means algorithm, the curve belonging decision is based on the combination of two hypothesis test statistics. The performance of their algorithm is compared to distance-based methods and some dimension reduction based methods. Those dimension reduction techniques main principle is to obtain a low-dimensional representation of functions. For example, Ieva and Paganoni (2013) present a generalized functional linear regression

model that cluster individuals in two categories. The first step consist of a multivariate functional principal component analysis applied on the variance-covariance matrix on the functional data and their first derivatives. Then, the obtained scores are used as covariates in a generalized linear model to predict the outcome.Yamamoto and Hwang (2017) propose a clustering method that combines a subspace separation technique with functional subspace clustering, named FGRC, that is less sensible to data variance than functional principal component k-means developed by Yamamoto (2012) and functional factorial k-means (Yamamoto and Terada (2014)). Finally, Jacques and Preda (2014b) present a Gaussian model-based clustering method based on a principal component analysis for multivariate functional data (MFPCA). One of the benefits of this method is that the dependency between functional variables is managed thanks to the MFPCA. More recently, new methods based on a mix between dimension reduction and nonparametric approaches appear. Indeed, Traore et al. (2019) propose a clustering technique for nuclear safety experiment where one individual curve is decomposed into two new curves that are used in the decision making process. The first step consists in doing a dimension reduction technique on the first curves and applying a hierarchical clustering on those obtained values. Then, a semi-metric is build to compare the second curves, and the clusters are refining thanks to this comparison. But, even if this method is developped to deal with two curves for a same individual, at first the functional data are univariate.

In Jacques and Preda (2014b), MFPCA scores are considered as random variables whose probability distributions are cluster specific. Although this model is far more flexible than other methods due to its probabilistic modeling, it suffers nevertheless from some limitations. Indeed, using an approximation of the notion of density distribution for functional data, the authors modeled only a given proportion of principal components and thus a significant part of the available information is ignored. In this paper, we propose a model which extends Jacques and Preda (2014b) work by modeling all principal components whose estimated variance are non-null. All available information is therefore taken into account. This is a significant advantage because it will give a finner modeling and, consequently, a better clustering in most cases. Moreover, our model allows to use an Expectation Maximisation (EM) algorithm for its inference, with the theoretical guaranties it implies, whereas Jacques and Preda (2014b) use an heuristic pseudo-EM algorithm with no theoretical guaranties. The resulting model can be also viewed as an extension of Bouveyron and Jacques (2011) method to the multivariate case, that is why we will refer to it as the funHDDC model in the following.

The paper is organized as follows. A quick reminder of function data analysis is done in Section 2 . Section 3 presents principal component analysis for multivariate functional data, as introduced in Jacques and Preda (2014b). Section 4 introduces the mixture model allowing the clustering of multivariate functional data. Section 5 discusses parameters estimation via an EM algorithm, proposes criteria for the selection of number of clusters and computational details. Comparisons between the proposed method and existing ones

on simulated and real datasets are presented in Sections 6 and 7. A discussion concludes the paper in Section 8.

2 Functional data analysis

Let us first assume that the observed curves $\mathbf{X}_1, ..., \mathbf{X}_n$ are independent realizations of a L_2-continuous multivariate stochastic process $\mathbf{X} = \{\mathbf{X}(t), t \in [0,T]\}$ $= \{(X^1(t), \ldots, X^p(t))\}_{t\in[0,T]}$ for which the sample paths, i.e. the observed curves $\mathbf{X}_i = (X_i^1, ..., X_i^p)$, belong to $L_2[0,T]$. Without loss of generality, let assume that $E(\boldsymbol{X}) = 0$.

In practice, the functional expressions of the observed curves are not known and it is only possible to have access to discrete observations at a finite set of times $X_i^j(t_1), \ldots, X_i^j(t_s)$ with $0 \leq t_1 \leq \ldots, \leq t_s \leq 1$ for every $1 \leq i \leq n$ and $1 \leq j \leq p$. The first task, when working with functional data, is therefore to convert these discretely observed values to a function $X_i^j(t)$, computable for any desired argument value $t \in [0,T]$. One way to do that is interpolation, which is used if the observed values are assumed to be errorless. However, if there is some noise that needs to be removed, a common way to reconstruct the functional form is to assume that the curves $X_i^j(t)$ can be decomposed into a finite dimensional space, spanned by a basis of functions (Ramsay and Silverman, 2005):

$$X_i^j(t) = \sum_{r=1}^{R_j} c_{ir}^j (X_i^j) \phi_r^j(t) \tag{1}$$

where $(\phi_r^j(t))_{1\leq r\leq R_j}$ is the basis of functions for the j-th component of the multivariate curve and R_j the number of basis functions. In order to ease the description of the model, let us introduce the following notations. The coefficients c_{ir}^j can be gathered in the matrix $\boldsymbol{C}$:

$$\mathbf{C} = \begin{pmatrix} c_{11}^1 \ ... \ c_{1R_1}^1 \ c_{11}^2 \ ... \ c_{1R_2}^2 \ ... \ c_{11}^p \ ... \ c_{1R_p}^p \\ \\ ... \\ \\ c_{n1}^1 \ ... \ c_{nR_1}^1 \ c_{n1}^2 \ ... \ c_{nR_2}^2 \ ... \ c_{n1}^p \ ... \ c_{nR_p}^p \end{pmatrix}.$$

Let also introduce the matrix $\boldsymbol{\phi(t)}$, gathering the basis functions:

$$\boldsymbol{\phi(t)} = \begin{pmatrix} \phi_1^1(t) \ ... \ \phi_{R_1}^1(t) & 0 & ... & 0 & ... & 0 & ... & 0 \\ 0 \ ... \ 0 & \phi_1^2(t) \ ... \ \phi_{R_2}^2(t) & ... & 0 & ... & 0 \\ & & & ... & & & & \\ 0 \ ... \ 0 & 0 & ... & 0 & ... \ \phi_1^p(t) \ ... \ \phi_{R_p}^p(t) \end{pmatrix}.$$

With these notations, Equation (1) can be rewritten as follows:

$$\boldsymbol{X}(t) = \boldsymbol{C}\boldsymbol{\phi}'(t).$$

The estimation of $\boldsymbol{C}$ is usually done through least square smoothing (see Ramsay and Silverman (2005)). The choice of the basis functions, contained in $\boldsymbol{\phi}$, has to be made by the user. There is no straight rules about how to choose the

appropriate ones (Jacques and Preda (2014a)). We can nevertheless recommend the use of a Fourier basis in the case of data with a repetitive pattern, and B-spline functions in most other cases.

3 Multivariate functional principal component analysis

Principal component analysis for multivariate functional data has already been suggested by various authors. Ramsay and Silverman (2005) propose to concatenate observations of the functions measured on a fine grid of points into a single vector and then to perform a standard principal component analysis (PCA) on these concatenated vectors. They also propose to express observations into a known basis of functions and apply PCA on the vector of concatenated coefficients. Both approaches may be problematic when the functions correspond to different observed phenomena. Moreover, the interpretation of multivariate scores for one individual is usually difficult. In Berrendero et al. (2011), the authors propose instead to summarize the curves with functional principal components. For this purpose, they carry out classical PCA for each value of the domain on which the functions are observed and suggest an interpolation method to build their principal functional components. In a different approach, Jacques and Preda (2014b) suggest the MFPCA method, with a normalization step if the units of measurement differ between functional variables. Their method relies on the multidimensional version of the Karhunen-Loève expansion (Saporta, 1981). Chiou et al. (2014) also present a normalized multivariate functional principal component analysis which takes into account the differences in degrees of variability and units of measurement among the components of the multivariate random functions. As in Jacques and Preda (2014b), it leads to a single set of scores for each individual. Chen and Jiang (2016) present a multi-dimensional functional principal component analysis and Happ and Greven (2015) a multivariate functional principal component analysis that both can handle data observed on more than one-dimensional domain. Happ and Greven (2015) method can be applied to sparse functional data and includes the MFPCA proposed by Jacques and Preda (2014b) when the interval is $[0, T]$ and steady.

Because our data are collected on the one-dimensional interval [0,T] and with a regular sampling scheme, the MFPCA proposed by Jacques and Preda (2014b) is used in combination with a fine probabilistic modeling of the group-specific densities. The MFPCA method is therefore summarized hereafter. MFPCA aims at finding the eigenvalues and eigenfunctions that solve the spectral decomposition of the covariance operator ν:

$$\nu \boldsymbol{f}_l = \lambda_l \boldsymbol{f}_l, \forall l \geq 1, \tag{2}$$

with λ_l a set of positive eigenvalues and $\boldsymbol{f}_l$ the set of associated multivariate eigenfunctions. The estimator of the covariance operator can be written as:

$$\widehat{\nu}(s,t) = \frac{1}{n-1}\boldsymbol{X}'(s)\boldsymbol{X}(t) = \frac{1}{n-1}\boldsymbol{\phi}(s)\boldsymbol{C}'\boldsymbol{C}\boldsymbol{\phi}'(t). \tag{3}$$

Let suppose that each principal factor $\boldsymbol{f}_l$ belongs to the linear space spanned by the matrix $\boldsymbol{\phi}$:

$$\boldsymbol{f}_l(t) = \boldsymbol{\phi}(t)\boldsymbol{b}_l' \tag{4}$$

with $\boldsymbol{b}_l = (b_{l11}, ..., b_{l1R_1}, b_{l21}, ..., b_{l2R_2}, ..., b_{lp1}, ..., b_{lpR_p})$. Using Equation (3), the eigen problem (2) becomes:

$$\frac{1}{n-1}\boldsymbol{\phi}(s)\boldsymbol{C}'\boldsymbol{C}\boldsymbol{W}\boldsymbol{b}_l' = \lambda_l\boldsymbol{\phi}(s)\boldsymbol{b}_l' \tag{5}$$

where $\boldsymbol{W} = \int_0^T \boldsymbol{\phi}'(t)\boldsymbol{\phi}(t)$ is a $R \times R$ matrix, where $R = \sum_{j=1}^p R_j$, which contains the inner products between the basis functions. The MFPCA then reduces to eigendecomposition of the matrix $\frac{1}{\sqrt{n-1}}\boldsymbol{C}\boldsymbol{W}^{1/2}$. Thus, each multivariate curve $\boldsymbol{X}_i$ is identified by its score $\boldsymbol{\delta}_i = (\delta_{il})_{l\geq 1}$ into the basis of multivariate eigenfunctions $(\boldsymbol{f}_l)_{l\geq 1}$. Scores are obtained from $\delta_{il} = \boldsymbol{C}_i\boldsymbol{W}\boldsymbol{b'}_l$ where $\boldsymbol{C}_i$ is the i-th row of matrix $\boldsymbol{C}$.

In practice, due to the fact that each component X_i^j of $\boldsymbol{X}_i$ is approximated into a finite basis of functions of size R_j, the maximum number of scores which can be computed is $R = \sum_{j=1}^p R_j$.

4 A generative model for multivariate functional data clustering

Our goal is to group the observed multivariate curves $\boldsymbol{X}_1, ..., \boldsymbol{X}_n$ into K homogeneous clusters. At this stage, K is fixed a priori and an estimation procedure for this parameter will be suggested in Section 5.3. Let Z_{ik} be the latent variable such that $Z_{ik} = 1$ if $\boldsymbol{X}_i$ belongs to cluster k and 0 otherwise. In order to ease the presentation of the modeling, let us assume at first that the values z_{ik} of Z_{ik} are known for all $1 \leq i \leq n$ and $1 \leq k \leq K$ (our goal is in practice to recover them from the data). Let $n_k = \sum_{i=1}^n z_{ik}$ be the number of curves within cluster k.

Let suppose that the curves of each cluster can be described into a low-dimensional functional latent subspace specific to each cluster, with intrinsic dimensions $d_k < R$, $k = 1, ..., K$. Curves can be expressed into a group-specific basis $\{\varphi_r^k\}_{1\leq r\leq d_k}$, which is determined thanks to the model, and is obtained from $\{\phi_r^j\}_{1\leq j\leq p, 1\leq r\leq R}$ through a linear transformation:

$$\varphi_r^k(t) = \sum_{\ell=1}^R q_{kr\ell}\phi_\ell(t), 1 \leq r \leq R$$

where $Q_k = (q_{kr\ell})_{1\leq r,\ell\leq R}$ is the orthogonal $R \times R$ matrix containing the basis expansion coefficients of the eigenfunctions. Q_k is split for later use into two parts: $Q_k = [U_k, V_k]$ with U_k of size $R \times d_k$ and V_k of size $R \times (R - d_k)$, $U_k'U_k = I_{d_k}$, $V_k'V_k = I_{R-d_k}$ and $U_k'V_k = 0$.

Let $(\boldsymbol{\delta}_i^k)_{1\leq i\leq n_k}$ be the MFPCA scores of the n_k curves of cluster k. These scores are assumed to follow a Gaussian distribution

$$\boldsymbol{\delta}_i^k \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Delta}_k)$$

with $\boldsymbol{\mu}_k \in \mathbb{R}^R$ the mean function and $\boldsymbol{\Delta}_k$ the covariance matrix with the following form:

$$\boldsymbol{\Delta}_k = \begin{pmatrix} \boxed{\begin{matrix} a_{k1} & & 0 \\ & \ddots & \\ 0 & & a_{kd_k} \end{matrix}} & \mathbf{0} \\ \mathbf{0} & \boxed{\begin{matrix} b_k & & 0 \\ & \ddots & \\ & & \ddots \\ 0 & & b_k \end{matrix}} \end{pmatrix} \begin{matrix} \left.\vphantom{\begin{matrix} a \\ a \\ a \end{matrix}}\right\} d_k \\ \left.\vphantom{\begin{matrix} a \\ a \\ a \\ a \end{matrix}}\right\} R - d_k \end{matrix}$$

The assumption on $\boldsymbol{\Delta}_k$ allows to finely model the variance of the first d_k principal components, while the remaining ones are considered as noise components and modeled by a unique parameter b_k. This model will be referred to as $[a_{kj} b_k Q_k d_k]$ hereafter. The model of Jacques and Preda (2014b) is similar but with the constraint $b_k = 0$, for all $k = 1, ..., K$. The latter leads to ignore information contained in the last eigenfunctions, whereas we propose to model it in a parsimonious way.

In addition, different sub-models can be defined depending on the constraints we apply on model parameters, within or between groups, leading to more parsimonious models. This possibility allows to fit into various situations. The following 5 sub-models can be derived from the most general one:

- $[a_k b_k Q_k d_k]$: this model is used if the first d_k eigenvalues are assumed to be common within each group. In this case, there is only 2 eigenvalues in $\boldsymbol{\Delta}_k$, a_k and b_k.
- $[a_{kj} b Q_k d_k]$: the parameters b_k are fixed to be common between groups. It assumes that the variance outside the group-specific subspaces is common, a usual hypothesis when data are obtained in a common acquisition process.
- $[a_k b Q_k d_k]$: the parameters a_k are fixed to be common within each group and b_k are fixed to be common between groups.
- $[a b_k Q_k d_k]$: the parameters a_{kj} are fixed to be common between and within groups.
- $[a b Q_k d_k]$: the parameters a_{kj} and b_k are fixed to be common between and within groups.

In practice, the z_{ik}'s are not known and our goal is to predict them. That is why an EM algorithm is proposed below in order to estimate model parameters and then to predict the z_{ik}'s.

5 Model inference and choice of the number of clusters

5.1 Model inference through an EM algorithm

In model-based clustering, the estimation of model parameters is traditionally done by maximizing the likelihood through the EM algorithm (Dempster et al.,

1977). The EM algorithm alternates between two steps: the expectation (E) and maximization (M) steps. The E step aims at computing the conditional expectation of the complete log-likelihood using the current estimate of parameters. Then, the M step computes parameter estimates maximizing the expected complete log-likelihood found in the E step.

This section presents the update formulae of the EM algorithm in the case of the $[a_{kj}b_kQ_kd_k]$ model. Update formulae can be easily derived in the same manner for other models. The following proposition provides the expression of the complete log-likelihood associated with the model described above. Proof of this result is provided in Appendix A.1.

Proposition 1 *The complete log-likelihood of the observed curves under the* $[a_{kj}b_kQ_kd_k]$ *model can be written as:*

$$\begin{aligned}\ell_c(\theta) = -\frac{1}{2}\sum_{k=1}^{K} n_k \Bigg[&\sum_{j=1}^{d_k}\left(\log(a_{kj}) + \frac{q_{kj}^t W^{1/2} C_k W^{1/2} q_{kj}}{a_{kj}}\right) \\ &+ \sum_{j=d_k+1}^{R}\left(\log(b_k) + \frac{q_{kj}^t W^{1/2} C_k W^{1/2} q_{kj}}{b_k}\right) - 2\log(\pi_k)\Bigg] \\ &+ \frac{nR}{2} log(2\pi), \end{aligned} \tag{6}$$

where $\theta = (\pi_k, \boldsymbol{\mu}_k, a_{kj}, b_k, q_{kj})_{kj}$ *for* $1 \leq k \leq K$ *and* $1 \leq j \leq d_k$, q_{kj} *is the* j*-th column of* Q_k, $C_k = \frac{1}{n_k}\sum_{i=1}^{n} Z_{ik}(c_i - \boldsymbol{\mu}_k)^t(c_i - \boldsymbol{\mu}_k)$ *and* $c_i = (c_{ir}^1, ..., c_{ir}^p)$ *is a vector of coefficients.*

As the group memberships Z_{ik} are unknown, the EM algorithm starts by computing their conditional expectation (*E step*) before maximizing the expected complete likelihood (*M step*).

E step This step aims at computing the conditional expectation of the complete log-likelihood and reduces to the computing of the conditional expectation $E[Z_{ik}|c_i, \theta^{(q-1)}]$, which can be computed as follows.

Proposition 2 *For the model* $[a_{kj}b_kQ_kd_k]$*, the posterior probability that each curve belongs to the k-th cluster can be written:*

$$t_{ik}^{(q)} = E[Z_{ik}|c_i, \theta^{(q-1)}] = 1/\sum_{l=1}^{K}\exp[\frac{1}{2}(H_k^{(q-1)}(c_i) - H_l^{(q-1)}(c_i))], \tag{7}$$

where $H_k^{(q-1)}(c)$ *is the cost function defined for* $c \in \mathbb{R}^R$ *as:*

$$\begin{aligned} H_k^{(q-1)}(c) = &||\boldsymbol{\mu}_k^{(q-1)} - P_k(c)||_{D_k}^2 + \frac{1}{b_k^{(q-1)}}||c - P_k(c)||^2 \\ &+ \sum_{j=1}^{d_k}\log(a_{kj}^{(q-1)}) + (R - d_k)\log(b_k^{(q-1)}) - 2\log(\pi_k^{(q-1)}), \end{aligned} \tag{8}$$

where $||.||^2_{\mathcal{D}_k}$ *is a norm on the latent space* $\mathbb{E}_k$ *defined by* $||y||^2_{\mathcal{D}_k} = y^t\mathcal{D}_k y$, $\mathcal{D}_k = \tilde{Q}\boldsymbol{\Delta}_k^{-1}\tilde{Q}^t$ *and* $\tilde{Q}$ *is a matrix containing the* d_k *vectors of* U_k *completed by zeros such as* $\tilde{Q} = [U_k, 0_{R-d_k}]$, P_k *is the projection operator on the functional latent space* $\mathbb{E}_k$ *defined by* $P_k(c) = \boldsymbol{W}U_kU_k^t\boldsymbol{W}^t(c - \boldsymbol{\mu}_k) + \boldsymbol{\mu}_k$.

Proof of this result is provided in Appendix A.2.

M step This step estimates the model parameters by maximizing the expectation of the complete log-likelihood conditionally on the posterior probabilities $t_{ik}^{(q)}$ computed in the previous step. The following proposition provides update formulae for mixture parameters. Proof of these results are provided in Appendix A.3.

Proposition 3 *For the model* $[a_{kj}b_kQ_kd_k]$, *the maximization of the conditional expected complete log-likelihood leads to the following update:*

- $\pi_k^{(q)} = \frac{\eta_k^{(q)}}{n}$, $\mu_k^{(q)} = \frac{1}{\eta_k^{(q)}}\sum_{i=1}^n t_{ik}^{(q)}c_i$,
- *the* d_k *first columns of the orientation matrix* Q_k *are updated by the eigenfunctions coefficients associated with the largest eigenvalues of* $\boldsymbol{W}^{1/2}C_k^{(q)}\boldsymbol{W}^{1/2}$,
- *the variance parameters* a_{kj}, $j = 1, ..., d_k$, *are updated by the* d_k *largest eigenvalues of* $\boldsymbol{W}^{1/2}C_k^{(q)}\boldsymbol{W}^{1/2}$,
- *the variance parameters* b_k *are updated by* $b_k^{(q)} = \frac{1}{R-d_k}[tr(\boldsymbol{W}^{1/2}C_k^{(q)}\boldsymbol{W}^{1/2}) - \sum_{j=1}^{d_k}\widehat{a}_{kj}^{(q)}]$.

where $\eta_k^{(q)} = \sum_{i=1}^n t_{ik}^{(q)}$ *and* $C_k^{(q)} = \frac{1}{\eta_k^{(q)}}\sum_{i=1}^n t_{ik}^{(q)}(c_i - \mu_k^{(q)})^t(c_i - \mu_k^{(q)})$ *is the sample covariance matrix of group* k.

To summarize, the algorithm introduced above, named hereafter funHDDC, clusters multivariate functional data through their projection into low dimensional subspaces. Those projections are obtained by performing a MFPCA per cluster thank to an iterative algorithm.

5.2 Estimation of intrinsic dimensions

In order to choose the intrinsic dimensions d_k of each cluster, the Cattell's scree-test (Cattell, 1966) is used. This test looks for a drop in the eigenvalues scree. The selected dimension is the one for which the subsequent eigenvalues differences are smaller than a threshold provided by the user or selected using Bayesian information criterion, Akaike information criterion, integrated completed likelihood or slope heuristic (described below).

This estimation of the number of intrinsic dimensions is done within the M step of EM algorithm. It may allow the estimated intrinsic dimensions to vary along the iterations in order to fit well data.

5.3 Choice of the number of clusters

We now focus on the choice of the hyper-parameter K, the number of clusters. The choice of this hyper-parameter is here viewed as model selection problem. Classical model selection tools include the Akaike information criterion (AIC, Akaike (1974)), the Bayesian information criterion (BIC, Schwarz (1978)) and the integrated completed likelihood criterion (ICL, Biernacki et al. (2000)). In the context of mixture model, BIC is certainly the most popular. The BIC criterion can be computed as follows:

$$BIC = l(\hat{\theta}) - \frac{m}{2} \times log(n),$$

with $l(\hat{\theta})$ the maximum log-likelihood value, m the number of model parameters and n the number of individuals. The criterion penalizes the log-likelihood through model complexity. The model maximizing the criterion is chosen.

Another criterion, that has proved its usefulness, is the slope heuristic (SH, Birge and Massart (2007)). This data-driven criterion penalty has a multiplicative factor provided by the linear part of the log-likelihood:

$$SH = l(\hat{\theta}) - 2\, s\, m,$$

where s is the slope of the linear part of the maximum log-likelihood value $l(\hat{\theta})$ when plotted against the model complexity. It has to be noticed that this method requires to test a large number of clusters number, or a large number of models, so that there is enough points in the log-likelihood versus model complexity plot (bottom left plot of Fig. 4) to detect a plateau in the log-likelihood.

For either of those criteria, different values for K need to be tested. Then, the one that maximizes the chosen criterion's value is the best one that has to be kept.

5.4 Computational details

As explained in section 4.1, funHDDC algorithm relies on an EM algorithm. The EM algorithm needs to be initialized, by setting initial values for the partitions. To this end, two initialization strategies are considered: random and kmeans initializations. In the case of random initialization, the partitions are randomly sampled using a multinomial distribution with uniform probabilities. The kmeans strategy consists in initializing the partitions with those obtained by a kmeans algorithm applied directly on the whole set of discretized observations. With kmeans initialization, the EM algorithm usually converges quicker than with random initialization. For both initialization, it is highly recommended, in order to prevent the convergence to a local maximum, to perform multiple initializations of the algorithm and keep the solution maximizing the log-likelihood. The number of initializations is a parameter of funHDDC algorithm that can be tuned by the user.

The funHDDC algorithm is stopped when the difference between two consecutive log-likelihood values is lower than a given threshold ϵ or after a maximal number of iterations.

Running times for different sizes of datasets will be presented later, in Section 6.4.

6 Numerical experimentation on simulated data

This section presents numerical experiments to illustrate the behavior of the proposed methodology and confront it to competitors from the literature. Firstly, the quality of the model inference algorithm is illustrated on simulated data. Secondly, the sensitivity of the proposed approach to sample size is investigated in term of correct classification rate as well as in term of computational time. Thirdly, BIC and SH are compared for selecting the number of clusters. Finally, funHDDC is confronted to competitors on several datasets. The R code (R Core Team, 2017) for our multivariate functional clustering algorithm is available on CRAN in the funHDDC package.

6.1 Simulation setup

In order to ease the reproducibility of the results, we consider 3 simulation scenarios designed as follows.

Scenario A: For this first scenario, a sample of 1,000 bivariate curves are simulated based on $[a_k b_k Q_k D_k]$ model. To do so, scores are simulated according to a Gaussian model with mean μ and diagonal variance Δ. Curves coefficients can be rebuild based on $(\delta_{il})_{l\geq 1} = \boldsymbol{CWb'}_l$ as shown in Section 2. The number of clusters is fixed to $K = 3$ and mixing proportions are equal. Scores are generated from a multivariate normal distribution with the following parameters:

$$\begin{aligned}
&\text{Group 1} : d = 5, a = 150, b = 5, \mu = (1, 0, 50, 100, 0, \ldots, 0),\\
&\text{Group 2} : d = 20, a = 15, b = 8, \mu = (0, 0, 80, 0, 40, 2, 0, \ldots, 0),\\
&\text{Group 3} : d = 10, a = 30, b = 10, \mu = (0, \ldots, 0, 20, 0, 80, 0, 0, 100),
\end{aligned}$$

where d is the intrinsic dimension of subgroups, μ is the mean vector of size 70, a is the value of the d-first diagonal elements of Δ and b the value of the (70-d)-last ones. Curves are smoothed using a basis of 35 Fourier functions (cf. top panel of Figure 1).

Scenario B: The second simulation setting is inspired by the data simulation process of Ferraty and Vieu (2003); Preda (2007); Bouveyron et al. (2015), and therefore will not favor our approach in the comparison. For this simulation,

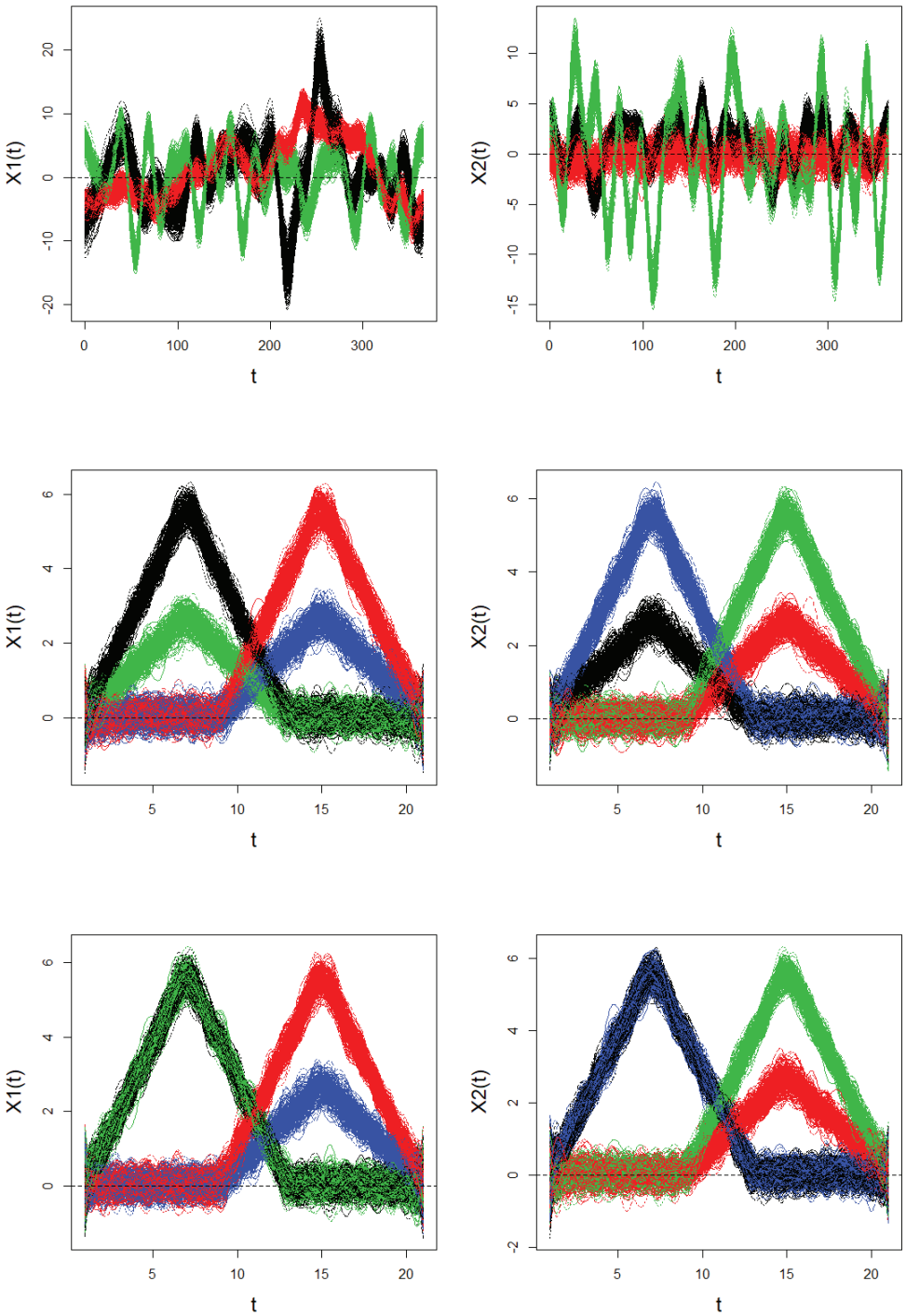


Fig. 1 Smooth data simulated for variable 1 (left) and variable 2 (right) for scenario A (top), scenario B (middle) and scenario C (bottom) colored by group for one simulation

the number of clusters is fixed to $K = 4$. A sample of 1000 bivariate curves is simulated according to the following model for $t \in [1, 21]$:

$$\begin{aligned}
\text{Group 1}: \quad & X_1(t) = U + (1-U)h_1(t) + \epsilon(t), \\
& X_2(t) = U + (0.5-U)h_1(t) + \epsilon(t), \\
\text{Group 2}: \quad & X_1(t) = U + (1-U)h_2(t) + \epsilon(t), \\
& X_2(t) = U + (0.5-U)h_2(t) + \epsilon(t), \\
\text{Group 3}: \quad & X_1(t) = U + (0.5-U)h_1(t) + \epsilon(t), \\
& X_2(t) = V + (1-V)h_2(t) + \epsilon(t), \\
\text{Group 4}: \quad & X_1(t) = U + (0.5-U)h_2(t) + \epsilon(t), \\
& X_2(t) = U + (1-U)h_1(t) + \epsilon(t),
\end{aligned}$$

where $U \sim \mathcal{U}(0, 0.1)$ and $\epsilon(t)$ is a white noise independent of U and such that $\mathbb{V}ar(\epsilon(t)) = 0.25$. The functions h_1 and h_2 are defined, for $t \in [1, 21]$, by $h_1(t) = (6 - |t-7|)_+$ and $h_2(t) = (6 - |t-15|)_+$ where $(\cdot)_+$ means the positive part. The mixing proportions are equal, and the curves are observed in 101 equidistant points. The functional form of the data is reconstructed using a cubic B-spline basis smoothing with 25 basis functions (cf. middle panel of Figure 1).

Scenario C: For this third simulation scenario, the number of clusters is fixed to $K = 4$. A sample of 1000 bivariate curves is simulated according to the following model for $t \in [1, 21]$:

$$\begin{aligned}
\text{Group 1}: \quad & X_1(t) = U + (1-U)h_1(t) + \epsilon(t), \\
& X_2(t) = U + (0.5-U)h_1(t) + \epsilon(t), \\
\text{Group 2}: \quad & X_1(t) = U + (1-U)h_2(t) + \epsilon(t), \\
& X_2(t) = U + (0.5-U)h_2(t) + \epsilon(t), \\
\text{Group 3}: \quad & X_1(t) = U + (1-U)h_1(t) + \epsilon(t), \\
& X_2(t) = U + (1-U)h_2(t) + \epsilon(t), \\
\text{Group 4}: \quad & X_1(t) = U + (0.5-U)h_2(t) + \epsilon(t), \\
& X_2(t) = U + (0.5-U)h_1(t) + \epsilon(t),
\end{aligned}$$

where $U, \epsilon(t), h_1$ and h_2 are defined as before. The mixing proportions are equal, and the curves are observed in 101 equidistant points. The functional form of the data is reconstructed using a cubic B-splines basis smoothing with 25 basis functions. As shown in Figure 1 (bottom), the 4 groups cannot be distinguished with one variable only: indeed group 3 (green) is similar to group 1 (black) for variable $X_1(t)$ and similarly group 4 (blue) is similar to group 1 (black) for variable $X_2(t)$. Consequently, any univariate functional clustering methods applied either on variable $X_1(t)$ or $X_2(t)$ should fail.

For each scenario, the estimated partitions are compared to the true partition with the adjusted Rand index (ARI, Rand (1971)). This criterion value is less than or equal to 1, with 1 representing a perfect agreement between the true partition and the one estimated by the algorithm, and 0 a random agreement. The algorithm settings used for all simulations are the following: the threshold of the Cattell's scree-test for the selection of intrinsic dimensions d_k is fixed to 0.2 (the optimal threshold value should be chosen using BIC or slope heuristic), the stopping criterion for the EM algorithm is a growth of the log-likelihood lower than $\epsilon = 10^{-3}$ or a maximal number of iterations of 200, the initialization of the algorithm is done through a *random* partition in the introductory example, and with the *kmeans* strategy the model selection and benchmark experiments, in order to speed up the convergence.

6.2 An introductory example

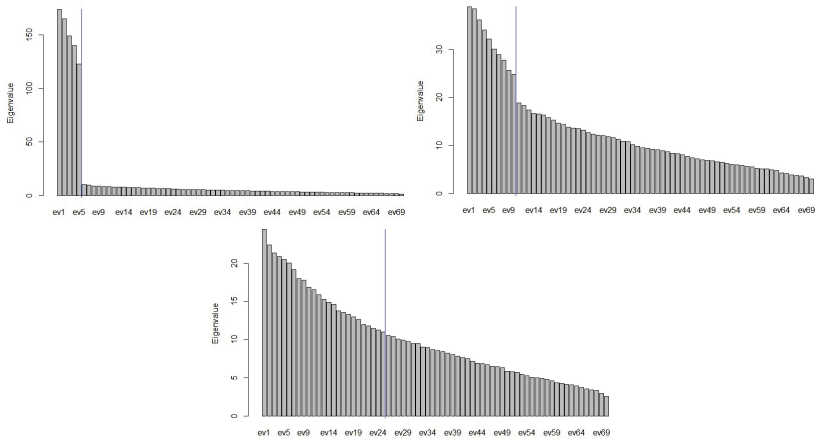
Table 1 Mean (and s.d.) of ARI for 50 simulations

funHDDC model	Mean (SD)
$[a_{kj}b_kQ_kD_k]$	0.99 (0.08)
$[a_{kj}bQ_kD_k]$	0.85 (0.26)
$[a_kb_kQ_kD_k]$	**1 (0)**
$[a_kbQ_kD_k]$	0.88 (0.26)
$[ab_kQ_kD_k]$	0.95 (0.16)
$[abQ_kD_k]$	0.49 (0.36)

In order to illustrate the good behavior of the inference algorithm, we first consider a single simulation according to *Scenario A*, which is overall a difficult situation. The algorithm is run for $K = 3$ groups with the model $[a_kb_kQ_kD_k]$ which has been used to generate the data and the simulation setting is repeated 50 times. Figure 2 allows the comparison of the fitted values with the actual ones of model parameters. Parameter a turns out to be well estimated for all 3 clusters, whereas parameters b and d are only well estimated for 2 clusters out of 3. Indeed, this simulation scenario has one mixture component with a low signal-to-noise ratio which disturbs the estimation of d_k for this component, and therefore also perturb the estimation of the noise variance b_k. Nevertheless, the fact that our model actually models the variance within the whole space (and not only a part as in Jacques and Preda (2014b)), allows us to correctly recover the cluster partition even in difficult estimation conditions.

To assess the clustering quality, the funHDDC algorithm is now run for $K = 3$ groups with all 6 sub-models and the simulation setting is repeated 50 times. The quality of the estimated partitions is evaluated using the ARI and results are given in Table 1. As expected, the best result is obtained for the model $[a_kb_kQ_kD_k]$ which has been used to generate data and it shows that

the algorithm correctly recovers the cluster pattern. It is worth noticing that the other models also have satisfying performances.



Parameters	Group 1	Group 2	Group 3
a	150.8 (5.1)	12.6 (2.9)	31.4 (2.4)
b	4.9 (0.1)	4.3 (1.7)	9.3 (0.4)
d	5 (0)	50.2 (20.1)	11.3 (3.0)

Fig. 2 Scree-test of Cattell performed for each group with the threshold set to 0.2 (blue line) for one simulation and mean (sd) of parameters estimation for the 50 simulations with the $[a_k b_k Q_k D_k]$ model

6.3 Sample size influence

In order to evaluate the sensitivity of the proposed approach to the sample size, we now consider 50 simulations according to *Scenario B*, and with different sample size: 1000, 500, 200 and 30. Table 2 presents the corresponding results. The impact of the sample size is not really significant between 1000 to 200. For very small sample size, 30 observations for 3 clusters, the quality of the partition estimation significantly decreases, but such small sample size are seldom used in practice for clustering studies.

Table 2 Mean (and s.d.) of ARI for 50 simulations

	sample size n			
funHDDC model	1000	500	200	30
$[a_{kj}b_kQ_kD_k]$	0.98 (0.08)	0.99 (0.05)	0.98 (0.07)	0.84 (0.20)
$[a_{kj}bQ_kD_k]$	0.82 (0.19)	0.71 (0.15)	0.70 (0.12)	0.67 (0.09)
$[a_kb_kQ_kD_k]$	1 (0)	0.99 (0.07)	0.98 (0.08)	0.80 (0.23)
$[a_kbQ_kD_k]$	0.88 (0.18)	0.66 (0.10)	0.69 (0.11)	0.66 (0.08)
$[ab_kQ_kD_k]$	0.98 (0.09)	1 (0)	0.99 (0.05)	0.90 (0.16)
$[abQ_kD_k]$	0.86 (0.18)	0.71 (0.14)	0.68 (0.11)	0.66 (0.08)

6.4 Computational time and cost

When dealing with multivariate functional data, a big issue consists of scalability and computational effort in performing analyses. We will present here the impact of sample size and number of functional variables on running time.

Firstly, funHDDC algorithm is applied for $K = 4$ groups on *scenario B* bivariate functional data. Then, a second scenario with four functional variables is built based on *scenario B* with the two additional functional variables be cosinus and sinus functions. The computer used for the experiments has a Windows 10 operating system, Intel(R) Core(TM) i7-6700 CPU 3.40GHz processor and 8.00 Go of RAM memory. The associated running times, estimated with Sys.time R function, are shown in Table 3.

Table 3 Computational effort in performing analyses

Sample size	Number of functional variables	Running time (sec)
1000	2	0.24
10 000	2	1.16
100 000	2	10.71
1000	4	0.59
10 000	4	3.50
100 000	4	30.85

6.5 Model selection

In this section, the selection of the number of clusters is investigated. As previously mentioned, two criteria are used: BIC and the slope heuristic. Data are generated from *Scenario A*. This simulation setting has been repeated 50 times and the 6 sub-models have been estimated for a number of clusters from 2 to 10.

Figures 3 and 4 show for one simulation with the model $[a_kb_kQ_kD_k]$, the values of BIC and the slope heuristic in view of the number of clusters. For this simulation, both the slope heuristic and BIC succeed in selecting the right number of clusters. On Figure 4, the left plot corresponds to the log-likelihood

function with respect to the number of free model parameters. The red line is estimated using a robust linear regression and its slope coefficient is used to compute the penalized log-likelihood function shown on the right plot.

Table 4 summarized the results of the 50 simulations for the BIC and the slope heuristic. The BIC criterion has some difficulties to estimate the actual number of clusters K. Indeed, depending on the simulation, BIC selects between 2 to 3 clusters and succeed in 46% of simulations in the case of $[a_k b_k Q_k D_k]$ model. The slope heuristic is conversely more efficient to recover the actual number of groups, in about 66% of simulations in the case of $[a_k b_k Q_k D_k]$ model.

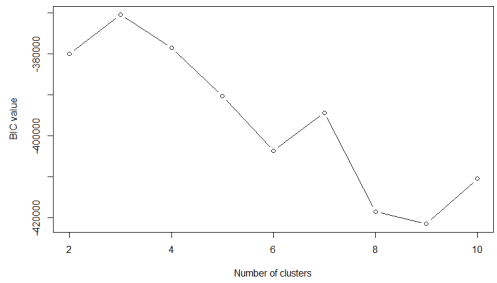


Fig. 3 BIC for one simulation for the model $[a_k b_k Q_k D_k]$

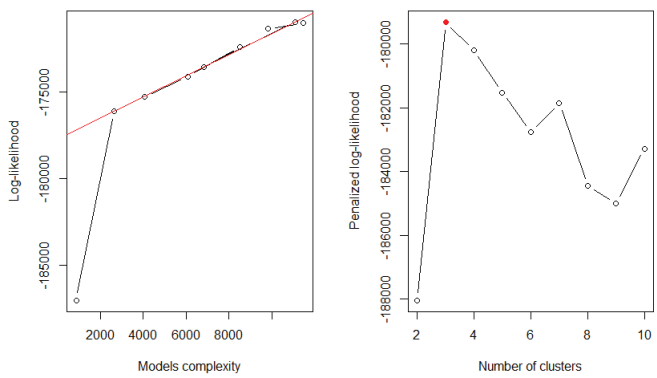


Fig. 4 Slope heuristic for one simulation for the model $[a_k b_k Q_k D_k]$

Table 4 Best model selected by BIC (top) and by the slope heuristic (SH, bottom) for 50 simulations as a percentage

BIC		Number K of clusters								
Method	Model	2	3	4	5	6	7	8	9	10
funHDDC	$[a_{kj}b_kQ_kD_k]$	36	**48**	10	6	-	-	-	-	-
funHDDC	$[a_{kj}bQ_kD_k]$	38	**54**	6	-	2	-	-	-	-
funHDDC	$[a_kb_kQ_kD_k]$	42	**46**	8	0	2	2	-	-	-
funHDDC	$[a_kbQ_kD_k]$	44	**48**	8	-	-	-	-	-	-
funHDDC	$[ab_kQ_kD_k]$	**46**	40	10	4	-	-	-	-	-
funHDDC	$[abQ_kD_k]$	**64**	24	10	2	-	-	-	-	-

SH		Number K of clusters								
Method	Model	2	3	4	5	6	7	8	9	10
funHDDC	$[a_{kj}b_kQ_kD_k]$	6	**60**	24	10	-	-	-	-	-
funHDDC	$[a_{kj}bQ_kD_k]$	10	**74**	12	4	-	-	-	-	-
funHDDC	$[a_kb_kQ_kD_k]$	18	**66**	14	2	-	-	-	-	-
funHDDC	$[a_kbQ_kD_k]$	26	**52**	14	8	2	-	-	-	-
funHDDC	$[ab_kQ_kD_k]$	34	**42**	16	6	2	-	-	-	-
funHDDC	$[abQ_kD_k]$	**38**	28	20	10	2	0	0	2	-

6.6 Benchmark with existing methods

In this section, the proposed clustering algorithm is compared to competitors of the literature: Funclust (from Funclustering package, Jacques and Preda (2014b)), *kmeans-d1* and *kmeans-d2* (our own implementation of Ieva et al. (2013)) and FGRC (provided at our request by the authors Yamamoto and Hwang (2017)). All algorithms are applied for $K = 3$ groups for *Scenario A* and $K = 4$ groups for *Scenario B* and *Scenario C*. These methods are compared on the basis of the 3 simulation settings and according to the adjusted Rand index (ARI).

Table 5 presents clustering accuracies for the 10 tested models and the best funHDDC model selected at each iteration by slope heuristic or BIC. These scenarios seem to be hard situations since only funHDDC performs well for the 3 of them. Those good results for funHDDC are due to the fact that the MFPCA are carried out cluster per cluster. FGRC performed well for 2 out of 3 scenarios, it is the second best method behind funHDDC. Let also remark that both *kmeans* methods have a high variance. The SH does not perform as well as in the previous example. SH seems to be a good criterion to select the number of clusters, but, with a number of clusters fixed, the BIC seems to be a better criterion for model selection. One can also wonder if this counter performance of the SH is not linked to the small number of tested models.

7 Case study: analysis of pollution in French cities

This section focuses on the analysis of pollution data in French cities. The monitoring and the analysis of such data is of course important in the sense

Table 5 Mean (and s.d) of ARI for all tested models on 50 simulations

Method	Model	Scenario A	Scenario B	Scenario C
funHDDC	$[a_{kj}b_kQ_kD_k]$	0.99 (0.08)	**0.98 (0.08)**	0.94 (0.14)
funHDDC	$[a_{kj}bQ_kD_k]$	0.85 (0.26)	0.82 (0.19)	0.76 (0.19)
funHDDC	$[a_kb_kQ_kD_k]$	**1 (0)**	0.96 (0.11)	0.94 (0.13)
funHDDC	$[a_kbQ_kD_k]$	0.88 (0.26)	0.88 (0.18)	0.81 (0.20)
funHDDC	$[ab_kQ_kD_k]$	0.95 (0.16)	**0.98 (0.09)**	**0.95 (0.13)**
funHDDC	$[abQ_kD_k]$	0.49 (0.36)	0.86 (0.18)	0.78 (0.23)
funHDDC	SH best model	0.48 (0.29)	0.76 (0.18)	0.70 (0.14)
funHDDC	BIC best model	0.97 (0.12)	0.86 (0.18)	0.79 (0.18)
Funclust	-	0.30 (0.27)	0.42 (0.25)	0.41 (0.24)
$kmeans - d_1$	-	0.57 (0.49)	0.18 (0.37)	0.30 (0.46)
$kmeans - d_2$	-	0.61 (0.48)	0.29 (0.43)	0.18 (0.37)
FGRC	-	0.87 (0.01)	0.65 (0.21)	0.81 (0.19)

that they could help cities in designing their policy against pollution. As a reminder, pollution kills at least nine million people and costs trillions of dollars every year, according to the most comprehensive global analyses to date [1].

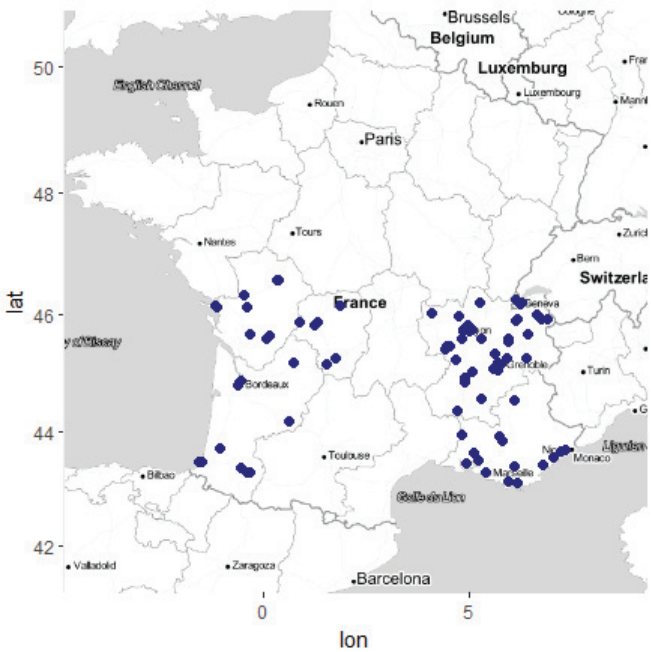


Fig. 5 Location of measured cities (dark blue)

[1] who.int/airpollution/en/

7.1 Data

This dataset deals with pollution in some French cities (available on 3 different websites [2]). It has been documented by Atmo France, a federation which monitor air quality in France. It gathered 5 categories of pollutants measured hourly since 1985. In this study we choose to work on Ozone value ($\mu g/m^3$) and PM10 particles ($\mu g/m^3$) measured in 84 France southern cities. The regions affected are Nouvelle Aquitaine, Auvergne Rhône Alpes and Provence Alpes Côte d'Azur (cf. Figure 5). A period of one year from 1/01/2017 to 31/12/2017 is considered. The goal of this study is to characterize the daily evolution of these polluants. In order to do this, data are split into daily curves, and the clustering algorithm has been carried out on all the daily curves for all the cities. Doing this, geographical and temporal dependencies between the daily curves are ignored in this preliminary study. Finally, we remove from the analysis the daily curves which have more than 4 missing values or for which there is missing values at the beginning or at the end of the period. The number of bivariate curves to analyse is thus 25,658.

The functional form of the data is reconstructed using a cubic B-spline smoothing with 10 basis functions. As we can see in Figure 6 (bottom), the presence of missing values does not disrupt smoothing. Data are collected through calibrated meteorological stations, we consider that data are obtained in a common acquisition process, then the noise is assumed to be the same for all stations. So, our algorithm has been applied with $[a_{kj}bQ_kD_k]$ model on smoothed data with a varying number of clusters, from 2 to 20. The BIC criteria is used to choose an appropriate number of clusters because there is here not enough models to use the slope heuristic criteria.

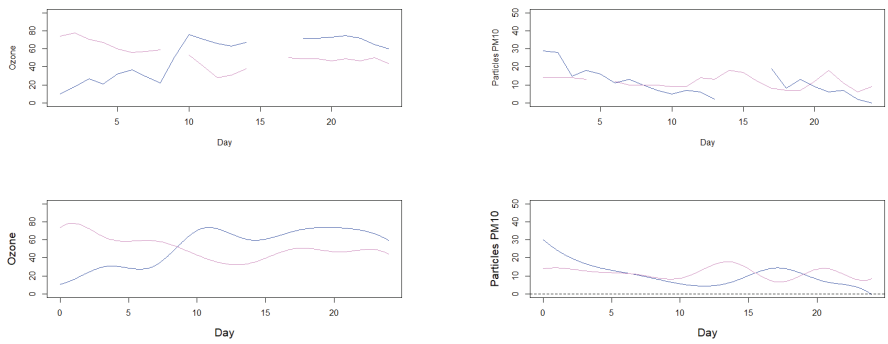


Fig. 6 Pollutants real curves (top) and smooth curves (bottom) for Avignon day 12 (blue) and La Rochelle day 177 (pink)

[2] https://www.airpaca.org/donnees/telecharger, https://www.atmo-auvergnerhonealpes.fr/donnees/telecharger, https://www.atmo-nouvelleaquitaine.org/donnees/telecharger

7.2 Results

Table 6 BIC values for the 10 first number of clusters, with $[a_{kj}bQ_kD_k]$

Number of clusters	Complexity	BIC
6	544	-4,756,123.71
9	849	-4,778,048.84
18	2,057	-4,819,115.13
17	1,929	-4,833,556.40
5	472	-4,939,371.01
14	1,545	-4,966,517.37
16	1,834	-4,969,406.42
15	1,735	-4,970,505.90
11	1,260	-4,972,458.71
2	101	-4,976,490.18

According to BIC, the best partition for $[a_{kj}bQ_kD_k]$ model is with 6 clusters (cf. Table 6).

The main sources of variation of Ozone and particles PM10 overall mean curves can be studied thanks to the MFPCA performed for each subgroup. The solid curve on Figure 7 and 8 represents the overall mean curve and the blue and red ones show the effect of adding (resp. subtracting) a multiple of the first eigenfunction and can be interpret as the first source of variation of the overall mean. According to this plot the first source is an amplitude variation for Ozone. For PM10 particles it is the peaks size that varies the most between groups.

Looking at the groups mean curves (cf. Figure 7 for Ozone and 8 for PM10 particles) we can see that Ozone mean curves are more variable than PM10 particles ones. We can also see a common pattern between groups. For the PM10 particles, the mean curves of each group have a wavy shape with a first summit at night and a second at mid-afternoon. There is two main patterns in the O3 curves. During a day, the Ozone concentration has a tendency to decrease from midnight to midday and to increase until reaching a plateau between 5 pm and 8 pm for the first pattern. For the second one, the Ozone concentration is stable from midnight to 2 pm and increases until reaching a maximum at 8 pm. But it is the level of concentration that varies the most from a group to another.

The dark blue group is characterized by the lower concentration of Ozone along the day. This group gathers winter days (cf. Figure 9) for cities mostly in urban area (cf. Table 7). Ozone is a product of photochemical reaction between various pollutants when there is a lot of sunlight. The low duration of sunshine during winter can explain those low values. Its Ozone mean curve is very close to turquoise group one, they differ from their PM10 particle concentration. Indeed, dark blue average curve is three times higher than turquoise group one. It gathers days the most contaminated by particles PM10 (with the highest concentration along the day). For that matter, the European Union recommends that PM10 concentrations should not to be higher than 50 $\mu g/m^3$ in daily mean more than 35 days per year and in this group the mean value is above this threshold at any time. Whereas turquoise group gathers fall and winter days (cf. Figure 9). The black group has the highest values of Ozone (cf. Figure 7 group 5). Its maximum is reached between 5pm and 10pm. This can be due to exhaust gas when people commute from their work to their home.

Table 7 Proportion of city type in each group

Type of city	Whole dataset	Dark blue Group	Pink Group	Turquoise Group
Urban	0.78	0.81	0.75	0.84
Suburban	0.17	0.16	0.19	0.14
Rural	0.05	0.03	0.06	0.03
Type of city	Grey Group	Black Group	Purple Group	
Urban	0.79	0.78	0.77	
Suburban	0.16	0.17	0.17	
Rural	0.06	0.05	0.05	

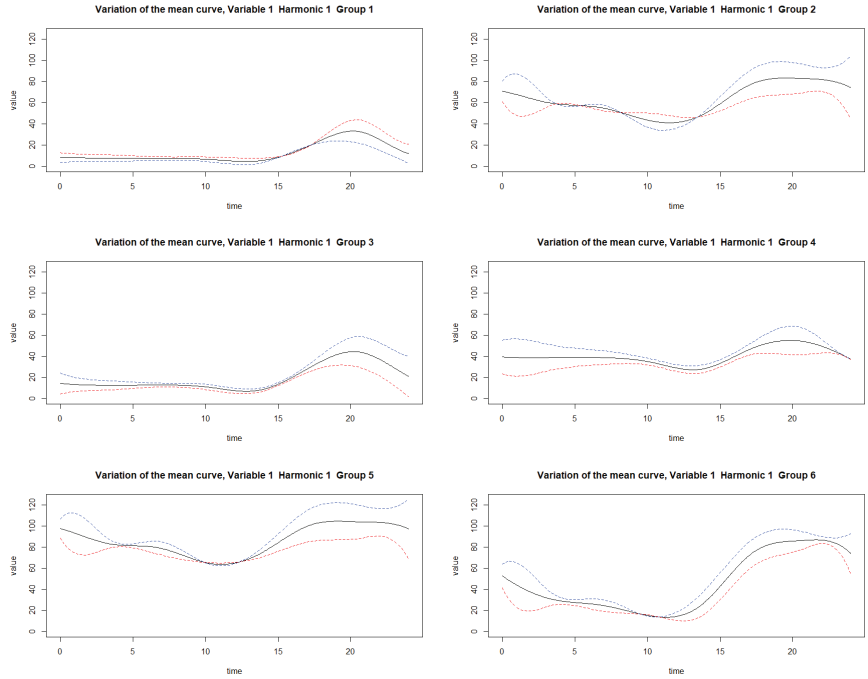


Fig. 7 O3 overall mean curve and its variation expressed in each group functional subspace. Blue (resp. red) curve shows the effect of adding (resp. subtracting) a multiple of the first eigenfunction and represents the first source of the overall mean variation

To conclude, the use of multiple variables to cluster cities allow the distinction between different pollution profiles. Those results enable local councils to have a look at the daily pollution of their towns along the year. Those results have especially highlighted critical days in particles PM10 pollution, that can lead to recommendations in order to try to lower these levels the next year. However we have to stay vigilant about the interpretation of those results. In fact, the measurement of contaminating elements are very localized, some sensors are located near companies and thus are not always representative of the pollution of the whole city in which they are located.

8 Discussion and conclusion

This work was motivated by the will to provide a new clustering method for multivariate functional data, called funHDDC, which takes into account the possibility that data live in subspaces of different dimensions. The method is based on a multivariate functional principal component analysis and a functional latent mixture model. Its efficiency has been demonstrated on simulated datasets and the proposed technique outperforms *state-of-the-art* methods for

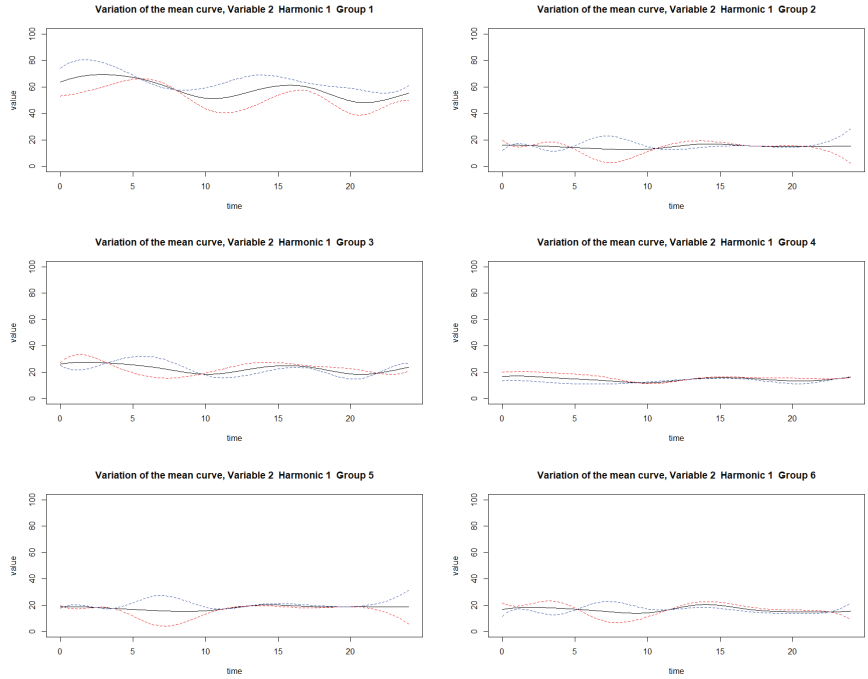


Fig. 8 PM10 particles overall mean curve and its variation expressed in each group functional subspace. Blue (resp. red) curve shows the effect of adding (resp. subtracting) a multiple of the first eigenfunction and represents the first source of the overall mean variation

clustering multivariate functional data. Notice also that this new algorithm works in the univariate case as well and, therefore, generalizes the original funHDDC algorithm (Bouveyron and Jacques (2011)). It is available on CRAN as the funHDDC package. The proposed methodology has been applied to analyze one-year pollution records in 84 cities in France, with meaningful results. It is worth noticing that smoothing data with basis functions allows to both filter the level of information one wants to keep and to deal with missing data. Let also remark that wavelet smoothing may keep more information in the case of peaked data than B-spline smoothing. It can be the subject of future work because a new model will have to be adapted to this smoothing. Similarly, further developments of the proposed approach could be investigated in order to take into account dependency between observations, following the large literature about dependent functional data.

Conflict of interest

The authors declare that they have no conflict of interest.

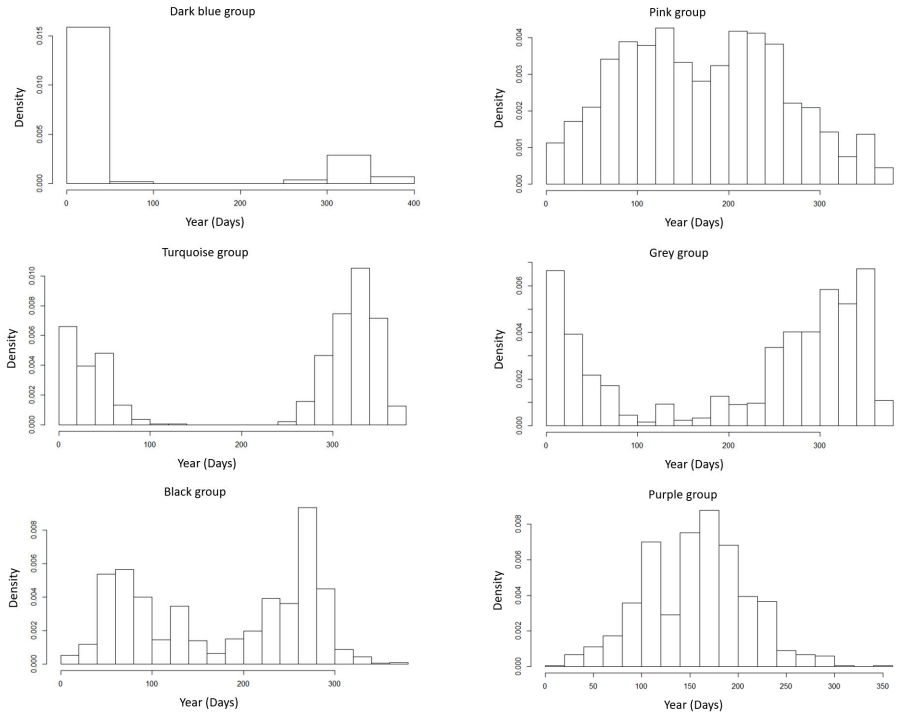


Fig. 9 Histogram of days in each group, from 1/01/2017 (0) to 31/12/2010 (364). Spring is from day 79 to 171, Summer from day 172 to 264, Fall from day 265 to 354 and Winter from day 355 to 365 and day 1 to 78.

A Appendix: proofs

A.1 Proof of Proposition 1.

$$l(\theta) = \sum_{i=1}^{n}\sum_{k=1}^{K} z_{ki} log(\pi_k f(x_i, \theta_k)),$$

where z_{ki}=1 if x_i belongs to the cluster k and $z_{ki} = 0$ otherwise. $f(x_i, \theta_k)$ is a Gaussian density, with parameters $\theta_k = \{\mu_k, \Sigma_k\}$. So the complete log-likelihood is written:

$$\begin{aligned} l(\theta) &= \sum_{i=1}^{n}\sum_{k=1}^{K} z_{ki} log[\pi_k \frac{1}{(2\pi)^{R/2}|\Sigma_k|^{1/2}} exp(\frac{-1}{2}(x_i-\mu_k)^t \Sigma_k^{-1}(x_i-\mu_k))] \\ &= \sum_{i=1}^{n}\sum_{k=1}^{K} z_{ki}[log(\pi_k) - \frac{1}{2}log|\Sigma_k| - \frac{1}{2}(x_i-\mu_k)^t\Sigma_k^{-1}(x_i-\mu_k) - \frac{R}{2}log(2\pi)]. \end{aligned}$$

For the $[a_{kj}b_kQ_kd_k]$ model, we have:

$$\begin{aligned} l(\theta) &= \frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{K} z_{ki}[-2log(\pi_k) + log(\prod_{j=1}^{d_k} a_{kj} \prod_{j=d_k+1}^{R} b_k) + (x_i-\mu_k)^t Q_k \Delta_k^{-1} Q_k^t (x_i-\mu_k)] \\ &- \frac{nR}{2}log(2\pi) \\ &= \frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{K} z_{ki}[-2log(\pi_k) + \sum_{j=1}^{d_k} log(a_{kj}) + \sum_{j=d_k+1}^{R} log(b_k)] \\ &+ (x_i-\mu_k)^t Q_k \Delta_k^{-1} Q_k^t (x_i-\mu_k)] - \frac{nR}{2}log(2\pi). \end{aligned}$$

Let $n_k = \sum_{i=1}^{n} z_{ki}$ be the number of curves within cluster k, the complete log-likelihood is then written:

$$\begin{aligned} l(\theta) &= -\frac{1}{2}\sum_{k=1}^{K} n_k[-2log(\pi_k) + \sum_{j=1}^{d_k} log(a_{kj}) + \sum_{j=d_k+1}^{R} log(b_k) \\ &+ \frac{1}{n_k}\sum_{i=1}^{n} z_{kj}(x_i-\mu_k)^t Q_k \Delta_k^{-1} Q_k^t (x_i-\mu_k)] - \frac{nR}{2}log(2\pi). \end{aligned}$$

The quantity $(x_i-\mu_k)^t Q_k \Delta_k^{-1} Q_k^t (x_i-\mu_k)$ is a scalar, so it is equal to it trace:

$$\frac{1}{n_k}\sum_{i=1}^{n} z_{ki}(x_i-\mu_k)^t Q_k \Delta_k^{-1} Q_k^t (x_i-\mu_k) = \frac{1}{n_k}\sum_{i=1}^{n} z_{ki} tr((x_i-\mu_k)^t Q_k \Delta_k^{-1} Q_k^t (x_i-\mu_k)).$$

Well $tr([(x_i-\mu_k)^t Q_k] \times [\Delta_k^{-1} Q_k^t (x_i-\mu_k)]) = tr([\Delta_k^{-1} Q_k^t (x_i-\mu_k)] \times [(x_i-\mu_k)^t Q_k])$, consequently:

$$\begin{aligned} \frac{1}{n_k}\sum_{i=1}^{n} z_{ki}(x_i-\mu_k)^t Q_k \Delta_k^{-1} Q_k^t (x_i-\mu_k) &= \frac{1}{n_k}\sum_{i=1}^{n} z_{ki} tr(\Delta_k^{-1} Q_k^t (x_i-\mu_k)(x_i-\mu_k)^t Q_k) \\ &= tr(\Delta_k^{-1} Q_k^t [\frac{1}{n_k}\sum_{i=1}^{n} z_{ki}(x_i-\mu_k)^t (x_i-\mu_k)] Q_k) \\ &= tr(\Delta_k^{-1} Q_k^t C_k Q_k), \end{aligned}$$

where $C_k = \frac{1}{n_k}\sum_{i=1}^{n} z_{ki}(x_i-\mu_k)^t(x_i-\mu_k)$ is the empirical covariance matrix of the k-th element of the mixture model. The Δ_k matrix is diagonal, so we can write:

$$\frac{1}{n_k}\sum_{i=1}^{n} z_{ki}(x_i-\mu_k)^t Q_k \Delta_k^{-1} Q_k^t (x_i-\mu_k) = \sum_{j=1}^{d_k} \frac{q_{kj}^t W^{1/2} C_k W^{1/2} q_{kj}}{a_{kj}} + \sum_{j=d_k+1}^{R} \frac{q_{kj}^t W^{1/2} C_k W^{1/2} q_{kj}}{b_k},$$

where q_{kj} is j-th column of Q_k.
Finally,

$$l(\theta) = -\frac{1}{2}\sum_{k=1}^{K} n_k[-2log(\pi_k) + \sum_{j=1}^{d_k} log(a_{kj}) + \sum_{j=d_k+1}^{R} log(b_k) + \sum_{j=1}^{d_k} \frac{q_{kj}^t W^{1/2} C_k W^{1/2} q_{kj}}{a_{kj}} + \sum_{j=d_k+1}^{R} \frac{q_{kj}^t W^{1/2} C_k W^{1/2} q_{kj}}{b_k}] + \frac{nR}{2} log(2\pi).$$

A.2 Proof of Proposition 2.

$$\begin{aligned} H_k(x) &= -2log(\pi_k f(x,\theta_k)) \\ &= -2log(\pi_k) - 2log(f(x,\theta_k)) \\ &= -2log(\pi_k) - 2log(\frac{1}{(2\pi)^{R/2}|\Sigma_k|^{1/2}} exp(\frac{-1}{2}(x-\mu_k)^t \Sigma_k^{-1}(x-\mu_k)) \\ &= -2log(\pi_k) - 2log(\frac{1}{(2\pi)^{R/2}|\Sigma_k|^{1/2})}) + (x-\mu_k)^t \Sigma_k^{-1}(x-\mu_k) \\ &= -2log(\pi_k) + Rlog(2\pi) + log|\Sigma_k| + (x-\mu_k)^t \Sigma_k^{-1}(x-\mu_k). \end{aligned}$$

But, $\Sigma_k = Q_k \Delta_k Q_k^t$ and $Q_k^t Q_k = I_R$, hence:

$$H_k(x) = -2log(\pi_k) + Rlog(2\pi) + log|\Sigma_k| + (x_k-\mu_k)^t (Q_k \Delta_k Q_k^t)^{-1}(x_k-\mu_k).$$

Let $Q_k = \tilde{Q_k} + \bar{Q_k}$ where $\tilde{Q_k}$ is the $R \times R$ matrix containing the d_k first columns of Q_k completed by zeros and where $\bar{Q_k} = Q_k - \tilde{Q_k}$. Notice that $\tilde{Q_k}\Delta_k^{-1}\bar{Q_k^t} = \bar{Q_k}\Delta_k^{-1}\tilde{Q_k^t} = O_p$ where O_p is the null matrix. So,

$$Q_k \Delta_k^{-1} Q_k^t = (\tilde{Q_k} + \bar{Q_k})\Delta_k^{-1}(\tilde{Q_k} + \bar{Q_k}) = \tilde{Q_k}\Delta_k^{-1}\tilde{Q_k} + \bar{Q_k}\Delta_k^{-1}\bar{Q_k}.$$

Hence,

$$\begin{aligned} H_k(x) &= -2log(\pi_k) + Rlog(2\pi) + log|\Sigma_k| + (x-\mu_k)^t \tilde{Q_k}\Delta_k^{-1}\tilde{Q_k^t}(x-\mu_k) \\ &+ (x-\mu_k)^t \bar{Q_k}\Delta_k^{-1}\bar{Q_k^t}(x-\mu_k). \end{aligned}$$

With definitions $\tilde{Q_k}[\tilde{Q_k^t}\tilde{Q_k}] = \tilde{Q_k}$ and $\bar{Q_k}[\bar{Q_k^t}\bar{Q_k}] = \bar{Q_k}$, we can rephrase $H_k(x)$ as:

$$\begin{aligned} H_k(x) &= -2log(\pi_k) + Rlog(2\pi) + log|\Sigma_k| + (x-\mu_k)^t \tilde{Q_k}\tilde{Q_k^t}\tilde{Q_k}\Delta_k^{-1}\tilde{Q_k^t}\tilde{Q_k}\tilde{Q_k^t}(x-\mu_k) \\ &+ (x-\mu_k)^t \bar{Q_k}\bar{Q_k^t}\bar{Q_k}\Delta_k^{-1}\bar{Q_k^t}\bar{Q_k}\bar{Q_k^t}(x-\mu_k) \\ &= -2log(\pi_k) + Rlog(2\pi) + log|\Sigma_k| + [\tilde{Q_k}\tilde{Q_k^t}(x-\mu_k)]^t \tilde{Q_k}\Delta_k^{-1}\tilde{Q_k}^t[\tilde{Q_k}\tilde{Q_k^t}(x-\mu_k)] \\ &+ [\bar{Q_k}\bar{Q_k^t}(x-\mu_k)]^t \bar{Q_k}\Delta_k^{-1}\bar{Q_k}^t[\bar{Q_k}\bar{Q_k^t}(x-\mu_k)]. \end{aligned}$$

We define $\mathcal{D}_k = \tilde{Q_k}\Delta_k^{-1}\tilde{Q_k^t}$ and the norm $||.||_{\mathcal{D}_k}$ on $\mathbb{E}_k$ such as $||x||_{\mathcal{D}_k} = x^t\mathcal{D}_k x$. So, on one hand:

$$[\tilde{Q_k}\tilde{Q_k^t}(x-\mu_k)]^t\tilde{Q_k}\Delta_k^{-1}\tilde{Q_k^t}[\tilde{Q_k}\tilde{Q_k^t}(x-\mu_k)] = ||\tilde{Q_k}\tilde{Q_k^t}(x-\mu_k)||^2_{\mathcal{D}_k}.$$

On the other hand:

$$[\bar{Q_k}\bar{Q_k^t}(x-\mu_k)]^t\bar{Q_k}\Delta_k^{-1}\bar{Q_k^t}[\bar{Q_k}\bar{Q_k^t}(x-\mu_k)] = \frac{1}{b_k}||\bar{Q_k}\bar{Q_k^t}(x-\mu_k)||^2.$$

Consequently,

$$H_k(x) = -2log(\pi_k) + Rlog(2\pi) + log|\Sigma_k| + ||\tilde{Q_k}\tilde{Q_k^t}(x-\mu_k)||^2_{\mathcal{D}_k} + \frac{1}{b_k}||\bar{Q_k}\bar{Q_k^t}(x-\mu_k)||^2.$$

Knowing P_k, $P_k^\perp$ and $||\mu_k - P_k^\perp||^2 = ||x - P_k(x)||^2$, we have:

$$H_k(x) = ||\mu_k - P_k(x)||^2_{\mathcal{D}_k} + \frac{1}{b_k}||x - P_k(x)||^2 + log|\Sigma_k| - 2log(\pi_k) + Rlog(2\pi).$$

Moreover, $log|\Sigma_k| = \sum_{j=1}^{d_k} log(a_{kj}) + (R-d_k)log(b_k)$.
Finally,

$$H_k(x) = ||\mu_k - P_k(x)||^2_{\mathcal{D}_k} + \frac{1}{b_k}||x - P_k(x)||^2 + \sum_{j=1}^{d_k} log(a_{kj}) + (R-d_k)log(b_k) - 2log(\pi_k) + Rlog(2\pi).$$

A.3 Proof of Proposition 3.

Parameter Q_k We have to maximise the log-likelihood under the constraint $q_{kj}^t q_{kj} = 1$, which is equivalent to looking for a saddle point of the Lagrange function:

$$\mathcal{L} = -2l(\theta) - \sum_{j=1}^{R}\omega_{kj}(q_{kj}^t q_{kj} - 1),$$

where ω_{kj} are Lagrange multiplier. So we can write:

$$\begin{aligned}\mathcal{L} = &\sum_{k=1}^{K}\eta_k[\sum_{j=1}^{d_k}(log(a_{kj}) + \frac{q_{kj}^t W^{1/2}C_k W^{1/2}q_{kj}}{a_{kj}}) \\ &+ \sum_{j=d_k+1}^{R}(log(b_k) + \frac{q_{kj}^t W^{1/2}C_k W^{1/2}q_{kj}}{b_k}) - 2log(\pi_k)] + \frac{nR}{2}log(2\pi) \\ &- \sum_{j=1}^{R}\omega_{kj}(q_{kj}^t q_{kj} - 1).\end{aligned}$$

Therefore, the gradient of $\mathcal{L}$ in relation to q_{kj} is:

$$\begin{aligned}\nabla_{q_{kj}}\mathcal{L} = &\nabla_{q_{kj}}(\sum_{k=1}^{K}\eta_k[\sum_{j=1}^{d_k}\frac{q_{kj}^t W^{1/2}C_k W^{1/2}q_{kj}}{a_{kj}} + \sum_{j=d_k+1}^{R}\frac{q_{kj}^t W^{1/2}C_k W^{1/2}q_{kj}}{b_k}] \\ &- \sum_{j=1}^{R}\omega_{kj}(q_{kj}^t q_{kj} - 1)).\end{aligned}$$

As a reminder, when W is symmetric, then $\frac{\partial}{\partial x}(x-s)^T W(x-s) = 2W(x-s)$ and $\frac{\partial}{\partial x}(x^T x) = 2x$ (cf. Petersen and Pedersen (2012)), so:

$$\nabla_{q_{kj}}\mathcal{L} = \eta_k[2\frac{W^{1/2}C_kW^{1/2}}{\sigma_{kj}}q_{kj}] - 2\omega_{kj}q_{kj}$$

where σ_{kj} is the j-th diagonal term of matrix Δ_k.
So,

$$q_{kj}^t\nabla_{q_{kj}}\mathcal{L} = 0 \Leftrightarrow \omega_{kj}q_{kj} = \frac{\eta_k}{\sigma_{kj}}q_{kj}^tW^{1/2}C_kW^{1/2}q_{kj}$$
$$\Leftrightarrow W^{1/2}C_kW^{1/2}q_{kj} = \frac{\omega_{kj}\sigma_{kj}}{\eta_k}q_{kj}.$$

q_{kj} is the eigenfunction of $W^{1/2}C_kW^{1/2}$ which match the eigenvalue $\lambda_{kj} = \frac{\omega_{kj}\sigma_{kj}}{\eta_k} = W^{1/2}C_kW^{1/2}$. We can write $q_{kj}^tq_{kl} = 0$ if $j \neq l$. So the log-likelihood can be written:

$$-2l(\theta) = \sum_{k=1}^{K}\eta_k[\sum_{j=1}^{d_k}(log(a_{kj}) + \frac{\lambda_{kj}}{a_{kj}}) + \sum_{j=d_k+1}^{R}(log(b_k) + \frac{\lambda_{kj}}{b_k})] + C^{te},$$

we substitute the equation $\sum_{j=d_k+1}^{R}\lambda_{kj} = tr(W^{1/2}C_kW^{1/2}) - \sum_{j=1}^{d_k}\lambda_{kj}$:

$$\begin{aligned}
-2l(\theta) &= \sum_{k=1}^{K}\eta_k[\sum_{j=1}^{d_k}log(a_{kj}) + \sum_{j=1}^{d_k}\frac{\lambda_{kj}}{a_{kj}} + \sum_{j=d_i+1}^{R}log(b_k) + \frac{1}{b_k}(tr(W^{1/2}C_kW^{1/2}) - \sum_{j=1}^{d_k}\lambda_{kj})] + C^{te} \\
&= \sum_{k=1}^{K}\eta_k[\sum_{j=1}^{d_k}log(a_{kj}) + \sum_{j=1}^{d_k}\lambda_{kj}(\frac{1}{a_{kj}} - \frac{1}{b_k}) + \sum_{j=d_i+1}^{R}log(b_k) + \frac{1}{b_k}tr(W^{1/2}C_kW^{1/2})] + C^{te} \\
&= \sum_{k=1}^{K}\eta_k[\sum_{j=1}^{d_k}log(a_{kj}) + \sum_{j=1}^{d_k}\lambda_{kj}(\frac{1}{a_{kj}} - \frac{1}{b_k}) + (p-d_k)log(b_k) + \frac{tr(W^{1/2}C_kW^{1/2})}{b_k}] + C^{te}.
\end{aligned}$$

In order to minimize $-2l(\theta)$ compared to q_{kj}, we minimize the quantity $\sum_{k=1}^{K}\eta_k\sum_{j=1}^{d_k}\lambda_{kj}(\frac{1}{a_{kj}} - \frac{1}{b_k})$ compared to λ_{kj}. Knowing that $(\frac{1}{a_{kj}} - \frac{1}{b_k}) \leq 0, \forall j = 1, ..., d_k$, λ_{kj} has to be as high as feasible. So, the j-th column q_{kj} of matrix Q is estimated by the eigenfunction associated to the j-th highest eigenvalue of $W^{1/2}C_kW^{1/2}$.

Parameter a_{kj} As a reminder $(ln(x))' = \frac{x'}{x}$ and $(\frac{1}{x})' = -\frac{1}{x^2}$. The partial derivative of $l(\theta)$ in relation to a_{kj} is:

$$-2\frac{\partial l(\theta)}{\partial a_{kj}} = \eta_k(\frac{1}{a_{kj}} - \frac{q_{kj}^tW^{1/2}C_kW^{1/2}q_{kj}}{a_{kj}^2})$$

The condition $\frac{\partial l(\theta)}{\partial a_{kj}} = 0$ is equivalent to:

$$\begin{aligned}
\eta_k(\frac{1}{a_{kj}} - \frac{q_{kj}^tW^{1/2}C_kW^{1/2}q_{kj}}{a_{kj}^2}) &= 0 \\
\Leftrightarrow \frac{1}{a_{kj}} &= \frac{q_{kj}^tW^{1/2}C_kW^{1/2}q_{kj}}{a_{kj}^2} \\
\Leftrightarrow a_{kj} &= q_{kj}^tW^{1/2}C_kW^{1/2}q_{kj} \\
\Leftrightarrow a_{kj} &= \lambda_{kj}
\end{aligned}$$

Parameter b_k The partial derivative of $l(\theta)$ in relation to b_k is:

$$\begin{aligned}-2\frac{\partial l(\theta)}{\partial b_k} &= \eta_k \sum_{j=d_k+1}^{R} (\frac{1}{b_k} - \frac{q_{kj}^t W^{1/2} C_k W^{1/2} q_{kj}}{b_k^2}) \\ &= \eta_k (\frac{R-d_k}{b_k} - \sum_{j=d_k+1}^{R} \frac{q_{kj}^t W^{1/2} C_k W^{1/2} q_{kj}}{b_k^2})\end{aligned}$$

But,

$$\sum_{j=d_k+1}^{R} q_j^t W^{1/2} C_k W^{1/2} q_j = tr(W^{1/2} C_k W^{1/2}) - \sum_{j=1}^{d_k} q_j^t W^{1/2} C_k W^{1/2} q_j,$$

so:

$$\begin{aligned}-2\frac{\partial l(\theta)}{\partial b_k} &= \eta_k \frac{(R-d_k)}{b_k} - \frac{\eta_k}{b_k^2}(tr(W^{1/2} C_k W^{1/2}) - \sum_{j=1}^{d_k} q_{kj}^t W^{1/2} C_k W^{1/2} q_{kj}) \\ &= \eta_k \frac{(R-d_k)}{b_k} - \frac{\eta_k}{b_k^2}(tr(W^{1/2} C_k W^{1/2}) - \sum_{j=1}^{d_k} \lambda_{kj})\end{aligned}$$

The condition $\frac{\partial l(\theta)}{\partial b_k} = 0$ is equivalent to:

$$\begin{aligned}&\eta_k \frac{(R-d_k)}{b_k} - \frac{\eta_k}{b_k^2}(tr(W^{1/2} C_k W^{1/2}) - \sum_{j=1}^{d_k} \lambda_{kj}) = 0 \\ \Leftrightarrow\ &\eta_k \frac{(R-d_k)}{b_k} = \frac{\eta_k}{b_k^2}(tr(W^{1/2} C_k W^{1/2}) - \sum_{j=1}^{d_k} \lambda_{kj}) \\ \Leftrightarrow\ &b_k = \frac{\eta_k}{\eta_k (R-d_k)}(tr(W^{1/2} C_k W^{1/2}) - \sum_{j=1}^{d_k} \lambda_{kj}) \\ \Leftrightarrow\ &b_k = \frac{1}{(R-d_k)}(tr(W^{1/2} C_k W^{1/2}) - \sum_{j=1}^{d_k} \lambda_{kj})\end{aligned}$$

References

Akaike H (1974) A new look at the statistical model identification. IEEE Transactions on Automatic Control 9:716–723

Berrendero J, Justel A, Svarc M (2011) Principal components for multivariate functional data. Computational Statistics and Data Analysis 55:2619–263

Biernacki C, Celeux G, Govaert G (2000) Assessing a mixture model for clustering with the integrated completed likelihood. IEEE Trans PAMI 22:719–725

Birge L, Massart P (2007) Minimal penalties for gaussian model selection. Probability theory and related fields 138:33–73

Bongiorno EG, Goia A (2016) Classification methods for hilbert data based on surrogate density. Comput Stat Data Anal 99(C):204–222, DOI 10.1016/j.csda.2016.01.019, URL `http://dx.doi.org/10.1016/j.csda.2016.01.019`

Bouveyron C, Jacques J (2011) Model-based clustering of time series in group-specific functional subspaces. Advances in Data Analysis and Classification 5(4):281–300

Bouveyron C, Come E, Jacques J (2015) The discriminative functional mixture model for the analysis of bike sharing systems. Annals of Applied Statistics 9(4):1726–1760

Cattell R (1966) The scree test for the number of factors. Multivariate Behaviour Research 1(2):245–276

Chen L, Jiang C (2016) Multi-dimensional functional principal component analysis. Statistics and Computing 27:1181–1192

Chiou J, Chen Y, Yang Y (2014) Multivariate functional principal component analysis: a normalization approach. Statistica Sinica 24:1571–1596

Chiou JM, Li PL (2007) Functional clustering and identifying substructures of longitudinal data. Journal of the Royal Statistical Society Series B Statistical Methodology 69(4):679–699

Dempster A, Laird N, Rubin D (1977) Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society 39(1):1–38

Dias R, Collazos J, Zambom A (2018) Functional data clustering via hypothesis testing k-means. Computational Statistics DOI 10.1007/s00180-018-0808-9

Ferraty F, Vieu P (2003) Curves discrimination: a nonparametric approach. Computational Statistics and Data Analysis 44:161–173

Happ C, Greven S (2015) Multivariate functional principal component analysis for data observed on different (dimensional) domains. Journal of the American Statistical Association p in press

Ieva F, Paganoni A (2013) Risk prediction for myocardial infarction via generalized functional regression models. Statistical methods in medical research 25, DOI 10.1177/0962280213495988

Ieva F, Paganoni A, Pigoli D, Vitelli V (2013) Multivariate Functional Clustering for the Morphological Analysis of ECG Curves. Journal of the Royal Statistical Society, Series C (Applied Statistics) 62(3):401–418

Jacques J, Preda C (2013) Funclust: a curves clustering method using functional random variable density approximation. Neurocomputing 112:164–171

Jacques J, Preda C (2014a) Functional data clustering: a survey. Advances in Data Analysis and Classification 8(3):231–255

Jacques J, Preda C (2014b) Model based clustering for multivariate functional data. Computational Statistics and Data Analysis 71:92–106

James G, Sugar C (2003) Clustering for sparsely sampled functional data. Journal of the American Statistical Association 98(462):397–408

Kayano M, Dozono K, Konishi S (2010) Functional Cluster Analysis via Orthonormalized Gaussian Basis Expansions and Its Application. Journal of Classification 27:211–230

Petersen KB, Pedersen MS (2012) The matrix cookbook. URL `http://www2.imm.dtu.dk/pubdb/p.php?3274`, version 20121115

Preda C (2007) Regression models for functional data by reproducing kernel hilbert spaces methods. Journal of Statistical Planning and Inference 137:829–840

R Core Team (2017) R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria, URL `https://www.R-project.org/`

Ramsay JO, Silverman BW (2005) Functional data analysis, 2nd edn. Springer Series in Statistics, Springer, New York

Rand WM (1971) Objective criteria for the evaluation of clustering methods. Journal of the American Statistical Association 66(336):846–850

Saporta G (1981) Méthodes exploratoires d'analyse de données temporelles. Cahiers du Buro 37–38

Schwarz G (1978) Estimating the dimension of a model. The Annals of Statistics 6(2):461–464

Singhal A, Seborg D (2005) Clustering multivariate time-series data. Journal of Chemometrics 19:427–438

Tarpey T, Kinateder K (2003) Clustering functional data. Journal of Classification 20(1):93–114

Tokushige S, Yadohisa H, Inada K (2007) Crisp and fuzzy k-means clustering algorithms for multivariate functional data. Computational Statistics 22:1–16

Traore OI, Cristini P, Favretto-Cristini N, Pantera L, Vieu P, Viguier-Pla S (2019) Clustering acoustic emission signals by mixing two stages dimension reduction and nonparametric approaches. Computational Statistics DOI 10.1007/s00180-018-00864-w

Yamamoto M (2012) Clustering of Functional Data in a Low-Dimensional Subspace. Advances in Data Analysis and Classification 6:219–247

Yamamoto M, Hwang H (2017) Dimension-Reduced Clustering of Functional Data via Subspace Separation. Journal of Classification 34:294–326

Yamamoto M, Terada Y (2014) Functional Factorial k-Means Analysis. Computational Statistics and Data Analysis 79:133–148

2 Comportement de l'algorithme sur un gros jeu de données avec plusieurs variables

Le but de ces simulations est de tester le comportement de notre algorithme dans le cas de jeux de données multivariés présentant beaucoup de groupes. En effet, nous voulions nous assurer que notre algorithme n'avait pas tendance à vider les classes lorsque celles-ci étaient nombreuses.

2.1 Schéma de simulation

Des données sont simulées selon plusieurs scénarios différents :
Scénario 1 Le nombre de groupes est fixé à $K = 6$. Un échantillon de 2000 courbes est simulé selon le modèle suivant pour $t \in [1, 21]$:

$$
\begin{aligned}
\text{Groupe 1 :} \quad & X_1(t) = U + (1-U)h_1(t) + \epsilon(t), \\
& X_2(t) = U + (1-U)h_1(t) + \epsilon(t), \\
& X_3(t) = U \times t, \\
& X_4(t) = U \times t, \\
\text{Groupe 2 :} \quad & X_1(t) = U + (1-U)h_2(t) + \epsilon(t), \\
& X_2(t) = U + (0.5-U)h_2(t) + \epsilon(t), \\
& X_3(t) = U \times t, \\
& X_4(t) = U \times t, \\
\text{Groupe 3 :} \quad & X_1(t) = U + (1-U)h_1(t) + \epsilon(t), \\
& X_2(t) = U + (1-U)h_2(t) + \epsilon(t), \\
& X_3(t) = U \times t, \\
& X_4(t) = U \times t \\
\text{Groupe 4 :} \quad & X_1(t) = U + (0.5-U)h_2(t) + \epsilon(t), \\
& X_2(t) = U + (1-U)h_1(t) + \epsilon(t), \\
& X_3(t) = U \times t, \\
& X_4(t) = U \times t,
\end{aligned}
$$

$$
\begin{aligned}
\text{Groupe 5} \quad : \quad & X_1(t) = U + (1-U)h_2(t) + \epsilon(t), \\
& X_2(t) = U + (1-U)h_1(t) + \epsilon(t), \\
& X_3(t) = cos(t/3) + \epsilon(t), \\
& X_4(t) = U \times t, \\
\text{Groupe 6} \quad : \quad & X_1(t) = U + (1-U)h_2(t) + \epsilon(t), \\
& X_2(t) = U + (1-U)h_1(t) + \epsilon(t), \\
& X_3(t) = U \times t, \\
& X_4(t) = -cos(t/3) + \epsilon(t),
\end{aligned}
$$

où $U \sim \mathcal{U}(0, 0.1)$ et $\epsilon(t)$ est un bruit blanc indépendant de U et tel que $\mathbb{V}ar(\epsilon(t)) = 0.25$. Les fonctions h_1 et h_2 sont définies, pour $t \in [1, 21]$, par $h_1(t) = (6 - |t-7|)_+$ et $h_2(t) = (6 - |t-15|)_+$ où $(\cdot)_+$ correspond à la partie positive. Les proportions de mélange sont égales et les courbes sont observées sur 101 points équidistants. La forme fonctionnelle des données est reconstruite à l'aide d'une base Bspline cubique ou une base de Fourier composée de 25 bases de fonctions. Comme observé en Figure 4.1, les 6 groupes ne peuvent être distingués à partir d'une seule variable.

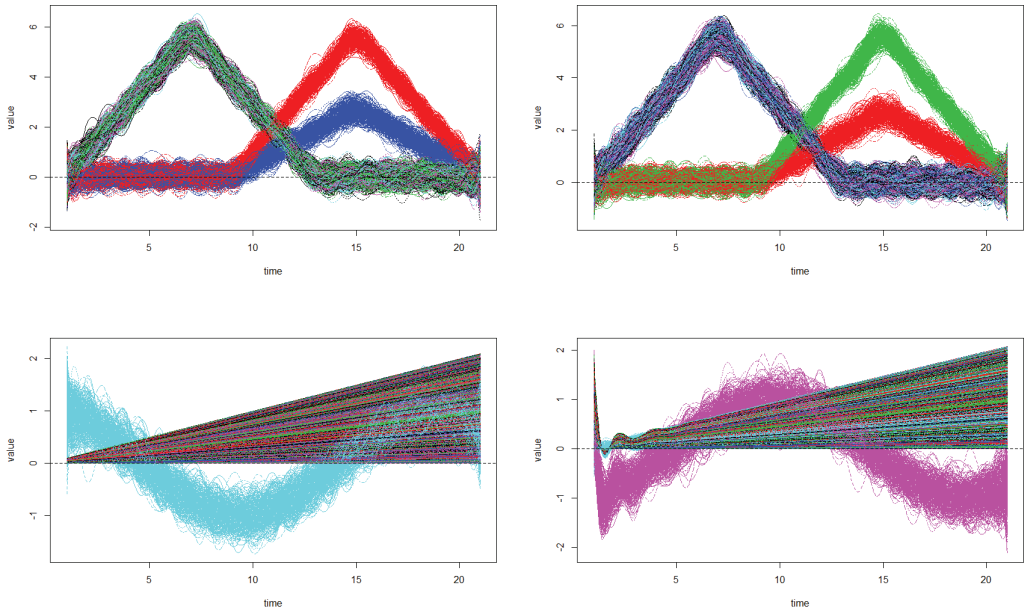


Figure 4.1: Données simulées pour les quatre variables (numérotées de haut en bas) colorées par groupe pour une simulation

Scénario 2 Le nombre de groupes est fixé à $K = 6$. Un échantillon de 2000 courbes est simulé selon le modèle suivant pour $t \in [1, 21]$:

$$
\begin{aligned}
\text{Groupe 1} \quad : \quad & X_1(t) = U + (1-U)h_1(t) + \epsilon(t), \\
& X_2(t) = U + (1-U)h_1(t) + \epsilon(t), \\
& X_3(t) = U \times t, \\
& X_4(t) = 2 + \epsilon(t), \\
\text{Groupe 2} \quad : \quad & X_1(t) = U + (1-U)h_2(t) + \epsilon(t), \\
& X_2(t) = U + (0.5-U)h_2(t) + \epsilon(t), \\
& X_3(t) = U \times t, \\
& X_4(t) = U \times t, \\
\text{Groupe 3} \quad : \quad & X_1(t) = U + (1-U)h_1(t) + \epsilon(t), \\
& X_2(t) = U + (1-U)h_2(t) + \epsilon(t), \\
& X_3(t) = U \times t, \\
& X_4(t) = U \times t, \\
\text{Groupe 4} \quad : \quad & X_1(t) = U + (0.5-U)h_2(t) + \epsilon(t), \\
& X_2(t) = U + (1-U)h_1(t) + \epsilon(t), \\
& X_3(t) = U \times t, \\
& X_4(t) = U \times t, \\
\text{Groupe 5} \quad : \quad & X_1(t) = U + (1-U)h_2(t) + \epsilon(t), \\
& X_2(t) = U + (1-U)h_1(t) + \epsilon(t), \\
& X_3(t) = cos(t/3) + \epsilon(t), \\
& X_4(t) = U \times t, \\
\text{Groupe 6} \quad : \quad & X_1(t) = U + (1-U)h_2(t) + \epsilon(t), \\
& X_2(t) = U + (1-U)h_1(t) + \epsilon(t), \\
& X_3(t) = U \times t, \\
& X_4(t) = -cos(t/3) + \epsilon(t),
\end{aligned}
$$

où $U \sim \mathcal{U}(0, 0.1)$ et $\epsilon(t)$ est un bruit blanc indépendant de U et tel que $\mathbb{V}ar(\epsilon(t)) = 0.25$. Les fonctions h_1 et h_2 sont définies, pour $t \in [1, 21]$, par $h_1(t) = (6 - |t-7|)_+$ et $h_2(t) = (6 - |t-15|)_+$ où $(\cdot)_+$ correspond à la partie positive. Les proportions de mélange sont égales et les courbes sont observées sur 101 points équidistants. La forme fonctionnelle des données est reconstruite à l'aide d'une base Bspline cubique ou une base de Fourier composée de 25 bases de fonctions (cf. Figure 4.2). Ce second scénario simplifie la détection du premier groupe (noir).

Pour chaque scénario de simulation, les partitions estimées sont comparées aux vraies partitions avec l'*Adjusted Rand Index* calculé avec la fonction *adjustedRandIndex* du package mclust.

Pour toutes les simulations, les options choisies pour l'algorithme sont : modèle $[a_{kj} b_k Q_k D_k]$, une initialisation aléatoire, 0.2 pour le seuil du scree-test de Cattel permettant le choix des dimensions intrinsèques d_k, l'initialisation de l'algorithme est faite de façon aléatoire, le critère d'arrêt de l'algorithme EM est une croissance de la log-vraissemblance inférieure à 10^{-3} ou un nombre maximal d'itérations de 200.

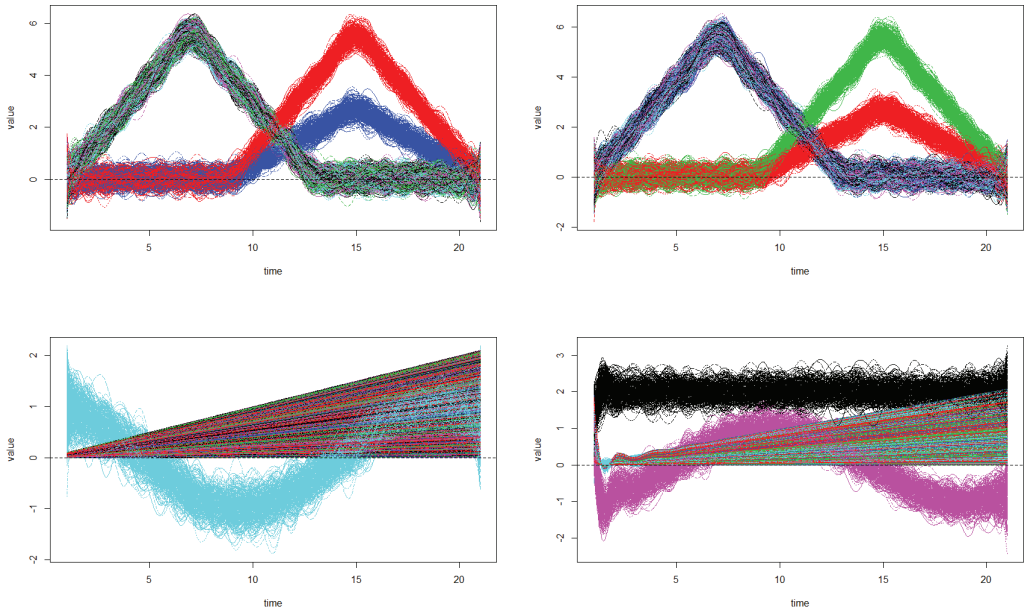


Figure 4.2: Données simulées pour les quatre variables (numérotées de haut en bas) colorées par groupe pour une simulation

2.2 Résultats

Les données sont générées d'après le *Scénario 1.* 10 simulations ont été réalisées avec l'algorithme appliqué pour $K = 6$ groupes. L'algorithme est lancé 20 fois, s'il n'y a pas de convergence vers une solution de partition avec 6 groupes après ces 20 lancés, le même schéma est répété pour $K = 5$ puis $K = 4$ si nécessaire.

Les résultats sont résumés en Table 4.1. On observe que l'on retrouve les 6 classes pour 100% des simulations dans le cas d'un lissage Fourier et 90% des simulations dans le cas d'un lissage Bsplines. De plus, quelles que soient les bases choisies pour le lissage, on obtient le même ARI moyen, mais l'écart type est moindre dans le cas des bases B-splines. On peut tout de même noter, d'un point de vue pratique, que la convergence de l'algorithme est plus difficile dans le cas des bases B-splines que dans le cas des bases de Fourier. Suite à ces résultats, nous avons choisi de tester notre algorithme sur un second schéma de simulation plus simple.

Lissage	Moyenne(sd)	K=6	K=5	K=4
Bsplines	0.78 (0.18)	9	1	-
Fourier	0.78(0.24)	10	-	-

Table 4.1: Moyenne (et sd) d'ARI pour 10 simulations avec une initialisation aléatoire et nombre de fois où l'on a eu une solution pour $K = 6$

Les données sont générées d'après le *Scénario 2.* 10 simulations ont été réalisées avec l'algorithme appliqué pour $K = 6$ groupes. Si, au bout de 20 initialisations, le modèle ne converge pas vers une solution de partition avec 6 groupes, le même schéma est répété pour $K = 5$. Les résultats sont résumés en Table 4.2.

Lissage	Mean(sd)	K=6	K=5
Bsplines	0.86 (0.13)	10	-
Fourier	0.93(0.17)	10	-

Table 4.2: Moyenne (et sd) d'ARI pour 10 simulations avec une initialisation aléatoire et nombre de fois où l'on a eu une solution pour $K = 6$

Avec ce schéma de simulation simplifié, on observe que l'algorithme retrouve, pour 100% des simulations, une partition à 6 groupes avec un ARI moyen supérieur à 0.8 quelque soit le lissage utilisé pour les données.

Pour conclure, notre algorithme n'a pas tendance a vider les classes lorsque celles ci sont nombreuses. Les meilleurs résultats sont obtenus dans le cas d'un lissage Fourier lorsque le schéma de simulation est simple. Lorsque les classes sont plus mélangées, les meilleurs résultats de classification sont obtenus avec le lissage Bspline. Par contre, l'algorithme a tendance à mettre plus de temps à converger dans le cas d'un lissage avec des bases Bsplines.

Chapitre 5

Application sur données réelles : estimation de la vitesse et de la longueur de foulée

Nous allons maintenant comparer l'utilisation de notre méthode de clustering avant l' application d'un modèle prédictif sur chaque cluster et les différentes méthodes de régression présentées au chapitre 3 pour l'estimation de la vitesse et de la longueur de foulée. Ces méthodes seront comparées à l'aide du pourcentage d'erreur de prédiction supérieur à 0.6 m/s pour la vitesse et supérieur à 30 cm pour la longueur de foulée.

1 Vitesse

Afin de comparer les performances des méthodes de régression présentées précédamment la base de données de 3221 foulées et 6 signaux (101 points par signal) est découpée en deux parties : une base d'entrainement composée d'un tirage aléatoire sans remise de 80% de la base de données et les 20% restants forment la base de test. Ce processus est répété 50 fois pour éviter les fluctuations de résultats liées au tirage aléatoire. Chaque modèle sera construit sur la base d'entrainement puis les valeurs moyennes, minimales et maximales de pourcentage d'erreur supérieur à 0.6 m/s seront calculées sur la base de test.

Dans un premier temps chaque méthode sera appliquée sur les données brutes. Puis dans un second temps, ces méthodes sont appliquées sur les données brutes des sous-groupes obtenus suite à l'application de notre modèle de clustering fonctionnel multivarié (funHDDC). En effet, nous avons voulu tester si la division de la base générale en sous-groupes plus homogènes d'un point de vue des signaux collectés, avant l'application d'un modèle de prédiction, permettrait d'affiner les résultats de prédiction.

Il a donc fallu se demander quelle variable fonctionnelle inclure dans le modèle de clustering et quelle était la meilleure partition pour les données. Dans le cas de notre étude, le critère de décision utilisé est la minimisation du pourcentage d'erreur de prédiction inférieur à 0.6 m/s aussi bien pour le nombre de variables à inclure que pour la sélection du nombre de classes. Pour cette dernière, nous aurions aussi pu utiliser un critère de sélection de modèle classique comme le BIC ou l'AIC mais qui est totalement décorrélé de notre objectif de prédiction final. Pour chaque modèle, nous avons donc testé différentes partitions (en 2 et 3 groupes, car nous n'avions pas de convergence de funHDDC sur notre base d'entrainement pour un plus grand nombre de groupes) et l'inclusion d'une

ou plusieurs variables fonctionnelles. Les résultats de toutes ces comparaisons ne sont pas présentés ici, les meilleurs résultats de prédiction sont obtenus pour un nombre de groupes K égal à 2 et l'inclusion des variables d'accélération Az et de vitesse angulaire Gy pour le clustering.

Les résultats de prédiction sont rassemblés dans le Tableau 5.1. On observe que pour les méthodes de régression en grande dimension, la division de la base de données en deux sous-groupes plus homogènes permet de réduire le pourcentage d'erreur tandis que pour les approches de type machine learning, cette subdivision n'impacte pas fortement le pourcentage d'erreur. D'un point de vue performance, les meilleurs résultats sont obtenus avec les SVM, suivi de près par la régression fonctionnelle non paramétrique. Pour ces deux méthodes, c'est l'application du modèle sur les données brutes qui conduit aux meilleurs résultats.

Ceci peut s'expliquer par le fait que notre base de données est constituée de 3221 foulées issues de 58 chevaux, ce qui est relativement peu. Il est possible que notre méthode de clustering multivarié ait un apport plus notable lorsque la base de données sera étoffée avec plus de chevaux, augmentant ainsi la variabilité au sein des données.

Le modèle SVM a donc été implémenté sur un serveur de calcul afin de pouvoir communiquer avec l'application téléphone de l'entreprise et fournir une vitesse par foulée précise à 0.6 m/s avec un taux d'erreur moyen inférieur à 10% au cavalier.

Méthode	Base de données	Moyenne	Min	Max
Ridge	Base de données brutes	27.7	24.2	30.7
Ridge	2 sous-groupes obtenus par funHDDC	24.0	21.2	27.6
Lasso	Base de données brutes	29.1	25.9	31.8
Lasso	2 sous-groupes obtenus par funHDDC	25.0	22.0	28.1
PCR	Base de données brutes	28.5	25.5	31.1
PCR	2 sous-groupes obtenus par funHDDC	26.4	21.2	34.8
PLS	Base de données brutes	28.8	25.0	32.2
PLS	2 sous-groupes obtenus par funHDDC	26.4	21.0	33.8
Elastic net $\alpha = 0.3$	Base de données brutes	27.9	24.6	30.9
Elastic net $\alpha = 0.3$	2 sous-groupes obtenus par funHDDC	24.1	21.0	27.7
Régression multiple	Base de données brutes	34.0	30.4	37.8
Régression multiple	2 sous-groupes obtenus par funHDDC	41.0	36.9	44.9
Régression fonctionnelle paramétrique	Données brutes	26.9	23.6	31.7
Régression fonctionnelle paramétrique	2 sous-groupes obtenus par funHDDC	26.4	23.1	29.4
Régression fonctionnelle non paramétrique	Base de données brutes	16.7	13.0	20.1
Régression fonctionnelle non paramétrique	2 sous-groupes obtenus par funHDDC	17.2	14.9	20.3
SVM	Base de données brutes	**9.9**	6.7	11.9
SVM	2 sous-groupes obtenus par funHDDC	10.7	7.8	12.8
Réseau de neurones	Base de données brutes	29.9	25.7	33.7
Réseau de neurones	2 sous-groupes obtenus par funHDDC	28.7	24.8	34.8
Forêt aléatoire	Base de données brutes	17.5	14.3	20.7
Forêt aléatoire	2 sous-groupes obtenus par funHDDC	18.3	14.7	20.7

Table 5.1: Moyenne, valeur minimale et maximale de pourcentage d'erreur supérieur à 0.6 m/s pour l'estimation de la vitesse

2 Longueur de foulée

La même logique que pour la vitesse a été appliquée ici. Le seuil de comparaison du pourcentage d'erreur est fixé à 0.3 m. Les meilleurs résultats sont obtenus dans le cas où on tient compte de la vitesse prédite dans le modèle, sans pour autant arriver à passer en dessous de la barre des 10% d'erreur (cf. Tables 5.2 et 5.3).

Méthode	Base de données	Moyenne	Min	Max
Ridge	Base de données brutes	36.5	33.1	39.3
Ridge	Base de données brutes+Vitesse prédite	20.4	15.6	22.9
Ridge	2 sous-groupes obtenus par funHDDC	33.4	30.5	36.3
Ridge	2 sous-groupes obtenus par funHDDC + Vitesse prédite	19.8	17.3	23.3
Lasso	Base de données brutes	38.7	34.6	42.8
Lasso	Base de données brutes+ vitesse prédite	19.2	15.1	23.3
Lasso	2 sous-groupes obtenus par funHDDC	34.0	30.0	37.6
PCR	Base de données brutes	37.1	33.5	41.9
PCR	Base de données brutes+vitesse prédite	19.2	16.2	22.9
PCR	2 sous-groupes obtenus par funHDDC	35.2	29.9	43.7
PLS	Base de données brutes	37.5	34.3	40.6
PLS	Base de données brutes+vitesse prédite	19.1	15.8	21.8
PLS	2 sous-groupes obtenus par funHDDC	35.2	29.1	42.8
Elastic net $\alpha = 0.3$	Base de données brutes	36.5	33.0	39.5
Elastic net $\alpha = 0.3$	Base de données brutes+vitesse prédite	18.3	15.5	22.2
Elastic net $\alpha = 0.3$	2 sous-groupes obtenus par funHDDC	33.4	29.2	36.7
Régression multiple	Base de données brutes	42.7	38.4	48.0
Régression multiple	Base de données brutes+vitesse prédite	21.3	17.3	25.3
Régression multiple	2 sous-groupes obtenus par funHDDC	49.6	46.9	54.9
Régression fonctionnelle paramétrique	Données brutes	36.3	32.2	40.8
Régression fonctionnelle paramétrique	Données brutes+vitesse prédite	-	-	-
Régression fonctionnelle paramétrique	2 sous-groupes obtenus par funHDDC	35.0	30.2	39.9
Régression fonctionnelle non paramétrique	Base de données brutes	24.2	20.1	27.0
Régression fonctionnelle non paramétrique	Base de données brutes+vitesse prédite	-	-	-
Régression fonctionnelle non paramétrique	2 sous-groupes obtenus par funHDDC	24.7	20.5	29.4

Table 5.2: Moyenne, valeur minimale et maximale de pourcentage d'erreur supérieur à 0.3 m pour l'estimation de la longueur de foulée

Pour conclure, afin de répondre aux objectifs de prédiction de la vitesse du cheval et de sa longueur de foulée, nous avons développé un modèle de clustering fonctionnel multivarié. Ce modèle a fait ses preuves et s'est démarqué par rapport à ses concurrents sur données simulées. En revanche, il n'a pas eu l'effet escompté sur les modèles de prédiction : le découpage en sous-groupes homogènes avant l'application d'un modèle de prédiction par sous-groupe ne permet pas de réduire drastiquement l'erreur de prédiction supérieure à 0.6 m/s pour la vitesse et 0.3 m pour la longueur de foulée. A ce jour, avec la base de données d'entrainement actuelle composée de 3221 foulées issues de 58 chevaux, l'application d'un modèle de prédiction global est plus performant et plus précisément d'un modèle SVM. Ces résultats devront être revus suite à de nouvelles campagnes de mesures.

Méthode	Base de données	Moyenne	Min	Max
SVM	Base de données brutes	17.8	14.3	21.0
SVM	Base de données brutes+vitesse prédite	**17.5**	12.5	20.9
SVM	2 sous-groupes obtenus par funHDDC	18.8	14.7	22.0
SVM	2 sous-groupes obtenus par funHDDC+Vitesse	18.2	13.8	21.2
Réseau de neurones	Base de données brutes	40.7	34.8	50.5
Réseau de neurones	Base de données brutes+vitesse prédite	62.6	53.8	77.5
Réseau de neurones	2 sous-groupes obtenus par funHDDC	38.4	33.8	42.6
Forêt aléatoire	Base de données brutes	25.5	22.0	30.5
Forêt aléatoire	Base de données brutes+vitesse prédite	18.5	15.3	21.4
Forêt aléatoire	2 sous-groupes obtenus par funHDDC	25.9	22.0	29.8

Table 5.3: Moyenne, valeur minimale et maximale de pourcentage d'erreur supérieur à 0.3 m pour l'estimation de la longueur de foulée

Chapitre 6

Extension vers le co-clustering fonctionnel multivarié

Dans le cadre de notre travail, six variables sont collectées simultanément au cours du temps. Avec la fréquence d'échantillonage de 100 Hz, cela représente rapidement un volume conséquent de données à stocker et analyser. Parmi nos axes de recherche, nous sommes intéressés par suivre les même chevaux sur plusieurs années pour étudier le lien entre leurs performances et leur locomotion au cours du temps. Dans ce cas, plutôt que d'analyser des profils de courbe au cours de longues périodes, il peut être intéressant de réduire la fenêtre de temps à un parcours ou un entrainement et de regarder si l'on distingue des groupes de foulées et de parcours qui se ressemblent. Cette analyse pourrait permettre de mettre en évidence des épreuves pour lesquelles la locomotion du cheval était identique ou encore détecter des inconforts sur certains types d'épreuves ou de sol.

Ainsi, il faudrait constituer une nouvelle base de données avec en ligne les foulées et en colonne les parcours réalisés par un même cheval avec à l'intersection d'une ligne et d'une colonne les six variables mesurées par l'IMU (cf. Figure 6.1). Pour analyser une telle table, nous proposons de faire un clustering simultané des lignes et des colonnes de façon à obtenir des groupes de foulées homogènes et des groupes d'épreuves homogènes. Ce type de clustering est appelé co-clustering [Govaert and Nadif, 2013]. Le co-clustering résulte en l'obtention de blocs de foulées et d'épreuves homogènes du point de vue des variables mesurées (les signaux de l'IMU).

La méthode de co-clustering que nous avons développé est présentée ci-dessous, sous la forme de la publication qui sera soumise prochainement. Puis nous présenterons son utilisation sur des données réelles de locomotion du cheval.

1 Article

Detection of energy waste in French households thanks to a co-clustering model for multivariate functional data

Amandine Schmutz*
Lim France, Chemin Fontaine de Fanny, Nontron, France
Julien Jacques,
Université de Lyon, Lyon 2, ERIC EA3083, Lyon, France
Charles Bouveyron,
Université Côte d'Azur, Inria, CNRS
LJAD, Maasai team, France
Laurent Bozzi,
EDF R&D, Paris-Saclay,France
Laurence Chèze,
Université de Lyon, Lyon 1, LBMC UMR T9406, Lyon, France
Pauline Martin,
Lim France, Chemin Fontaine de Fanny, Nontron, France,
CWD-VetLab, Ecole Nationale Vétérinaire d'Alfort,
Maisons-Alfort, F-94700, France

October 21, 2019

*The authors would like to thank the LabCom 'CWD-VetLab' for its financial support. The LabCom 'CWD-VetLab' is financially supported by the Agence Nationale de la Recherche (contract ANR 16-LCV2-0002-01). This research has also benefited from the support of the "FMJH Research Initiative Data Science for Industry".

Abstract

The exponential growth of smart devices in all aspects of everyday life leads to make common the collection of high frequency data. Those data can be seen as multivariate functional data: quantitative entities evolving along time, for which there is a growing needs of methods to summarize and understand them. The database that have motivated our project is supplied by the historical French electricity provider whose aim is to detect poorly insulated buildings, anomalies or long periods of absence. Their motivation is to answer COP24 requirements to reduce energy waste and to adapt electric load. To this end, a novel co-clustering model for multivariate functional data is defined. The model is based on a functional latent block model which assumes for each block a probabilistic distribution for multivariate functional principal component scores. A Stochastic EM algorithm, embedding a Gibbs sampler is proposed for model inference, as well as model selection criteria for choosing the number of co-clusters.

Keywords: latent block model, multivariate functional PCA, SEM-Gibbs algorithm, electricity consumption.

1 Introduction

Sensor networks are undergoing a great expansion in several domains such as industry 4.0, environment, transport, defense, smart cities, etc. The emergence of small size sensors at a reduced cost leads to an increasing use of connected devices for the mainstream. These sensors are put into smart houses to simultaneously follow, at a high frequency, the temperature of different rooms and the outdoor temperature for example. Those data can also be correlated to the electric consumption of the household. Studying electric consumption allows cities to have a better network management, users to reduce their energetic cost, and electric providers to adapt a strategy to meet demand. It is also an opportunity to gather customers consumption data and therefore improve customer knowledge. EDF, the historical and main French electricity provider, plans to access smart meters data every half an hour, which means 17472 measures per year for each of its 27 million clients. In the near future, many other information will be collected simultaneously by EDF thanks to IoT devices on all production facilities (nuclear power stations, hydrolic centrals, windmills, ...). Indeed, having the opportunity to cluster both wind production and electric consumption would allow a better steering of the energy use in order to use at the right time what is produced by the windmill for example. One of the major challenge of those devices is that they represent a mass of data to store and analyze. Thus, it may be necessary to build summaries allowing an easier storage and analysis. One way to achieve that is to cluster those data into homogeneous groups. In this work, we focus on the analysis of three features observed on a set of households: power consumption, indoor temperature and outdoor temperature. Of course, the proposed model can be extended to any set of multivariate temporal series.

There is an abundant literature on understanding electricity consumption patterns.

Simplest methods are based on averaging electric consumption per day and then applying a hierarchical clustering on those averaged values (Gouveia et al., 2015). Other works rely on studying temporal series monitored for a month with fuzzy methods such as fuzzy c-means algorithm (Zhou et al., 2017), but those methods neglect the time dependency inherent in such data. Thus, extensions of k-means methods are proposed to take into account the temporal component through preprocessing steps before applying the k-means algorithm (Tureczek et al., 2018) or with a dynamic k-means clustering algorithm where the similarity distances are calculated taking into account all Euclidean distances between each pair of objects at the same time stamp (Benitez et al., 2014). The same authors also proposed a dynamic clustering algorithm where two objects are compared thanks to a final distance which is the average of all comparisons of objects features (Benitez et al., 2016). Contrary to the previous methods, the dynamic of the data is taken into account by decomposing the temporal series into smaller linear surfaces which are compared by applying a Hausdorff-based similarity distance. Lastly, Bouveyron et al. (2017) study households electric consumption monitored for two years and proposed an algorithm that allows to cluster simultaneously households and days of measurements that have the same pattern. This method is based on both functional data analysis and co-clustering techniques that will be developed later in this paper. However, all of these works share one restrictive characteristic: they are limited to the study of only one functional variable.

In the present work, we want to follow three variables: electricity consumption, indoor and outdoor temperature. Such data can be seen as multivariate functional data: multiple quantitative entities evolving along time collected simultaneously for a same individual (Ramsay and Silverman, 2005; Bouveyron et al., 2019). In order to analyze and understand such data, it may be interesting to identify subgroups of individuals (households here) that

have the same profile for these three quantities. For example, to understand the profile of consumption depending on both indoor and outdoor temperatures in order to adapt the electric production or detect poorly insulated buildings. However, analyzing these profiles over a long period of time is in practice complex because it represents a large amount of data. The window of time that practitioners are used to analyze is the day (24 hours). Consequently the observed time period is cut into daily observations. The data set under study is consequently a large table, where rows correspond to households and columns to days, and in which each elements is a set of three curves: electricity consumption, indoor and outdoor temperatures.

To analyze such table, we propose to simultaneously cluster the rows into homogeneous groups of households and the columns into homogeneous groups of days. Such kind of analysis is called co-clustering (Govaert and Nadif, 2013). The co-clustering will result in exhibiting homogeneous blocks of households and days having similar behaviour according to the three functional variables under study.

One of the most famous model for co-clustering is the Latent Block Model (LBM, (Govaert and Nadif, 2013)). According to the LBM, the elements of a block are modeled by a parametric distribution. Each block is therefore interpretable thanks to the block-distribution's parameters. Moreover, model selection criterion, such as the ICL criterion (Biernacki et al., 2000), can be used for model selection purpose, including the choice of the number of co-clusters. This technique proved its efficiency for the co-clustering of numerous types of data: continuous (Nadif and Govaert, 2008), nominal (Bhatia et al., 2014), binary (Laclau et al., 2017), ordinal (Jacques and Biernacki, 2018; Corneli et al., 2019), functional data (Bouveyron et al., 2017; Chamroukhi and Biernacki, 2017; Slimen et al., 2018) or even mixed-type data (Selosse et al., 2019).

Slimen et al. (2018) proposed a co-clustering algorithm based on a vectorial LBM applied on the functional principal components scores of the curves. Bouveyron et al. (2017) extended this work by proposing a functional latent block model assuming that the functional principal components of the curves are block-specific and live into a low-dimensional subspace. Chamroukhi and Biernacki (2017) presented another co-clustering model based on a latent block model where the probability density function is estimated thanks to a regression model with a hidden logistic process. In the present work, a co-clustering algorithm for multivariate functional data is proposed as an extension of Bouveyron et al. (2017) to the multivariate case. Moreover, a more flexible probabilistic model is presented.

The paper is organized as follows. Section 2 presents the co-clustering model and Section 3 its inference. Results of the algorithm on simulated data is presented in Section 4. Section 5 is dedicated to the application on electricity consumption and indoor and outdoor temperatures. Our algorithm succeed in detecting poor insulated buildings and periods of low consumption. Then a discussion concludes the paper in Section 6.

2 A co-clustering model for multivariate functional data

Functional data, which are the observations of a random variable living into a infinite dimensional space, are in practice observed only at a finite set of time points. Consequently, a first step in functional data analysis is the reconstruction of the functional nature of data. This aspect is discussed in the second part of this section, just after having introduced the notations. Then, the proposed co-clustering model for multivariate functional data is presented.

2.1 Data and notations

Let $\boldsymbol{x} = (\boldsymbol{x}_{ij})_{1\leq i\leq n, 1\leq j\leq p}$ be the data table of dimension $n \times p$, where each element $\boldsymbol{x}_{ij}$ is a multivariate curve $\boldsymbol{x}_{ij} = (x_{ij}^1(t), \ldots, x_{ij}^S(t))$ with $t \in [0, T]$. Let us recall that i is the row index, j is the column index and s corresponds to the component of the multivariate curves. In our application i refers to the household, j to the day and s either to electricity consumption, indoor or outdoor temperature. Nevertheless, the model and its inference presented in this work can be used for any other matrix of multivariate functional data.

2.2 Functional data reconstruction

In practice, the functional expressions of the curves $x_{ij}^s(t)$ are not known and we only have access to discrete observations at a finite set of times: $x_{ij}^s(t_1), x_{ij}^s(t_2), \ldots$. A common way to reconstruct the functional form is to assume that the observations can be decomposed into a finite dimensional space spanned by a basis of functions. So each observed curve x_{ij}^s $(1 \leq i \leq n, 1 \leq j \leq p,\ 1 \leq s \leq S)$ can be expressed as a linear combination of basis functions $\{\phi_r^s\}_{r=1,\ldots,R_s}$:

$$x_{ij}^s(t) = \sum_{r=1}^{R_s} c_{ijr}^s \phi_r^s(t)$$

with R_s the number of basis functions. These basis functions can be for instance Fourier bases, spline bases, etc. The automatic choice of this basis (as well as the number of basis functions) is an open problem (Jacques and Preda, 2014a). In practice, this choice is done empirically such that the reconstruction is judged reasonable by the expert. Estimation of the basis expansion coefficients c_{ijr}^s is classically done by least squares smoothing. We refer the reader to Ramsay and Silverman (2005) for a complete survey on this aspect. Let $c = (c_{ij})_{ij}$ be the whole data set of coefficients, where $c_{ij} = (c_{ij1}^1, \ldots, c_{ijR_1}^1, \ldots, c_{ij1}^S, \ldots, c_{ijR_S}^S)$

contains the coefficients for individual i at day j which corresponds to the concatenation of coefficients c^s_{ijr} for all S functional variables. Let $c_i = (c_{ij})_j$ be the coefficients for the ith individual and similarly $c_j = (c_{ij})_i$ the coefficients for day j.

For clarity of the presentation, the same number of basis functions as well as the same basis function $\{\phi_r\}_{r=1,\dots,R}$ is considered for each component of the multivariate functional variable. But extension is straightforward.

2.3 The proposed latent block model

The aim of a co-clustering model is to define row and column partitions in order to summarize the data matrix $\boldsymbol{x}$ into smaller subgroups, usually called blocks. Let $z = (z_{ik})_{1\leq i\leq n, 1\leq k\leq K}$ be the row partition of the n rows into K groups, and $w = (w_{jl})_{1\leq j\leq p, 1\leq l\leq L}$ the column partition of the p columns into L groups, such as $z_{ik} = 1$ if row i belongs to row-cluster k and 0 otherwise (and similarly for w_{jl}). Thus, one block is defined by a set of curves which belong to a row and column cluster such as $z_{ik}w_{jl} = 1$.

Let us first assume that z and w partitions are independent:

$$p(c;\theta) = \sum_{z\in Z}\sum_{w\in W} p(z;\theta)p(w;\theta)p(c|z,w;\theta) \tag{1}$$

where Z the set of all possible rows partitions into K groups and W the set of all possible columns partitions into L groups. Let α_k and β_l be the row and column mixing proportions (belonging to $[0,1]$ and summing to 1), such that $p(z;\theta) = \prod_{ik}\alpha_k^{z_{ik}}$ and $p(w;\theta) = \prod_{jl}\beta_l^{w_{jl}}$. Let us also assume that, conditionally on (z,w), the basis expansion coefficients c_{ij} are independent and generated by a block-specific distribution: $p(c|z,w;\theta) = \prod_{ijkl} p(c_{ij};\theta_{kl})^{z_{ik}w_{jl}}$.

Thus,

$$p(c;\theta) = \sum_{z\in Z}\sum_{w\in W}\prod_{ik}\alpha_k^{z_{ik}}\prod_{jl}\beta_l^{w_{jl}}\prod_{ijkl} p(c_{ij};\theta_{kl})^{z_{ik}w_{jl}} \tag{2}$$

Depending on the grid of the study window, the studied time series can be very long leading to high dimensional coefficients c_{ij}. In order to suggest a parsimonious data modelling, we further suppose that the curves of each block kl ($k = 1, ..., K$, $l = 1, ..., L$) can be described into a low-dimensional functional latent subspace specific to each cluster, with intrinsic dimension $d_{kl} < S \times R$, through a principal component analysis for multivariate functional data (MFPCA, (Jacques and Preda, 2014b)) performed per block.

MFPCA is an extension of PCA for functional data (Ramsay and Silverman, 2005) to the multivariate functional case, representing the multivariate curves by a vector of principal scores into an eigen space formed by multivariate eigen functions. Thus each multivariate curve x_{ij}, conditionally to its belonging to block (k, l), can be represented by its scores $\delta_{ij} = (\delta_{ijr})_{1 \leq r \leq SR}$, with $S \times R$ the maximum number of non null principal components. Let us also define for later use Q_{kl}, a matrix of dimension $SR \times SR$ of eigen functions coefficients, which describes the linear mapping from the original space of c_{ij} to the low-dimensional subspace.

Conditionnaly to the block belonging, the scores are assumed to follow a Gaussian distribution with a parsimonious parametrization of the covariance matrix:

$$\delta_{ijr}^{kl} \sim \mathcal{N}(\mu_{kl}, \Delta_{kl}), \tag{3}$$

with $\mu_{kl} \in \mathbb{R}^{SR}$ and Δ_{kl} the diagonal covariance matrix defined as follows:

$$\Delta_{kl} = \begin{pmatrix} \boxed{\begin{matrix} a_{kl1} & & 0 \\ & \ddots & \\ 0 & & a_{kld_{kl}} \end{matrix}} & \mathbf{0} \\ \mathbf{0} & \boxed{\begin{matrix} b_{kl} & & 0 \\ & \ddots & \\ 0 & & b_{kl} \end{matrix}} \end{pmatrix} \begin{matrix} \left.\vphantom{\begin{matrix} a \\ a \\ a \end{matrix}}\right\} d_{kl} \\ \left.\vphantom{\begin{matrix} a \\ a \\ a \end{matrix}}\right\} SR - d_{kl} \end{matrix}$$

where $a_{kl1} > \ldots > a_{kld_{kl}} > b_{kl}$. With this assumption on Δ_{kl}, the first d_{kl} eigenvalues express the main part of the variability of the data, while the remaining ones reflect the noise and are modeled by a unique parameter b_{kl}. Thus, the space formed by the d_{kl} first eigen functions is a low-dimensional space which contains the main part of information about the data of a given block. The remaining information is considered as noise and modeled by a reduced number of parameters.

Thus, we can infer the distribution of one block curves coefficients according to:

$$c_{ij}|z_{ik}w_{jl} = 1 \sim \mathcal{N}(U_{kl}\mu_{kl}, U_{kl}\Sigma_{kl}U_{kl}^t + \Xi_{kl}), \tag{4}$$

where:

- $Q_{kl} = [U_{kl}, V_{kl}]$ with U_{kl} of size $SR \times d_{kl}$, V_{kl} of size $SR \times (SR - d_{kl})$ with $U_{kl}^t U_{kl} = I_{d_{kl}}$, $V_{kl}^t V_{kl} = I_{SR-d_{kl}}$ and $U_{kl}^t V_{kl} = 0$.
- Σ_{kl} is the matrix $diag(a_{kl1}, ..., a_{kld_{kl}})$.
- Ξ_{kl} is the noise variance matrix of size $SR \times SR$ such that $\Delta_{kl} = Q_{kl}^t(U_{kl}\Sigma_{kl}U_{kl}^t + \Xi_{kl})Q_{kl}$.

Thus, $\theta_{kl} = (\mu_{kl}, a_{kl}, b_{kl}, Q_{kl})$ and the whole set of model parameters is denoted by $\theta = (\alpha_k, \beta_l, \theta_{kl})_{1 \leq k \leq K, 1 \leq l \leq L}$.

In order to provide more parsimonious models, additional assumptions can be made on the different parameters $a_{kl\cdot}$, b_{kl} and d_{kl}, considering that they are common over cluster, over dimension, etc. This approach allows to generate a family of submodels of the general model introduced above. In this paper, we will detail the inference procedure to the submodel assuming $a_{klm} = a_{kl}, \forall m = 1, ..., d_{kl}$, since a good behaviour has been observed in practice. Nevertheless, the co-clustering method presented here can be derived for all models of the family extension, following the approach detailed in Schmutz et al. (2018).

3 Model inference

3.1 Model inference through a SEM-Gibbs algorithm

In co-clustering, the goal is to estimate the unknown row and column partitions z_{ik} and w_{jl}. Usually in mixture model, the maximum a posteriori rule is used, based on the estimation of model parameter θ maximizing the observed log-likelihood:

$$l(\theta; c) = \sum_{z,w} \log p(c; \theta). \tag{5}$$

Proof of this result is provided in Supplementary material A2. LBM implies a double missing structure (z and w) which makes inference harder than in a classical mixture model. Indeed, the use of the well known EM algorithm is not possible because the E step will need the computation of too much terms, which is not tractable in practice. We propose to use the stochastic version of the EM algorithm, embedding a Gibbs sampler for

generating the row and column partitions without having to compute their joint probability distribution (Keribin et al., 2010).

Starting from an initial column partition $w^{(0)}$ and initial parameter value $\theta^{(0)}$, the qth iteration of the partial SEM-Gibbs algorithm alternates between:

- **SE step** (Gibbs sampling): Alternates the two following steps until convergence:
  - simulate $z^{(q+1)}|c, w^{(q)}$ according to:

$$p(z_{ik} = 1|c, w^{(q)}; \theta^{(q)}) \quad = \quad \frac{\alpha_k^{(q)} f_k(c_i|w^{(q)}; \theta^{(q)})}{\sum_{k'} \alpha_{k'}^{(q)} f_{k'}(c_i|w^{(q)}; \theta^{(q)})}$$

    with $f_k(c_i|w^{(q)}; \theta^{(q)}) = \prod_{jl} p(c_{ij}; \theta_{kl}^{(q)})^{w_{jl}^{(q)}}$.
  - simulate $w^{(q+1)}|c, z^{(q+1)}$ according to:

$$p(w_{jl} = 1|c, z^{(q+1)}; \theta^{(q)}) = \frac{\beta_l^{(q)} f_l(c_j|z^{(q+1)}; \theta^{(q)})}{\sum_{l'} \beta_{l'}^{(q)} f_{l'}(c_j|z^{(q+1)}; \theta^{(q)})}.$$

    with $f_l(c_j|z^{(q+1)}; \theta^{(q)}) = \prod_{ik} p(c_{ij}; \theta_{kl}^{(q)})^{z_{ik}^{(q)}}$.

- **M step**: Update of $\theta^{(q+1)}$. The update of each parameter can be done in the same way than Schmutz et al. (2018). The mixing proportion and the block mean are updated by:
  - $\alpha_k^{(q+1)} = \frac{1}{n} \sum_i z_{ik}^{(q+1)}$ and $\beta_l^{(q+1)} = \frac{1}{p} \sum_j w_{jl}^{(q+1)}$,
  - $\mu_{kl}^{(q+1)} = \frac{1}{n_{kl}^{(q+1)}} \sum_{ij} c_{ij}^{z_{ik}^{(q+1)} w_{jl}^{(q+1)}}$ with $n_{kl}^{(q+1)} = \sum_{ij} z_{ik}^{(q+1)} w_{jl}^{(q+1)}$.

For the variance parameters a_{kl}, b_{kl} and Q_{kl}, let us define the sample covariance matrix C_{kl} of block kl:

$$C_{kl}^{(q)} = \frac{1}{n_{kl}^{(q)}} \sum_{i=1}^{n} \sum_{j=1}^{p} z_{ik}^{(q+1)} w_{jl}^{(q+1)} (c_{ij} - \mu_{kl}^{(q)})^t (c_{ij} - \mu_{kl}^{(q)}).$$

Let also introduce W the $SR \times SR$ matrix of inner products: $W = \int_0^T \phi^t(t)\phi(t)dt$, with $\phi(t)$ the matrix that gathers basis functions of all S functional variables:

$$\phi(t) = \begin{pmatrix} \phi_{11}(t) & ... & \phi_{1R}(t) & 0 & ... & 0 & ... & 0 & ... & 0 \\ 0 & ... & 0 & \phi_{21}(t) & ... & \phi_{2R}(t) & ... & 0 & ... & 0 \\ & & & & ... & & & & & \\ 0 & ... & 0 & 0 & ... & 0 & ... & \phi_{S1}(t) & ... & \phi_{SR}(t) \end{pmatrix}.$$

Then,

- the d_{kl} first columns of the matrix of eigen functions coefficients $Q_{kl}^{(q)}$ are updated by the eigen functions coefficients associated with the largest eigenvalues of $W^{1/2}C_{kl}^{(q)}W^{1/2}$,
- the variance parameters $a_{kl}^{(q+1)}$, are updated by the mean of the d_{kl} largest eigenvalues of $W^{1/2}C_{kl}^{(q)}W^{1/2}$,
- the variance parameters b_{kl} are updated by $\frac{1}{R-d_{kl}}(trace(W^{1/2}C_{kl}^{(q)}W^{1/2}) - d_{kl}a_{kl}^{(q)})$.

Proofs of those results are available in Supplementary material A3, A4 and A5.

In brief, the SEM-Gibbs algorithm is run for a given number of iterations. After a burn-in period, the final estimation $\hat{\theta}$ of the parameters is obtained by the mean of the sample distribution (without the burn-in iterations). Then, a new Gibbs sampler is used to sample $(\hat{z}, \hat{w})$ according to $\hat{\theta}$, and the final partition $(\hat{z}, \hat{w})$ is obtained by the marginal mode of this sample distribution.

Initialization of the algorithm As said previously, our algorithm relies on a SEM-Gibbs algorithm. This algorithm needs to be initialized with values for column partitions and parameters. In practice, column and row partitions are initialized, and corresponding

initial parameter values are deduced. To this end, three initialization strategies are available here : random, *k-means* and *funFEM*. In the random case, partitions are randomly sampled from a multinomial distribution with uniform probabilities. *k-means* strategy consists of initializing partitions with those obtained by *k-means* method directly applied on discretized data. Finally, *funFEM* aims to initialize partitions by applying the *funFEM* algorithm (Bouveyron et al., 2015). We will see later, in the numerical experimentation section, that *funFEM* is the one that gives the best results.

3.2 Choice of the number of clusters

We now discuss the choice of the hyper-parameters K and L, *i.e.* the number of row clusters and column clusters respectively. The choice of these hyper-parameters is viewed here as a model selection problem. Well established model selection tools are Akaike information criterion (AIC, (Akaike, 1974)), Bayesian information criterion (BIC, (Schwarz, 1978)) and Integrated Classification Likelihood (ICL, (Biernacki et al., 2000)). However, in the co-clustering case, the likelihood is not tractable for the same reason than the EM algorithm is not usable. Consequently, AIC and BIC are not tractable. Conversely, the ICL criterion can be considered since it relies on the completed log-likelihood, which is tractable. Adapted to our model, the ICL criterion is:

$$ICL(K,L) = \log p(c,\hat{z},\hat{w};\hat{\theta}) - \frac{K-1}{2}log(n) - \frac{L-1}{2}log(p) - \frac{\nu}{2}log(np)$$

where $\nu = KLSR + 2KL + \sum_{kl} d_{kl}(SR - \frac{d_{kl}+1}{2})$ is the number of continuous parameters per block and

$$log\ p(c,\hat{z},\hat{w};\hat{\theta}) = \prod_{ik} \hat{z}_{ik} log(\alpha_k) + \prod_{jl} \hat{w}_{jl} log(\beta_l) + \sum_{ijkl} \hat{z}_{ik}\hat{w}_{jl} log\ p(c_{ij};\hat{\theta}_{kl}).$$

The couple (K, L) leading to the highest ICL value is selected as the most appropriate number of row and column clusters.

4 Numerical experimentation on simulated data

This section presents numerical experiments on simulated data in order to illustrate the behavior of the proposed methodology in presence of different noise ratio in data and to study the selection of the number of row and column clusters. The R code for our multivariate functional co-clustering algorithm is available under request and will be soon available on CRAN as an R package.

For all examples, we set to 50 iterations the burn-in period of the algorithm and the SEM-Gibbs maximal number of iterations is set to 100.

4.1 Introductory example

4.1.1 Simulation setup

A sample of $n = 100$ bivariate curves are simulated with $K = 4$, $L = 3$ and $p = 100$. The proportions of row clusters α used is (0.2,0.4,0.1,0.3) and column clusters β is (0.4,0.3,0.3). The first functional variable is designed from four different functions that are used as blocks mean at 31 equi-spaced time points, $t = 0, 1/30, 2/30, ..., 1$:

$$x_{ij}(t)|z_{ik}w_{jl} = 1 \sim \mathcal{N}(m_{kl}(t), s^2),$$

where $s = 0.3$ and the mean function is taken from $m_{11} = m_{21} = m_{33} = m_{42} = f_1$, $m_{12} = m_{22} = m_{31} = f_2$, $m_{13} = m_{32} = f_3$ et $m_{23} = m_{41} = m_{43} = f_4$, with $f_1(t) = sin(4\pi t)$, $f_2(t) = 0.75 - 0.5\mathbb{1}_{t\in]0.7,0.9[}$, $f_3(t) = h(t)/max(h(t))$ where $h(t) = \mathcal{N}(0.2, \sqrt{0.02})$ and $f_4(t) =$

$sin(10\pi t)$. Then the second variable is designed according to the same process than the first one but with four different functions: $f_1(t) = cos(4\pi t)$, $f_2(t) = 0.75 - 0.5\mathbb{1}_{t\in]0.2,0.4[}$, $f_3(t) = h(t)/max(h(t))$ where $h(t) = \mathcal{N}(0.2, \sqrt{0.05})$ and $f_4(t) = cos(10\pi t)$. The block means functions are shown in Figure 1.

Starting from this simulation setting, five scenarios are build by adding some noise fraction within the blocks by randomly simulating a percentage τ of curves using other block means: 0% (scenario 1), 10% (scenario 2), 30% (scenario 3), 50% (scenario 4) and 80% (scenario 5).

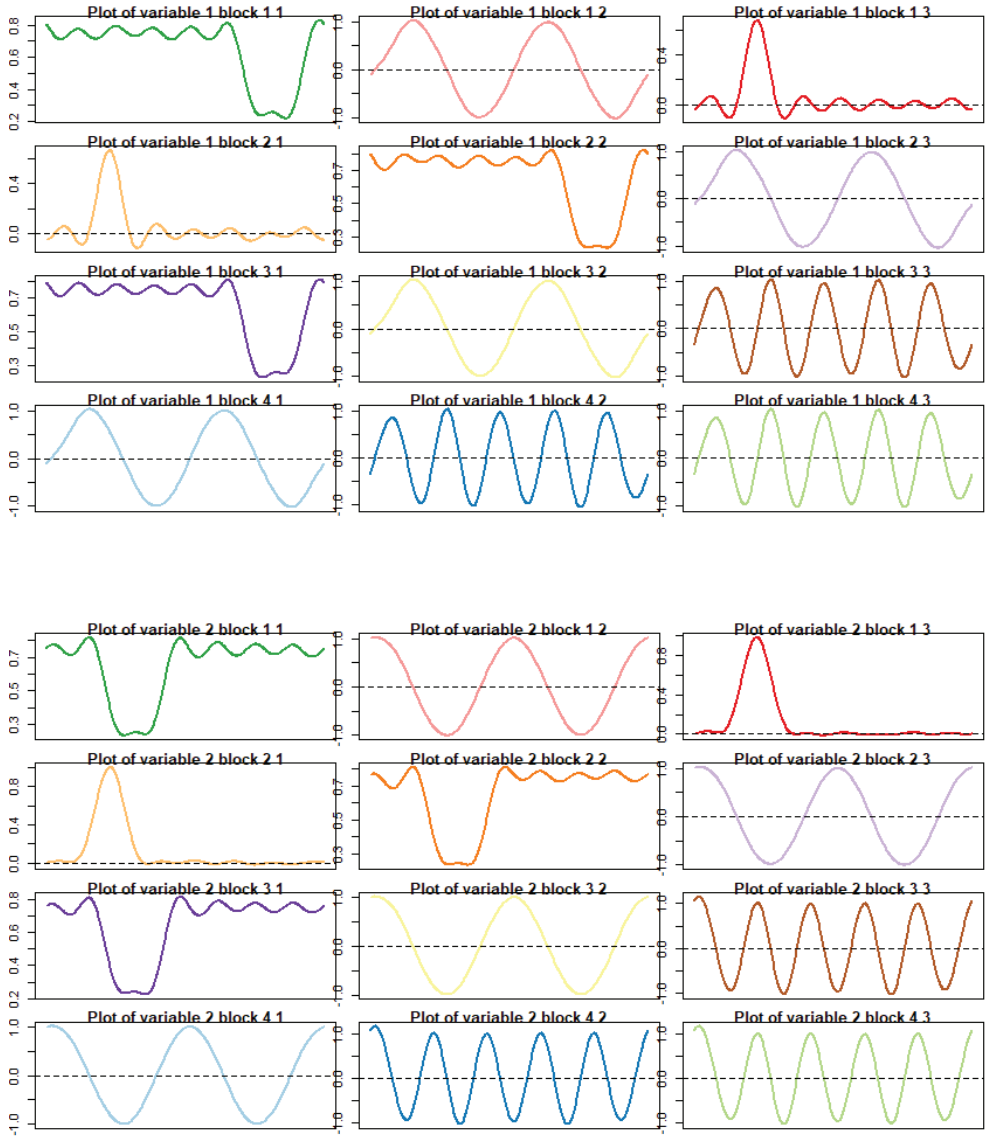


Figure 1: Block means functions for the first variable (top) and second variable (bottom)

4.1.2 Results

In order to illustrate the good behaviour of our algorithm in the case of noisy data and to compare the influence of algorithm initialization on co-clustering results, 20 simulations have been performed for each scenario with both *k-means*, *funFEM* and *random* initializations. The algorithm is applied for $K = 4$ and $L = 3$ and with Fourier smoothing with 15 basis functions. The quality of estimated partitions is assessed with the Adjusted Rand Index (ARI, (Rand, 1971)).

Results are shown in Figure 2. We can see that co-clustering results are almost perfect for the 4 first scenarios with *funFEM* initialization, and the 2 first scenarios for *k-means* initialization. As expected, the algorithm performance decreases while noise increases, but median ARI value is always above 0.8 in the case of four first scenarios with *k-means* and *funFEM* initialization. Moreover *k-means* initialization performs better than random when the noise ratio is upper than 50%. And the *funFEM* initialization performs better than the *k-means* one.

To conclude, in view of the good behaviour of *funFEM* initialization in both previous examples, we recommend to use firstly *funFEM* initialization rather than the two others available in the algorithm.

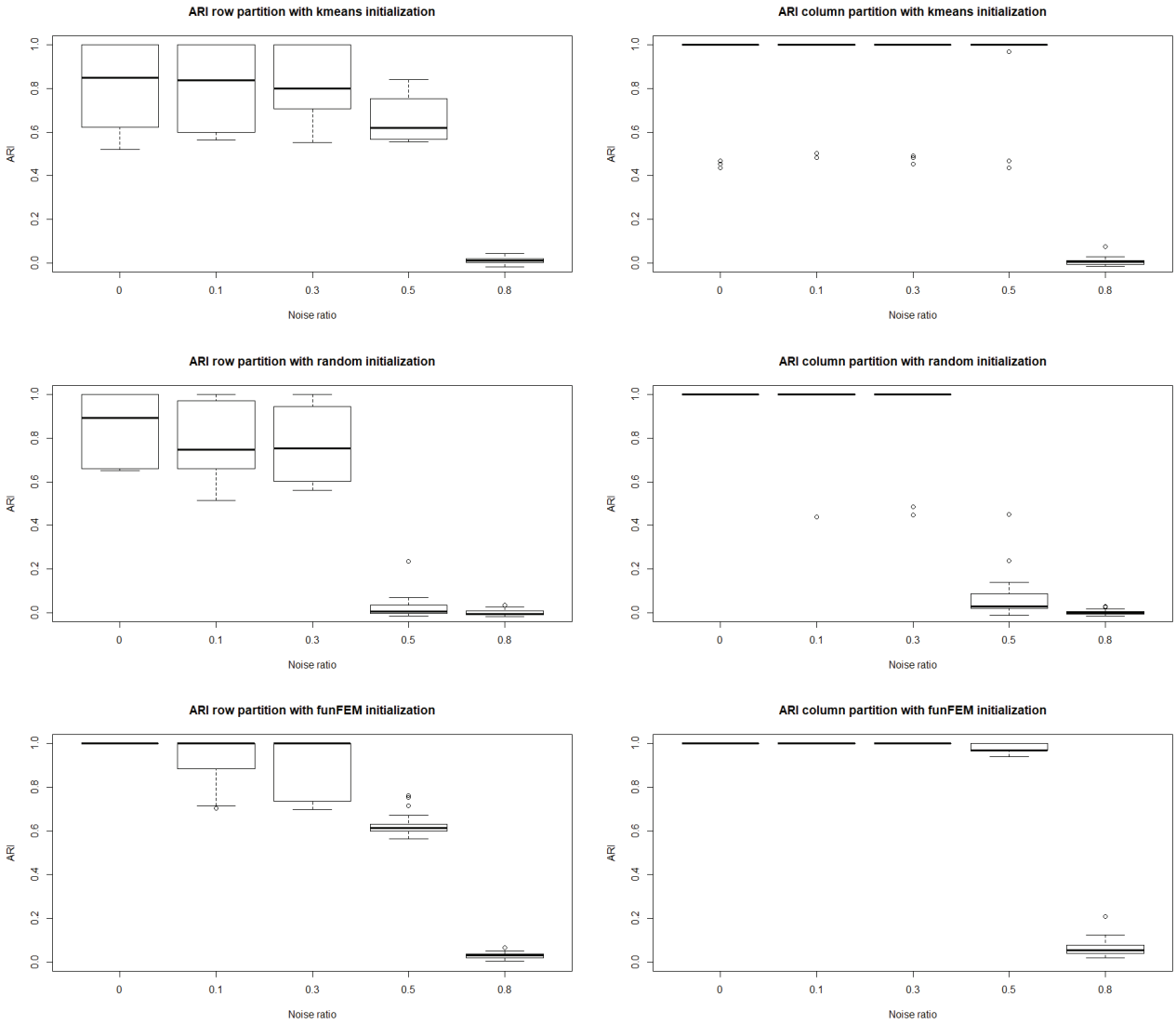


Figure 2: Results of ARI for each scenario for *k-means* (top), random (middle) and *funFEM* (bottom) initialization

Table 1: Number of times each partition is selected by ICL for 20 simulations (in percent). Highlighted rows and columns correspond to the actual values for K and L.

Scenario $\tau = 0$						Scenario $\tau = 0.1$					
K/L	2	3	4	5	6	K/L	2	3	4	5	6
2	0	0	0	0	0	2	0	0	0	0	0
3	0	0	0	0	0	3	0	0	0	0	0
4	0	**100**	0	0	0	4	0	**100**	0	0	0
5	0	0	0	0	0	5	0	0	0	0	0
6	0	0	0	0	0	6	0	0	0	0	0

Scenario $\tau = 0.3$						Scenario $\tau = 0.5$					
K/L	2	3	4	5	6	K/L	2	3	4	5	6
2	0	0	0	0	0	2	0	0	0	0	0
3	0	0	0	0	0	3	0	0	0	0	0
4	0	**100**	0	0	0	4	0	**90**	0	0	0
5	0	0	0	0	0	5	0	10	0	0	0
6	0	0	0	0	0	6	0	0	0	0	0

4.2 Model selection

In this section, the selection of the number of clusters is investigated. Data are generated as previously. The simulation setting is repeated 20 times with $n = 500$, $p = 500$ and $t = 30$. The algorithm is run for 2 to 6 clusters with *funFEM* initialization and the best model selected by ICL is noted down. Results are shown in Table 1.

Model selection with *funFEM* is perfect with a noise ratio from 0 to 30% of the data

volume. Then, as expected, the performance of the criterion decreases, for a noise ratio of 50% the ICL criterion gets back to the true partition in 90% of cases.

5 Co-clustering for energy waste detection in French households

This section focus on the analysis of French households' electric consumption according to their indoor and outdoor temperatures. With more and more involvement of the mainstream on environmental issues and with the COP24 aim of evaluating government efforts to tackle against global warming, it is important to find ways to save energy. This type of data can help detecting potentially poor insulated homes and target those households with rehabilitation offers which would lead to less energy waste.

5.1 Data

This data set deals with electric consumption, indoor temperature and outdoor temperature of 356 French households representatives of metropolitan France. It has been provided by EDF company, a French electricity provider. Data are collected every 30 minutes for 173 days (48 time points per day, cf. Figure 3), from July 2009 to December 2009. Their objectives are to model insulation efficiency, detect weeks of absence in order to adapt electric load and evaluate the impact of indoor temperature on the electric consumption. The data set contains some missing observations. Since the first step of our analysis consists in estimating the basis expansion approximations of the curves, it is not a problem if there is some missing observations (except if they are at the beginning or at the end of the

period). Indeed, smoothing the data into a basis expansion can be performed even if some of the 48 time points are not observed. Nevertheless, we remove from the analysis daily curves with too many missing time points (more than 41) or for which missing values occur at the beginning or at the end of the period.

The functional form of the whole database is reconstructed using a Fourier smoothing with 15 basis functions. Our algorithm has been applied with *funFEM* initialization with a varying number of row and column clusters, from 2 to 20 on normalized data. The ICL criteria is used to choose an appropriate number of row and column clusters.

Table 2: ICL values for the 10 first partitions

Number of row clusters K	Number of column clusters L	ICL
5	14	-3944668
5	13	-3946480
5	11	-3951143
4	14	-3958896
3	13	-3961862
5	10	-3963644
4	12	-3966577
3	14	-3967567
6	8	-3968532
6	9	-3970104

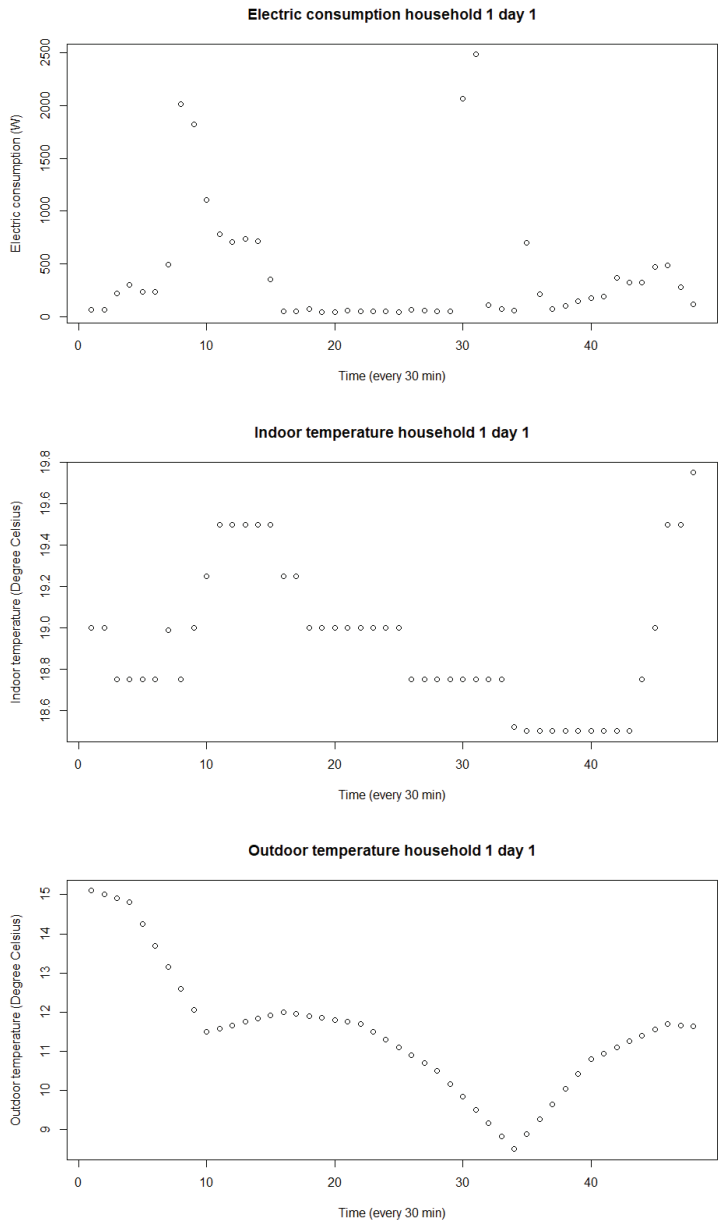


Figure 3: Raw data for one individual and one day of measurement

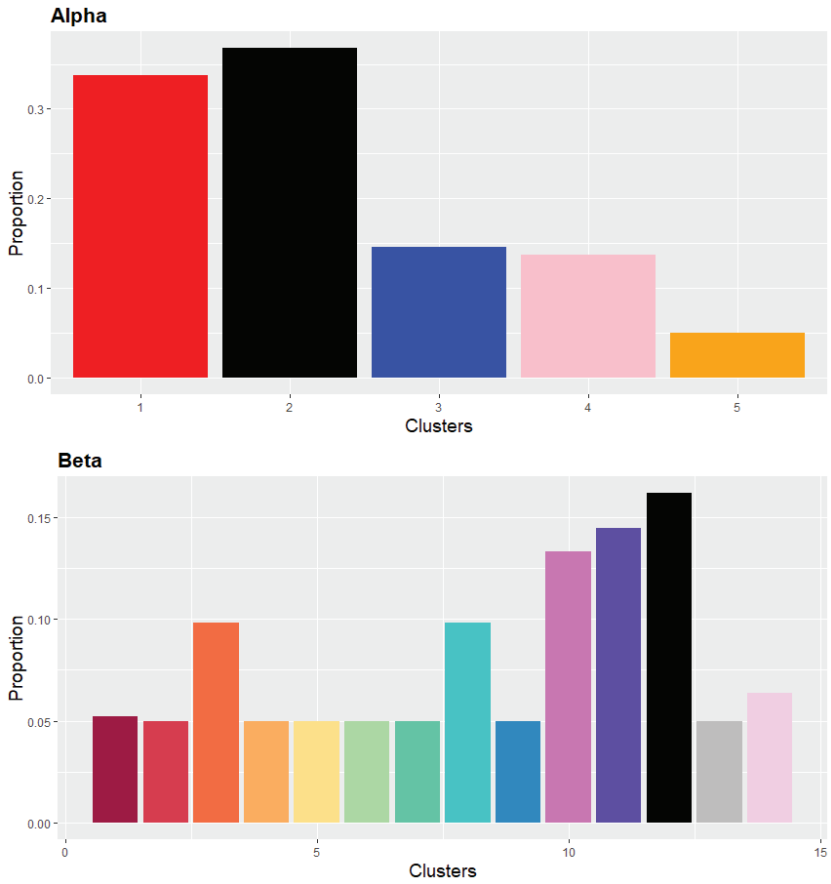


Figure 4: Proportion of households in row clusters (top) and days in column clusters (bottom)

5.2 Results

According to ICL, the best partition is with 5 row clusters and 14 column clusters (cf. Table 2). Clusters proportions can be seen on Figure 4.

The obtained block clusters can be described with their mean curves. Since the number of column clusters is large, we focus our interpretation on four typical column clusters in order to ease the analysis: Figure 5 plots the mean curves for row-cluster 1 to 5 and column cluster 1, 6, 10 and 14. The whole set of mean curves is plotted in supplementary material A1 (Figures 10, 11 and 12).

On average French households indoor temperature is between 17°C and 25°C over the 6 months measurement period (Figure 5) and whatever the geographical location (Figure 8). Presumably it is more dictated by the way of life than by the area climate type. For instance, the third cluster of households (row-cluster) is the one with the lowest electric consumption (third row on Figure 7) although it is distributed all over the country (blue points on Figure 8). This cluster may correspond to well insulated houses or households whose heating system is not electrical. The first and second cluster of households (red and black points on Figure 8) have a medium consumption (first and second rows on Figure 7) and are distinguished by the outdoor temperature (first and second rows on Figure 6) which is a bit more flat for the first cluster (red): the delta of temperature for one day is less important for red households on the measured period than for black ones. The fifth cluster of households (orange points on Figure 8) have a high and uncontrolled electricity consumption (last row on Figure 7) and an indoor temperature well below the one of other clusters on the whole time period (last row on Figure 5). We may suppose that those households belong to retired persons who have periods of extended vacations because they are the only one where the indoor temperature stays at 15°C during 4 time periods. Those

households are also the only ones with a very variable aspect of the indoor temperature mean curves and consumption curves, it may be explained by switching on and off their heating system many times along a day, whereas other households may be equipped with a timer that holds the wanted temperature. Lastly, the fourth cluster of households (pink points on Figure 8) has a very high electricity consumption (about twice that of other clusters) whereas the outdoor temperature is not especially lower (fourth row on Figure 6) and the indoor temperature not especially greater (between 18°C to 22°C, fourth row on Figure 5). Those households may be poor insulated and it may be important to alert them on their consumption in order to assist them in reducing their invoices by undertaking renovations to their property and also to reduce their ecological footprint.

Finally, Figure 9 presents the distribution of the column clusters (clusters of days) along the period under study. Note that there is no link between the color of the row and column clusters. We can see that the pink cluster (14th column cluster) corresponds to the coldest days of the period (outdoor temperature below 5°C). For those days the electric consumption of all households is higher than their average one on the other periods (last column on Figure 7).

To the contrary, the hottest days where mainly in July (first and second column cluster on Figure 12). We can see that the maximum electric consumption reached in a day does not differ a lot between those days and days of medium outdoor temperatures. We suppose that for these periods the main consumption areas are other electric devices than the heating system.

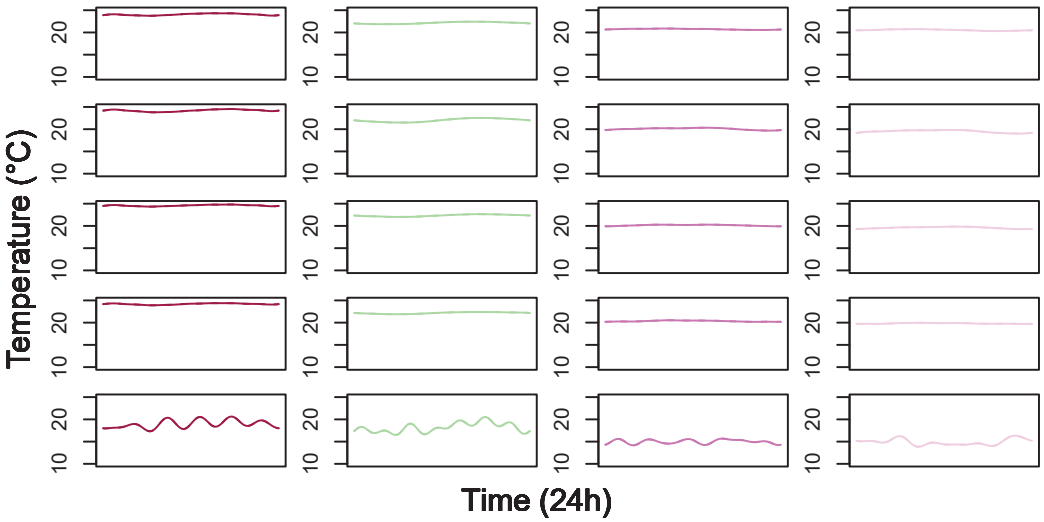


Figure 5: Co-clustering results for indoor temperature for column cluster 1, 6, 10 and 14

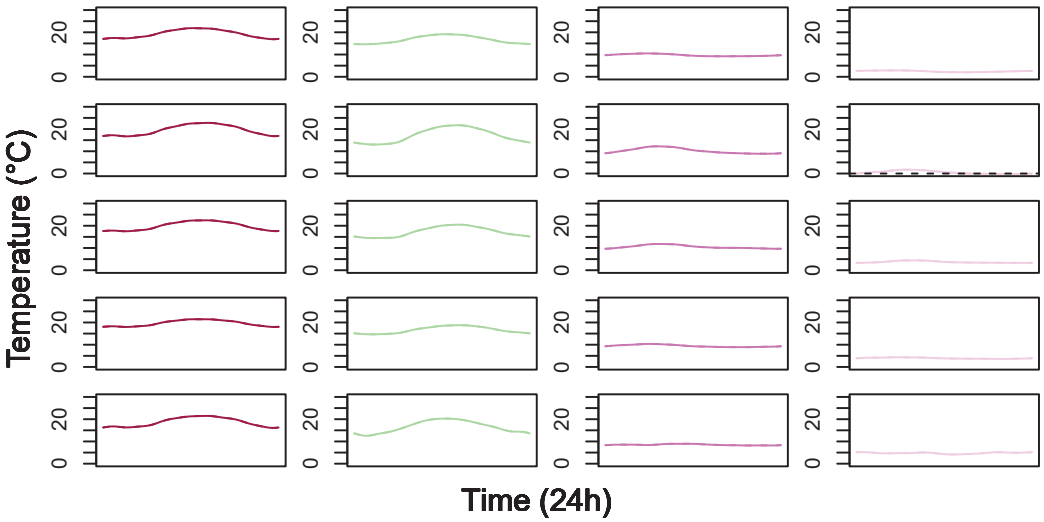


Figure 6: Co-clustering results for outdoor temperature for column cluster 1, 6, 10 and 14

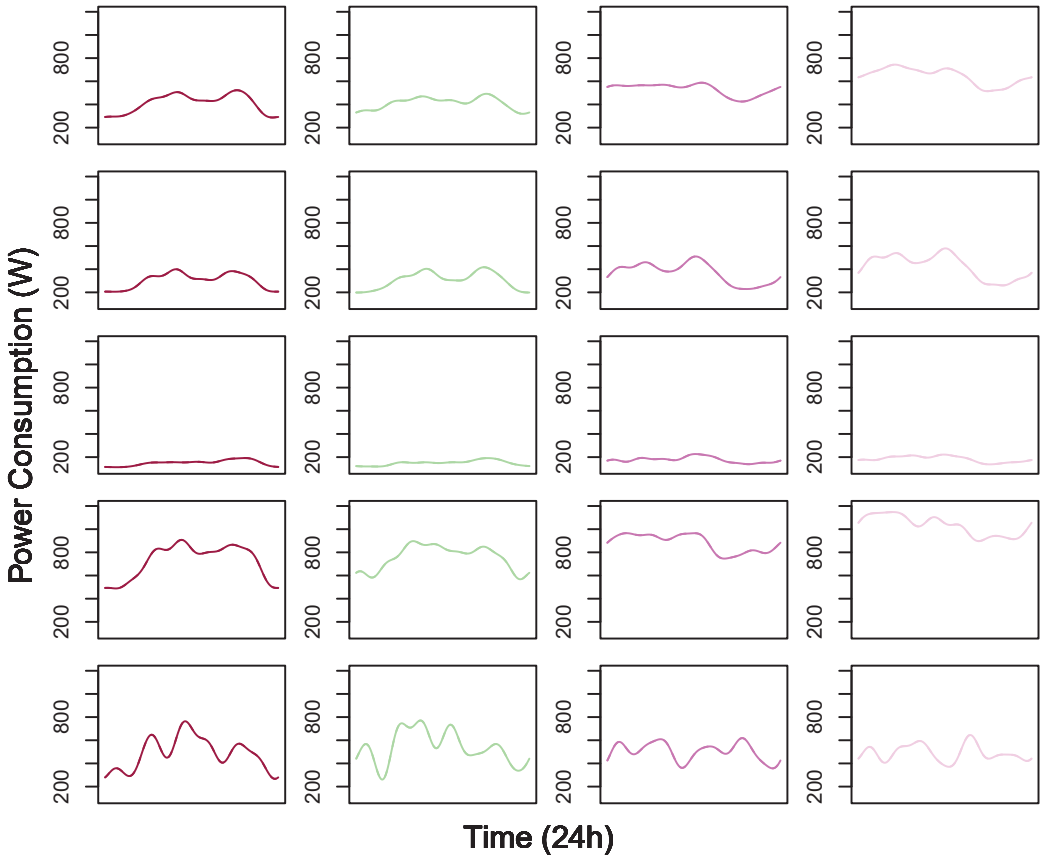


Figure 7: Co-clustering results for electric consumption for column cluster 1, 6, 10 and 14

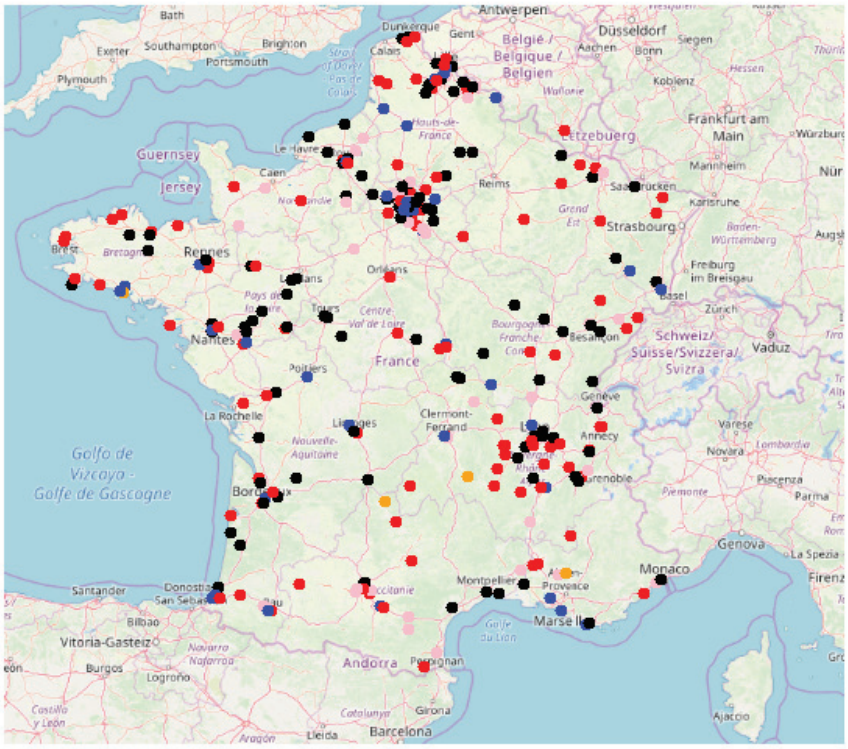


Figure 8: Geographical position of households by row clusters (colored by cluster)

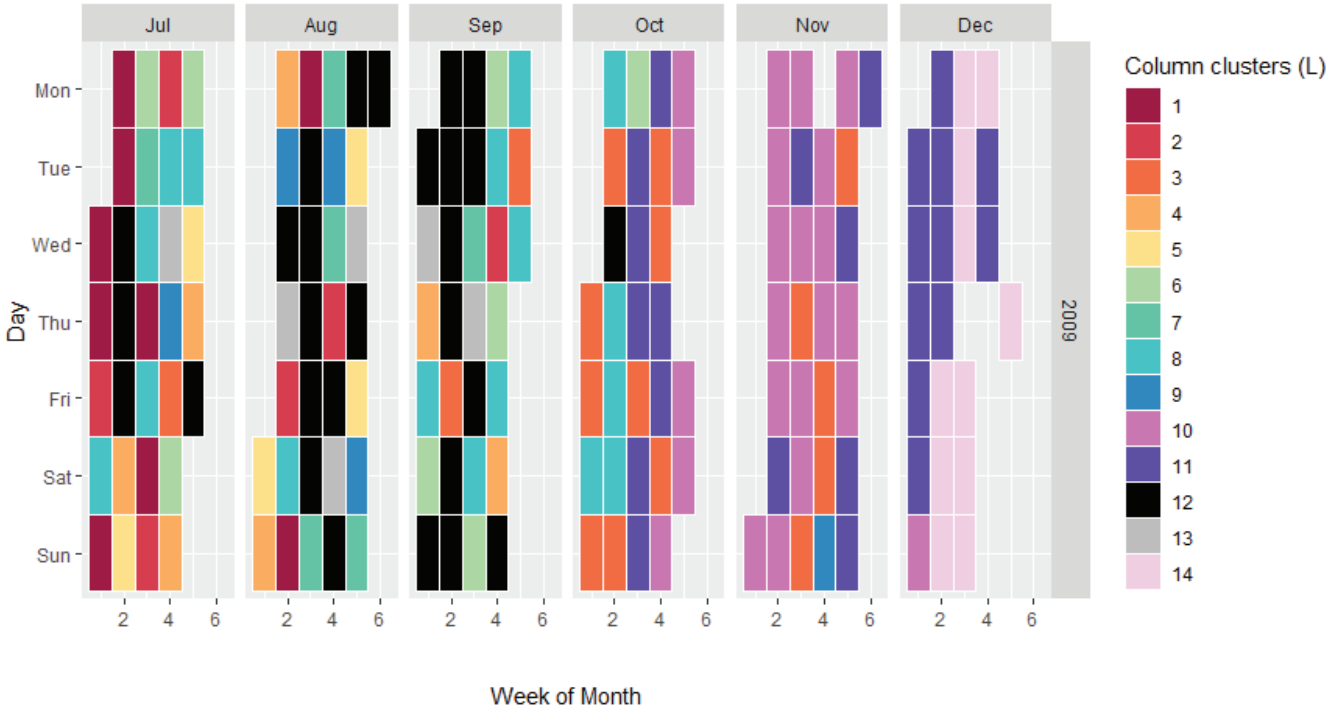


Figure 9: Calendar position of column clusters (colored by cluster)

To conclude, the analysis of this large multivariate functional data set with the co-clustering model has allowed the distinction of different habits profiles. Those results have especially highlighted that French households' electric consumption and indoor temperature do not depend on the geographical location and thus on the outdoor temperature, but more on the way of life of people and the insulation level. Actions can be undertaken to further raise awareness on the impact of reducing their environmental footprint by reducing their indoor temperature to a maximum value of, at least, 21°C. Moreover some households have been identified by their abnormally high electric consumption, those houses should be targeted by the electric provider with rehabilitation offers to reduce their energy waste.

6 Discussion and conclusion

This work was motivated by the will to detect poor insulated buildings and long periods of absence in order to reduce energy waste. The proposed co-clustering model answers both objectives, which will allow EDF company to alert users on their over-consumption and advise them on renovation work. Furthermore, the field of operational applications of co-clustering methods for multivariate functional curves is wide at EDF company. For instance, it can help design new marketing offers or services such as co-clustering load curves and photo-voltaic production data in order to identify clusters with low yields or detect anomalies. In the future, EDF will have more and more data coming from connected devices, like connected thermostats or connected weather stations, this data flood deserves to be classified to ease it interpretation. The proposed algorithm answers this need. The co-clustering method for multivariate functional data, allows to cluster both individuals and variables simultaneously. The proposed approach relies on a functional latent

block model, which assumes for each block a probabilistic distribution for the scores of the multivariate curves obtained from a multivariate functional principal component analysis. Model inference relies on a SEM-Gibbs algorithm which alternates a SE-step where row and column partitions are simulated according to Gibbs algorithm, and a M-step where model parameters are updated thanks to the previous simulated partitions. Lastly, the best number of row and column clusters is selected thanks to the ICL criterion. As far as the authors know, it is the first algorithm available for functional multivariate co-clustering.

SUPPLEMENTARY MATERIAL

A1. Co-clustering results for $K = 5$ and $L = 14$

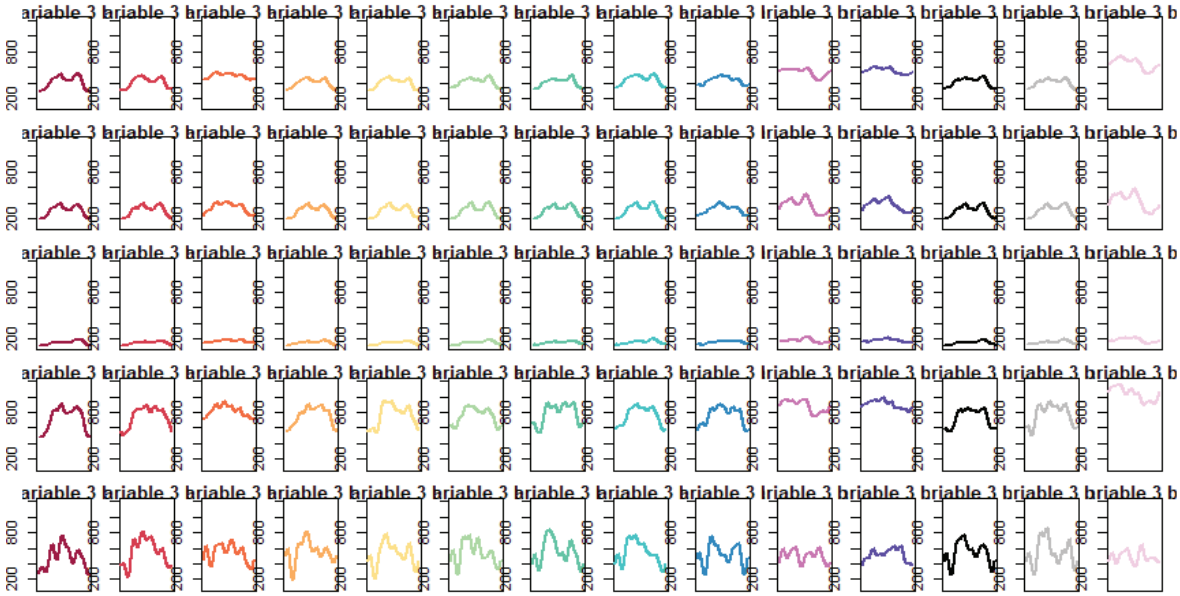


Figure 10: Co-clustering results for electric consumption

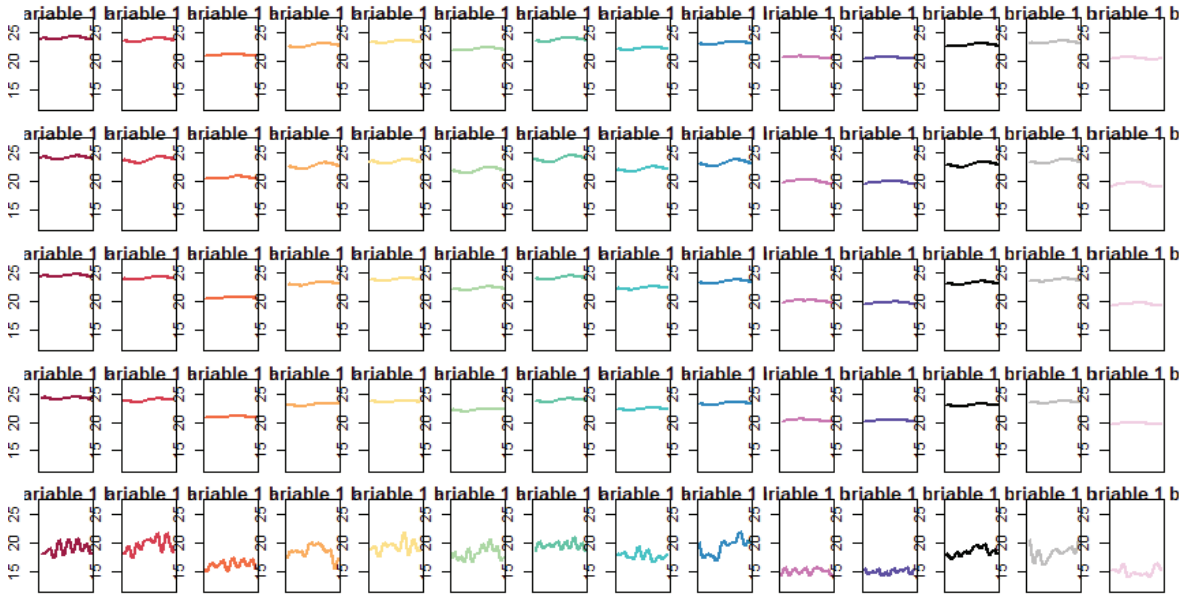


Figure 11: Co-clustering results for indoor temperature

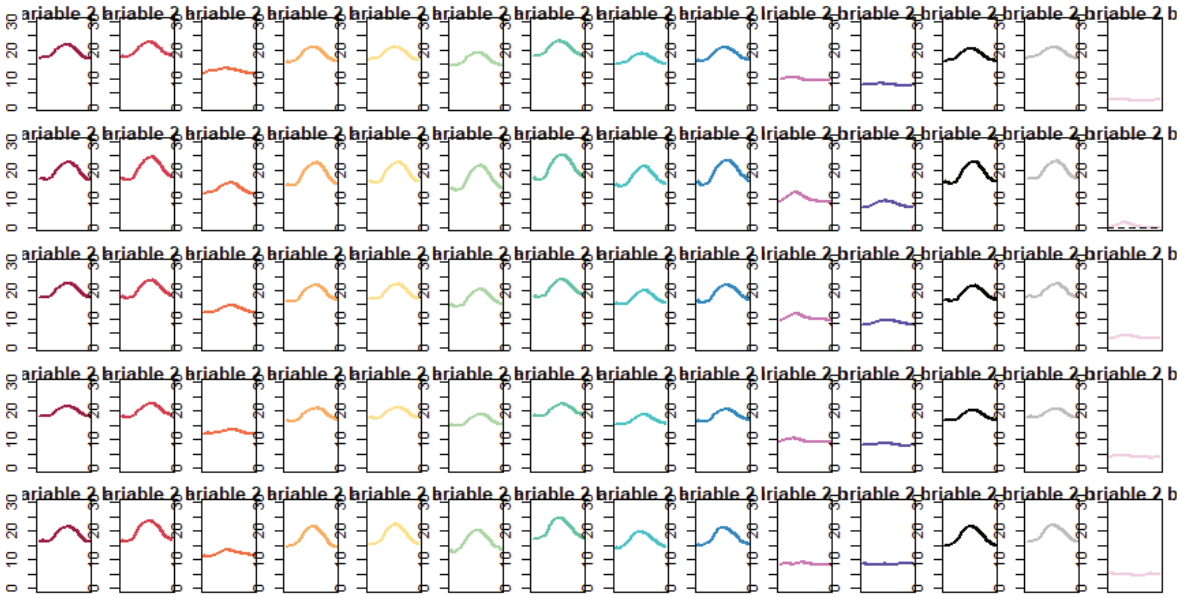


Figure 12: Co-clustering results for outdoor temperature

A2. Log-likelihood formula, proof of Equation 5:

$$
\begin{aligned}
l(\theta) &= \sum_{i=1}^{n}\sum_{j=1}^{SR}\sum_{k=1}^{K}\sum_{l=1}^{L} z_{ik}w_{jl}log(\pi_{kl}f(c_{ij},\theta_{kl})) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{SR}\sum_{k=1}^{K}\sum_{l=1}^{L} z_{ik}w_{jl}log(\frac{\pi_{kl}}{(2\pi)^{SR/2}|\Sigma_{kl}|^{1/2}}exp(-\frac{1}{2}(c_{ij}-\mu_{kl})^t\Sigma^{-1}(c_{ij}-\mu_{kl}))) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{SR}\sum_{k=1}^{K}\sum_{l=1}^{L} z_{ik}w_{jl}[log(\pi_{kl})-\frac{1}{2}log|\Sigma_{kl}|-\frac{SR}{2}log(2\pi)-\frac{1}{2}(c_{ij}-\mu_{kl})^t\Sigma^{-1}(c_{ij}-\mu_{kl})] \\
&= -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{SR}\sum_{k=1}^{K}\sum_{l=1}^{L} z_{ik}w_{jl}[-2log(\pi_{kl})+d_{kl}log(a_{kl})+(SR-d_{kl})log(b_{kl}) \\
&+ (c_{ij}-\mu_{kl})^t\Sigma^{-1}(c_{ij}-\mu_{kl})]-\frac{nSR}{2}log(2\pi)
\end{aligned}
$$

Let $n_{kl}=\sum_{i=1}^{n}\sum_{j=1}^{SR} z_{ik}w_{jl}$ be the number of block kl curves,

$$
\begin{aligned}
l(\theta) &= -\frac{1}{2}\sum_{k=1}^{K}\sum_{l=1}^{L} n_{kl}[-2log(\pi_{kl})+d_{kl}log(a_{kl})+(SR-d_{kl})log(b_{kl}) \\
&+ \frac{1}{n_{kl}}\sum_{i=1}^{n}\sum_{j=1}^{SR} z_{ik}w_{jl}(c_{ij}-\mu_{kl})^tQ_{kl}\Delta_{kl}^{-1}Q_{kl}^t(c_{ij}-\mu_{kl})]-\frac{nSR}{2}log(2\pi)
\end{aligned}
$$

The quantity $(c_{ij}-\mu_{kl})^tQ_{kl}\Delta_{kl}^{-1}Q_{kl}^t(c_{ij}-\mu_{kl})$ is a scalar, so it is equal to it trace:

$$
\frac{1}{n_{kl}}\sum_{i=1}^{n}\sum_{j=1}^{SR} z_{ik}w_{jl}(c_{ij}-\mu_{kl})^tQ_{kl}\Delta_{kl}^{-1}Q_{kl}^t(c_{ij}-\mu_{kl}) = \frac{1}{n_{kl}}\sum_{i=1}^{n}\sum_{j=1}^{SR} z_{ik}w_{jl}tr((c_{ij}-\mu_{kl})^tQ_{kl}\Delta_{kl}^{-1}Q_{kl}^t(c_{ij}-\mu_{kl})).
$$

Well $tr([(c_{ij}-\mu_{kl})^tQ_{kl}]\times[\Delta_{kl}^{-1}Q_{kl}^t(c_{ij}-\mu_{kl})]) = tr([\Delta_{kl}^{-1}Q_{kl}^t(c_{ij}-\mu_{kl})]\times[(c_{ij}-\mu_{kl})^tQ_{kl}])$,

consequently:

$$
\begin{aligned}
& \frac{1}{n_{kl}} \sum_{i=1}^{n} \sum_{j=1}^{SR} z_{ik} w_{jl} (c_{ij} - \mu_{kl})^t Q_{kl} \Delta_{kl}^{-1} Q_{kl}^t (c_{ij} - \mu_{kl}) \\
= & \frac{1}{n_{kl}} \sum_{i=1}^{n} \sum_{j=1}^{SR} z_{ik} w_{jl} tr(\Delta_{kl}^{-1} Q_{kl}^t (c_{ij} - \mu_{kl})(c_{ij} - \mu_{kl})^t Q_{kl}) \\
= & tr(\Delta_{kl}^{-1} Q_{kl}^t [\frac{1}{n_{kl}} \sum_{i=1}^{n} \sum_{j=1}^{SR} z_{ik} w_{jl} (c_{ij} - \mu_{kl})^t (c_{ij} - \mu_{kl})] Q_{kl}) \\
= & tr(\Delta_{kl}^{-1} Q_{kl}^t C_{kl} Q_{kl}),
\end{aligned}
$$

where $C_{kl} = \frac{1}{n_{kl}} \sum_{i=1}^{n} \sum_{j=1}^{SR} z_{ik} w_{jl} (c_{ij} - \mu_{kl})^t (c_{ij} - \mu_{kl})$ is the empirical covariance matrix of the k-th element of the mixture model. The Δ_{kl} matrix is diagonal, so we can write:

$$
\begin{aligned}
\frac{1}{n_{kl}} \sum_{i=1}^{n} \sum_{j=1}^{SR} z_{ik} w_{jl} (c_{ij} - \mu_{kl})^t Q_{kl} \Delta_{kl}^{-1} Q_{kl}^t (c_{ij} - \mu_{kl}) &= \sum_{j=1}^{d_{kl}} \frac{q_{klj}^t W^{1/2} C_{kl} W^{1/2} q_{klj}}{a_{klj}} \\
&+ \sum_{j=d_{kl}+1}^{SR} \frac{q_{klj}^t W^{1/2} C_{kl} W^{1/2} q_{klj}}{b_{kl}},
\end{aligned}
$$

where q_{klj} is j-th column of Q_{kl}.

Finally,

$$
\begin{aligned}
l(\theta) &= -\frac{1}{2} \sum_{k=1}^{K} \sum_{l=1}^{L} n_{kl} [-2log(\pi_{kl}) + d_{kl} log(a_{kl}) + (R - d_{kl}) log(b_{kl}) \\
&+ \sum_{j=1}^{d_{kl}} \frac{q_{klj}^t W^{1/2} C_{kl} W^{1/2} q_{klj}}{a_{klj}} + \sum_{j=d_{kl}+1}^{SR} \frac{q_{klj}^t W^{1/2} C_{kl} W^{1/2} q_{klj}}{b_{kl}}] + \frac{nSR}{2} log(2\pi)
\end{aligned}
$$

A3. Parameter Q_{kl} update:

We have to maximize the log-likelihood under the constraint $q_{klj}^t q_{klj} = 1$, with q_{klj} the jth column of Q_{kl}. This is equivalent to look for a saddle point of the Lagrange function:

$$\mathcal{L} = -2l(\theta) - \sum_{j=1}^{SR} \omega_{klj}(q_{klj}^t q_{klj} - 1)$$

where ω_{klj} are Lagrange multipliers. Thus, we can write:

$$\begin{aligned}
\mathcal{L} &= \sum_{k=1}^{K}\sum_{l=1}^{L} n_{kl}[d_{kl}log(a_{kl}) + \sum_{j=1}^{d_{kl}} \frac{q_{klj}^t W^{1/2} C_{kl} W^{1/2} q_{klj}}{a_{klj}} \\
&+ (SR - d_{kl})log(b_{kl}) + \sum_{j=d_{kl}+1}^{SR} \frac{q_{klj}^t W^{1/2} C_{kl} W^{1/2} q_{klj}}{b_{klj}} - 2log(\pi_{kl})kl)] \\
&+ \frac{nSR}{2} log(2\pi) - \sum_{j=1}^{SR} \omega_{klj}(q_{kj}^t q_{klj} - 1).
\end{aligned}$$

Therefore, the gradient of $\mathcal{L}$ in relation to q_{kjl} is:

$$\begin{aligned}
\nabla_{q_{klj}}\mathcal{L} &= \nabla_{q_{klj}}(\sum_{k=1}^{K}\sum_{l=1}^{L} n_{kl}[\sum_{j=1}^{d_{kl}} \frac{q_{klj}^t W^{1/2} C_{kl} W^{1/2} q_{klj}}{a_{klj}} + \sum_{j=d_{kl}+1}^{SR} \frac{q_{klj}^t W^{1/2} C_{kl} W^{1/2} q_{klj}}{b_{kl}}] \\
&- \sum_{j=1}^{SR} \omega_{klj}(q_{klj}^t q_{klj} - 1)).
\end{aligned}$$

As a reminder, when W is symmetric, then $\frac{\partial}{\partial x}(x-s)^T W(x-s) = 2W(x-s)$ and $\frac{\partial}{\partial x}(x^T x) = 2x$ (cf. Petersen and Pedersen (2012)), so:

$$\nabla_{q_{klj}}\mathcal{L} = n_{kl}[2\frac{W^{1/2} C_{kl} W^{1/2}}{\sigma_{klj}} q_{klj}] - 2\omega_{klj} q_{klj}$$

where σ_{klj} is the j-th diagonal term of matrix Δ_k.

Thus,

$$\begin{aligned} q_{klj}^t \nabla_{q_{klj}} \mathcal{L} = 0 \quad &\Leftrightarrow \quad \omega_{klj} q_{klj} = \frac{n_{kl}}{\sigma_{klj}} q_{klj}^t W^{1/2} C_{kl} W^{1/2} q_{klj} \\ &\Leftrightarrow \quad W^{1/2} C_{kl} W^{1/2} q_{klj} = \frac{\omega_{klj}\sigma_{klj}}{n_{kl}} q_{klj}. \end{aligned}$$

q_{klj} is the eigenfunction of $W^{1/2}C_{kl}W^{1/2}$ which match the eigenvalue $\lambda_{klj} = \frac{\omega_{klj}\sigma_{klj}}{n_{kl}} = W^{1/2}C_{kl}W^{1/2}$. We can write $q_{klj}^t q_{klm} = 0$ if $j \neq m$. So the log-likelihood can be written:

$$\begin{aligned} -2l(\theta) &= \sum_{k=1}^{K}\sum_{l=1}^{L} n_{kl}[d_{kl}log(a_{kl}) + \sum_{j=1}^{d_{kl}} \frac{\lambda_{klj}}{a_{kl}} + (SR - d_{kl})log(b_{kl}) + \sum_{j=d_{kl}+1}^{R} \frac{\lambda_{klj}}{b_{kl}} - 2log(\pi_{kl})] \\ &+ \frac{nSR}{2} log(2\pi), \end{aligned}$$

we substitute the equation $\sum_{j=d_{kl}+1}^{R} \lambda_{klj} = tr(W^{1/2}C_{kl}W^{1/2}) - \sum_{j=1}^{d_{kl}} \lambda_{klj}$:

$$\begin{aligned} -2l(\theta) &= \sum_{l=1}^{L}\sum_{k=1}^{K} n_{kl}[-2log(\pi_{kl}) + d_{kl}log(a_{kl}) + \sum_{j=1}^{d_{kl}} \frac{\lambda_{klj}}{a_{kl}} + (SR - d_{kl})log(b_{kl}) \\ &+ \frac{1}{b_{kl}}(tr(W^{1/2}C_{kl}W^{1/2}) - \sum_{j=1}^{d_{kl}} \lambda_{klj})] + \frac{nSR}{2} log(2\pi) \\ &= \sum_{k=1}^{K}\sum_{l=1}^{L} n_{kl}[d_{kl}log(a_{kl}) + \sum_{j=1}^{d_{kl}} \lambda_{klj}(\frac{1}{a_{kl}} - \frac{1}{b_{kl}}) + (SR - d_{kl})log(b_{kl}) \\ &+ -2log(\pi_{kl}) + \frac{tr(W^{1/2}C_{kl}W^{1/2})}{b_{kl}}] + \frac{nSR}{2} log(2\pi). \end{aligned}$$

In order to minimize $-2l(\theta)$ compared to q_{klj}, we minimize the quantity $\sum_{k=1}^{K}\sum_{l=1}^{L} n_{kl} \sum_{j=1}^{d_{kl}} \lambda_{klj}(\frac{1}{a_{kl}} - \frac{1}{b_{kl}})$ compared to λ_{klj}. Knowing that $(\frac{1}{a_{kl}} - \frac{1}{b_{kl}}) \leq 0$, λ_{kl} has to be as high as feasible. So, the j-th column q_{klj} of matrix Q_{kl} is estimated by the eigen function associated to the j-th highest eigenvalue of $W^{1/2}C_{kl}W^{1/2}$.

A4. Parameter a_{kl} update:

Partial derivative of $l(\theta)$ according to a_{kl} correspond to:

$$\begin{aligned}\frac{\partial l(\theta)}{\partial a_{kl}} &= -\frac{1}{2}n_{kl}(\frac{d_{kl}}{a_{kl}} - \sum_{j=1}^{d_{kl}} \frac{q_{kjl}^t W^{1/2} C_{kl} W^{1/2} q_{kjl}}{a_{kl}^2}) \\ &= -\frac{1}{2}n_{kl}(\frac{d_{kl}}{a_{kl}} - \sum_{j=1}^{d_{kl}} \frac{\lambda_{klj}}{a_{kl}^2})\end{aligned}$$

The prerequisite $\frac{\partial l(\theta)}{\partial a_{kl}} = 0$ implies:

$$\begin{aligned}& \frac{\partial l(\theta)}{\partial a_{kl}} = 0 \\ \Leftrightarrow\ & -\frac{1}{2}n_{kl}(\frac{d_{kl}}{a_{kl}} - \sum_{j=1}^{d_{kl}} \frac{\lambda_{klj}}{a_{kl}^2}) = 0 \\ \Leftrightarrow\ & \frac{n_{kl}d_{kl}}{a_{kl}} = \frac{n_{kl}}{a_{kl}^2} \sum_{j=1}^{d_{kl}} \lambda_{klj} \\ \Leftrightarrow\ & a_{kl} = \frac{1}{d_{kl}} \sum_{j=1}^{d_{kl}} \lambda_{klj}\end{aligned}$$

with λ_{kl} the eigen values of block kl.

A5. Parameter b_{kl} update:

Partial derivative of $l(\theta)$ according to b_{kl} correspond to:

$$\frac{\partial l(\theta)}{\partial b_{kl}} = -\frac{1}{2}[\frac{SR - d_{kl}}{b_{kl}} - \sum_{j=d_{kl}+1}^{SR} \frac{q_{klj}^t W^{1/2} C_{kl} W^{1/2} q_{klj}}{b_{kl}^2}]$$

The prerequisite $\frac{\partial l(\theta)}{\partial b_{kl}} = 0$ implies:

$$-\frac{1}{2}[\frac{SR - d_{kl}}{b_{kl}} - \sum_{j=d_{kl}+1}^{SR} \frac{q_{klj}^t W^{1/2} C_{kl} W^{1/2} q_{klj}}{b_{kl}^2}] = 0$$

$$\Leftrightarrow \frac{SR - d_{kl}}{b_{kl}} = \sum_{j=d_{k+1}}^{SR} \frac{q_{klj}^t W^{1/2} C_{kl} W^{1/2} q_{klj}}{b_k^2}$$

$$b_{kl} = \frac{1}{SR - d_{kl}}[tr(W^{1/2} C_{kl} W^{1/2}) - \sum_{j=1}^{d_{kl}} \lambda_{klj}]$$

R-package for co-clustering algorithm: R-package containing code to perform all analyses described in the article. (GNU zipped tar file)

References

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control 9*, 716–723.

Benitez, I., J.-L. Dez, A. Quijano, and I. Delgado (2016). Dynamic clustering of residential electricity consumption time series data based on hausdorff distance. *Electric Power Systems Research 140*, 517 – 526.

Benitez, I., A. Quijano, J.-L. Dez, and I. Delgado (2014). Dynamic clustering segmentation applied to load profiles of energy consumption from spanish customers. *International Journal of Electrical Power Energy Systems 55*, 437 – 448.

Bhatia, P., S. Iovleff, and G. Govaert (2014, December). blockcluster: An R Package for Model Based Co-Clustering. working paper or preprint.

Biernacki, C., G. Celeux, and G. Govaert (2000). Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE Trans. PAMI 22*, 719–725.

Bouveyron, C., L. Bozzi, J. Jacques, and F.-X. Jollois (2017, December). The Functional Latent Block Model for the Co-Clustering of Electricity Consumption Curves. *Journal of the Royal Statistical Society: Series C Applied Statistics*.

Bouveyron, C., G. Celeux, T. B. Murphy, and A. E. Raftery (2019). *Model-Based Clustering and Classification for Data Science: With Applications in R*. Cambridge University Press.

Bouveyron, C., E. Côme, and J. Jacques (2015). The discriminative functional mixture model for a comparative analysis of bike sharing systems. *Annals of Applied Statistics*, in press.

Chamroukhi, F. and C. Biernacki (2017, July). Model-Based Co-Clustering of Multivariate Functional Data. In *ISI 2017 - 61st World Statistics Congress*, Marrakech, Morocco.

Corneli, M., C. Bouveyron, and P. Latouche (2019, January). Co-Clustering of ordinal data via latent continuous random variables and a classification EM algorithm. working paper or preprint.

Gouveia, J., J. Seixas, S. Luo, N. Bilo, and A. Valentim (2015, 06). Understanding electricity consumption patterns in households through data fusion of smart meters and door to door surveys. pp. ECEEE SUMMER STUDY PROCEEDINGS.

Govaert, G. and M. Nadif (2013). *Co-Clustering* (1st ed.). Wiley-IEEE Press.

Jacques, J. and C. Biernacki (2018, July). Model-Based Co-clustering for Ordinal Data. *Computational Statistics and Data Analysis 123*, 101–115.

Jacques, J. and C. Preda (2014a). Functional data clustering: a survey. *Advances in Data Analysis and Classification 8*(3), 231–255.

Jacques, J. and C. Preda (2014b). Model based clustering for multivariate functional data. *Computational Statistics and Data Analysis 71*, 92–106.

Keribin, C., G. Govaert, and G. Celeux (2010). Estimation d'un modèle à blocs latents par l'algorithme SEM. In *42èmes Journées de Statistique*, Marseille, France, France.

Laclau, C., I. Redko, B. Matei, Y. Bennani, and V. Brault (2017, August). Co-clustering through Optimal Transport. In *34th International Conference on Machine Learning*, Volume 70 of *Proceedings of the 34th International Conference on Machine Learning*, Sydney, Australia, pp. 1955–1964. Proceedings of Machine Learning Research.

Nadif, M. and G. Govaert (2008). Algorithms for model-based block gaussian clustering. In *Proceedings of The 2008 International Conference on Data Mining, DMIN 2008, July 14-17, 2008, Las Vegas, USA, 2 Volumes*, pp. 536–542.

Petersen, K. B. and M. S. Pedersen (2012, nov). The matrix cookbook. Version 20121115.

Ramsay, J. O. and B. W. Silverman (2005). *Functional data analysis* (Second ed.). Springer Series in Statistics. New York: Springer.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association 66*(336), 846–850.

Schmutz, A., J. Jacques, C. Bouveyron, L. Cheze, and P. Martin (2018, July). Clustering multivariate functional data in group-specific functional subspaces. working paper or preprint.

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics 6*(2), 461–464.

Selosse, M., J. Jacques, and C. Biernacki (2019, October). Model-based co-clustering for mixed type data. *Computational Statistics and data analysis*.

Slimen, Y. B., S. Allio, and J. Jacques (2018). Model-Based Co-clustering for Functional Data. *Neurocomputing 291*, 97–108.

Tureczek, A., P. Nielsen, and H. Madsen (2018, 04). Electricity consumption clustering using smart meter data. *Energies 11*, 859.

Zhou, K., C. Yang, and J. Shen (2017). Discovering residential electricity consumption patterns through smart-meter data mining: A case study from china. *Utilities Policy 44*, 73 – 84.

2 Etude de cas : application à la locomotion du cheval au cours du temps

Cet exemple ne correspond pas au système de mesure présenté en introduction. Dans le futur une base de données sera constituée à partir des données issues de la selle iJump. Les données présentées dans cette partie sont issues d'un autre système de mesure interne à l'entreprise composé de plusieurs capteurs IMUs dont un positionné sur le garrot et un sur chaque membre. Pour illustrer la capacité de notre algorithme à identifier des groupes homogènes de locomotion, une base de données a été constituée à l'aide d'un cheval mené en main (non monté) aux trois allures (pas, trot et galop) sur trois types de sol (asphalte, sable et herbe) selon trois conditions (ligne droite, cercle à main gauche, cercle à main droite). Un total de 38 foulées a ainsi été mesuré pour chacune des 18 combinaisons différentes, le galop n'ayant pas été mesuré en ligne droite ni sur l'asphalte.

Nous souhaitons voir si notre algorithme va arriver à différencier les foulées en fonction des différentes conditions. On se demande, de plus, si ce cheval a une locomotion différente ou un potentiel inconfort en fonction du type de sol.

Pour cela, les données ont été découpées en foulées en détectant la protraction maximale sur la vitesse angulaire Gy du capteur attaché sur la face dorsale du canon. Lors de la protraction maximale, un pic apparait nettement sur ce capteur lié au fait que le sens de rotation du membre change. Tous les signaux ont ainsi été découpés à partir des timestamps relevés sur Gy. Puis, chaque foulée a été ré-échantillonnée de façon à avoir 100 points par foulée pour comparer, au sein de notre algorithme, des évènements identiques. La base de données a donc la forme suivante : en ligne les foulées, en colonne chaque passage et à l'intersection d'une ligne et d'une colonne les six signaux collectés par l'IMU (cf. Figure 6.1 où seule l'accélération Ax a été représentée pour faciliter la lisibilité du graphe). L'algorithme funLBM a été appliqué sur cette base de données multivariée avec l'initialisation funFEM, et une recherche de 2 à 6 clusters ligne et colonne.

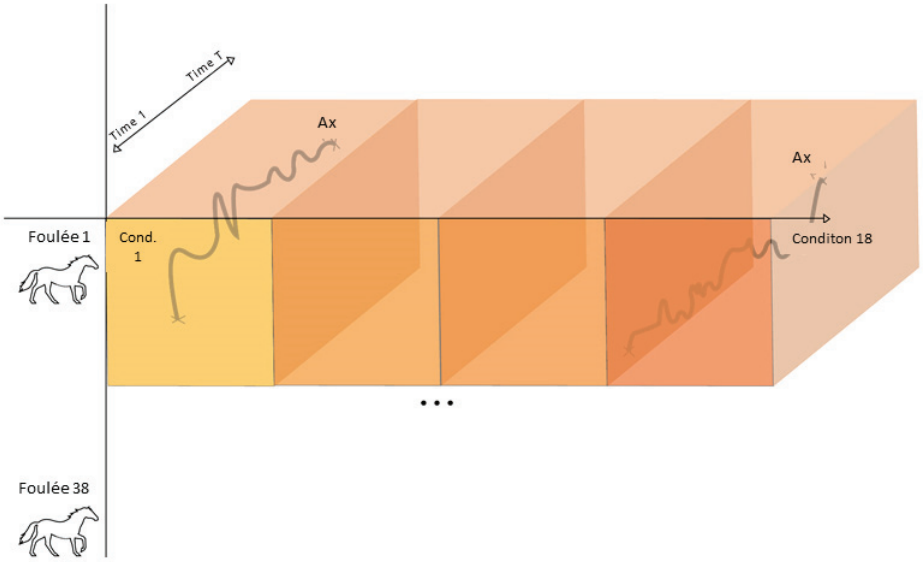


Figure 6.1: Format de la base de données de l'accélération Ax pour application du co-clustering

La meilleure partition est choisie à l'aide de l'ICL, c'est la partition en 2 clusters lignes

et 3 clusters colonne qui est conservée (cf. Table 6.1).

K	L	ICL
2	3	-295478.0
2	4	-296424.1
3	3	-303206.7
2	2	-304977.9
3	4	-305336.4
3	2	-310140.9
4	3	-311414.6
4	2	-315322.5

Table 6.1: Huit premières valeurs d'ICL

Lorsque l'on caractérise les clusters colonne, on observe que ces trois clusters correspondent aux trois allures quel que soit le type de sol ou la figure réalisée (cf. Table 6.2).

Pour caractériser les deux clusters ligne, nous avons représenté les fonctions moyennes de chaque variable par bloc aux Figures 6.2 et 6.3. Ces clusters sont assez déséquilibrés avec 13 foulées dans le premier cluster et 25 dans le second. Ces deux groupes diffèrent par la hauteur de certains pics. En effet, on observe que pour l'accélération Ax au trot, les deux groupes diffèrent par la taille du premier creux et du deuxième pic. Pour le pas, c'est la taille des deux premiers creux qui diffère. Pour finir, pour le galop, les deux groupes diffèrent principalement par l'absence du double-pic final dans le cas du deuxième groupe. Lorsque l'on regarde les trois allures, on peut noter que les différences entre les deux groupes lignes sont plus marquées pour le galop que pour les autres allures. D'un point de vue biomécanique, nous ne savons pas précisément à quel évènement correspondent ces différences de pic car nous n'avons pas filmé le cheval sur les différents passages. Par la suite, pour arriver à caractériser plus finement les clusters, il pourrait être intéressant de filmer le cheval en mouvement pour pouvoir mettre en rapport les signaux analysés et la vidéo afin de voir précisement à quels évènements correspondent les différences détectées entre les deux groupes : est-ce que les foulées du premier groupe sont liées à un inconfort? à un déséquilibre? à une défense contre l'exercice?

Pour conclure, notre modèle de co-clustering a permis de détecter parfaitement les trois allures du cheval de façon non supervisée. En revanche, nous ne sommes pas parvenus pour le moment à caractériser les deux groupes de foulées obtenus pour chaque allure. Néanmoins, notre modèle devrait permettre, couplé à un suivi vidéo, de mettre en évidence d'une part, des périodes pour lesquelles la locomotion des chevaux était "identique" et d'autre part, des périodes d'éventuels inconforts pouvant permettre de prévenir des boiteries, symptôme le plus répandu chez le cheval de sport.

Allure	Sol	Figure	Cluster
Pas	Asphalte	Ligne droite	2
Trot	Asphalte	Ligne droite	1
Galop	Sable	Cercle main droite	3
Galop	Sable	Cercle main gauche	3
Pas	Sable	Cercle main droite	2
Pas	Sable	Cercle main gauche	2
Pas	Sable	Ligne droite	2
Trot	Sable	Cercle main droite	1
Trot	Sable	Cercle main gauche	1
Trot	Sable	Ligne droite	1
Galop	Terre	Cercle main droite	3
Galop	Terre	Cercle main gauche	3
Pas	Terre	Cercle main droite	2
Pas	Terre	Cercle main gauche	2
Pas	Terre	Ligne droite	2
Trot	Terre	Cercle main droite	1
Trot	Terre	Cercle main gauche	1
Trot	Terre	Ligne droite	1

Table 6.2: Caractérisation de la partition colonne obtenue par funLBM

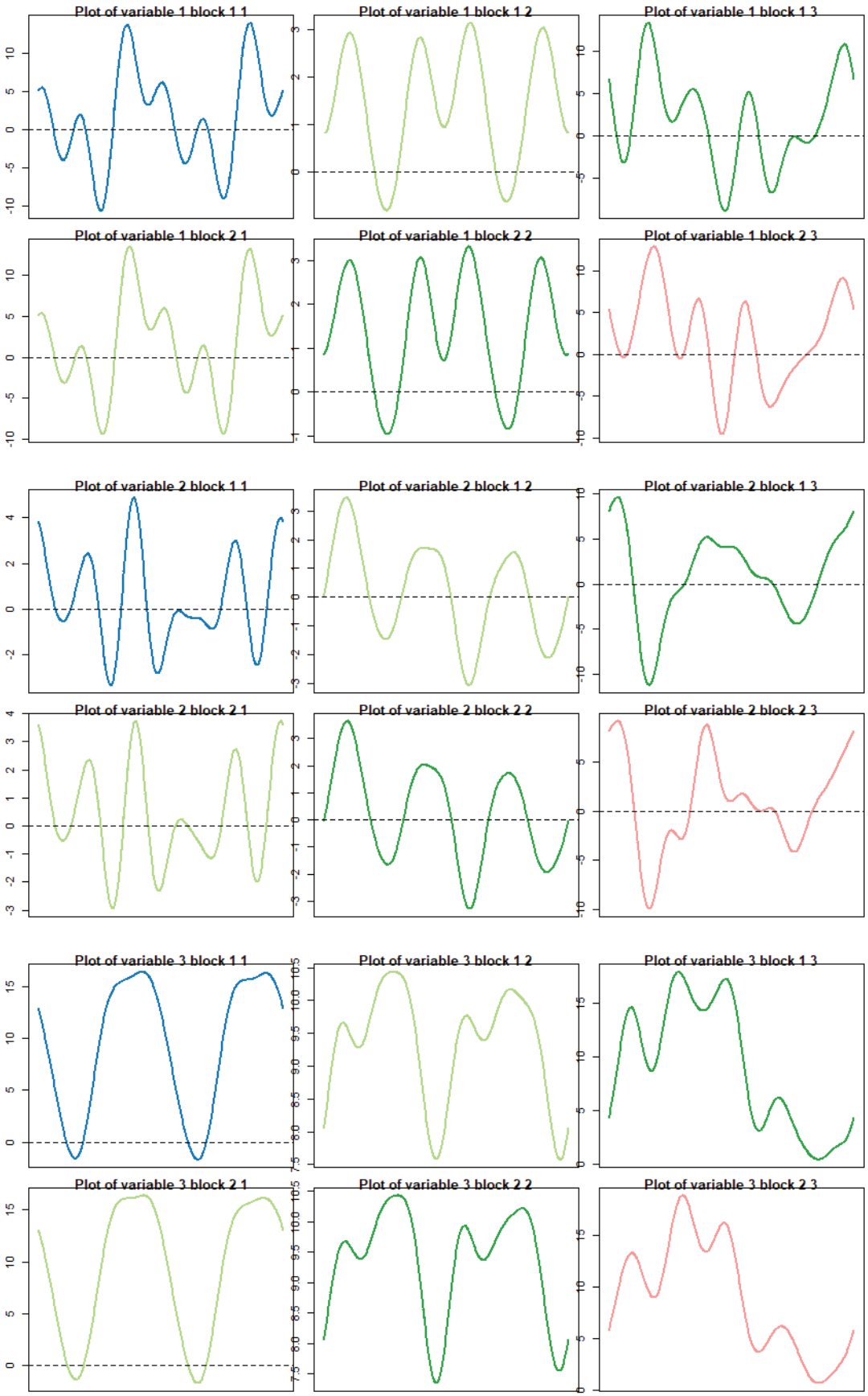


Figure 6.2: Fonction moyenne des blocs pour les trois accélérations Ax, Ay et Az

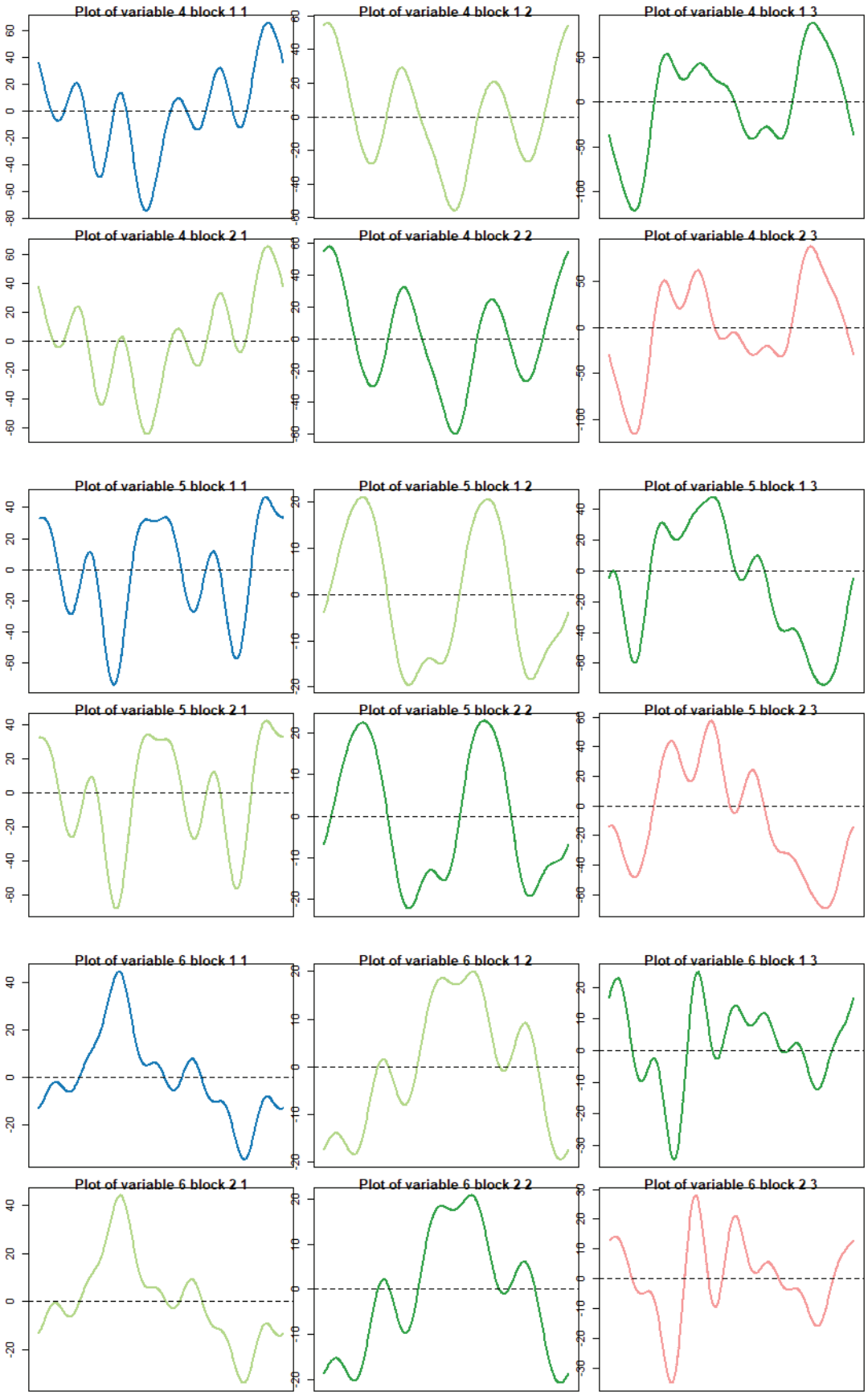


Figure 6.3: Fonction moyenne des blocs pour les trois vitesses angulaires Gx, Gy et Gz

Part II

Prédiction de la qualité de saut

Le dernier objectif de cette thèse était de pouvoir caractériser et prédire la qualité de saut à partir des signaux collectés par la selle. Dans un premier temps nous allons faire un rapide état des lieux des méthodes permettant de prédire une variable réponse ordinale à partir de variables explicatives quantitatives. Puis nous verrons présenterons l'application de ces méthodes sur la base de données à notre disposition.

Chapitre 7

Etat des lieux des modèles de régression ordinale

Une variable ordinale est une variable qualitative pour laquelle la valeur mesurée pour chaque individu possède un ordonnancement clair des niveaux, chaque modalité utilisée faisant référence à une plus grande ou à une plus petite magnitude d'une certaine caractéristique. Une telle variable est différente des variables qualitatives, qui sont mesurées sur une échelle nominale, et qui possèdent des catégories qui ne sont pas liées à différentes magnitudes d'une caractéristique.

A ce jour aucune méthode n'a été développée spécifiquement pour prédire une variable réponse ordinale à partir de plusieurs variables explicatives fonctionnelles. Nous allons présenter ici des méthodes développées pour le cas de données en grande dimensions que l'on peut appliquer à nos données si on les considère comme un vecteur de points multidimensionnels (cf. Chapitre 1).

1 Modèle de régression logistique basé sur les logit additifs

Quand la variable réponse est ordinale, on peut construire un modèle logit de façon à conserver l'ordre des catégories. Une manière de le faire est de regrouper les catégories qui sont voisines sur l'échelle ordinale. Par exemple, on peut appliquer une transformation logit aux probabilités cumulées. Il existe de nombreux autres modèles logit comme les logit pour catégories adjacentes ou les logit pour ratio continu [Agresti, 2010] mais nous ne présenterons ici que le logit additif.

Pour G catégories de la variable résultat, de probabilité $\pi_1, ..., \pi_G$, les logits cumulatifs se définissent de la façon suivante :

$$\begin{aligned} logit[P(Y \leq j)] &= log\frac{P(Y \leq j)}{1 - P(Y \leq j)} \\ &= log\frac{\pi_1 + ... + \pi_j}{\pi_{j+1} + ... + \pi_G}, j = 1, ..., G \end{aligned}$$

où $\pi_j = P(Y = j)$.

Pour un sujet i, soit y_i la variable réponse et x_i un vecteur colonne des valeurs des

variables explicatives. Le modèle s'écrit :

$$logit[P(y_i \leq j)] = \alpha_j + \beta' x_i = \alpha_j + \beta_1 x_{i1} + ... + \beta_j x_{ij}, j = 1, ..., G-1 \tag{7.1}$$

où $\beta = (\beta_1, ..., \beta_j)$ un vecteur colonne des paramètres qui décrit les effets des variables explicatives avec $\beta_1, ..., \beta_j \in \mathbb{R}$ et $x_i = (x_{i1}, ..., x_{ip})$ avec $x_{i1}, ..., x_{ip} \in \mathbb{R}$

L'expression du modèle s'écrit également :

$$P(Y \leq j) = \frac{exp(\alpha_j + \beta' x)}{1 + exp(\alpha_j + \beta' x)}, j = 1, ..., G-1$$

et on en déduit

$$P(Y = j) = \frac{exp(\alpha_j + \beta' x)}{1 + exp(\alpha_j + \beta' x)} - \frac{exp(\alpha_{j-1} + \beta' x)}{1 + exp(\alpha_{j-1} + \beta' x)}.$$

Une façon possible d'estimer les paramètres du modèle est d'utiliser le modèle de maximum de vraisemblance. Le choix des covariables à inclure dans le modèle peut se faire en ajoutant des termes et en regardant si la modélisation s'améliore ou non, ou à l'aide de critères numériques tels que l'AIC.

Le package *ordinal* [Christensen, 2018] propose de nombreux modèles logistiques.

2 Arbres de classification pour une réponse ordinale

Soit un jeu de données $\{(y_i, x_{i1}, ..., x_{ip}); i = 1, ..., n\}$ qui contient les valeurs d'une variable réponse Y à G catégories ordonnées et p variables explicatives $X_1, ..., X_p$ observées sur les n échantillons.

Quand on construit un arbre de classification, toutes les observations commencent ensembles au noeud racine t. Puis, pour chacune des p variables explicatives, la meilleure partition binaire est déterminée. Les partitions produisent un nombre croissant de noeuds homogènes du point de vue du nombre de classes désiré. Par la suite, parmi toutes les partitions basées sur les p variables explicatives, la meilleure partition est choisie pour partitionner les observations en des noeuds descendants gauche et droit, notés t_L et t_R. Ce processus est répété pour tous les noeuds descendants jusqu'à ce que les noeuds terminaux soient homogènes en ce qui concerne la classe ou qu'il y ait trop peu d'observations pour pouvoir réaliser une nouvelle partition.

Quel que soit le type d'arbre de classification utilisé (CART, C4.5, ...), quatre éléments essentiels sont nécessaires pour la procédure de croissance de l'arbre : un ensemble de questions binaires qui constituent l'ensemble candidat de partitions s, un critère de "bon" partitionnement qui peut être évalué pour chaque partition ou noeud t, une règle d'arrêt du partitionnement et une règle pour assigner chaque noeud terminal à une des G classes.

Le critère de "bon" partitionnement est le plus souvent formulé comme une décroissance d'impureté du noeud. Soit $p(Y = j|t)$ la proportion d'individus au noeud t qui appartient à la j-ième catégorie de Y, avec $j = 1, ..., G$.

La mesure d'impureté au sein du noeud communément utilisée pour la classification d'une variable nominale est la fonction généralisée d'impureté de Gini définie pour le

noeud t par :

$$i(t) = \sum_{k=1}^{G} \sum_{l=1}^{G} C(Y = k|Y = l)p(Y = k|t)p(Y = l|t),$$

où $C(Y = k|Y = l)$ représente le coût de mauvaise classification, donc d'assigner la catégorie k à un individu qui appartient à la catégorie l. On obtient la fonction nominale d'impureté de Gini en fixant $C(Y = k|Y = l) = 1, \forall k \neq l$.

Pour toute partition binaire s du noeud t, les individus assignés au noeud t sont répartis entre deux noeuds fils, t_R et t_L. Cette répartition résulte de la plus grande décroissance d'impureté du noeud définie par

$$\Delta i(s, t) = p(t)i(t) - p(t_R)i(t_R) - p(t_L)i(t_L).$$

où $p_L = \frac{p(t_L)}{p(t)}$ est une proportion d'individus de t qui vont dans t_L et $p_R = \frac{p(t_R)}{p(t)}$ est une autre proportion de cas qui vont dans t_R et $p_L + p_R = 1$ [Galimberti et al., 2012a, Zhang and Ye, 2008].

Le package *rpartScore* [Galimberti et al., 2012b] permet d'appliquer ce modèle et différentes pénalisations C.

3 Forêts pour réponse ordinale

Nous allons présenter ici trois méthodes différentes disponibles sous la forme de packages R, ces méthodes seront comparées pour la prédiction de la qualité du saut au chapitre 8.

3.1 Forêts aléatoires

Les forêts aléatoires sont utilisées dans de nombreuses applications impliquant des données en grande dimension. Etant une méthode non paramétrique, les forêts aléatoires peuvent gérer la non linéarité, les interactions, les prédicteurs corrélés et l'hétérogénéité. Les forêts aléatoires peuvent être utilisées en classification dans le cas d'une réponse nominale, et en régression dans le cas d'une réponse numérique. En utilisant un ensemble d'arbres de classification et de régression, on peut obtenir des prédictions et identifier les prédicteurs qui sont associés à la réponse via une mesure d'importance de variable interne aux forêts aléatoires [Janitza et al., 2014].

Plusieurs approches ont été dévelopées dans le cas des arbres de régression pour tenir compte de l'ordre au sein de la variable réponse. Elles sont basées sur des mesures d'impureté alternatives à l'index de Gini. Les exemples les plus connus sont la fonction d'impureté ordinale suggérée par [Piccarreta, 2001] et le critère Gini généralisé introduit par [Breiman et al., 1984]. Avec ces mesures, une plus grande pénalité est placée sur la mauvaise classification dans une catégorie qui est plus distante de la vraie classe qu'une mauvaise classification dans une catégorie proche de la vraie classe, prenant ainsi en compte la nature ordinale de la réponse. Le critère de *twoing* ordonné introduit par [Breiman et al., 1984] est une autre mesure populaire qui ne repose pas sur un coût de mauvaise classification mais plutôt sur la réduction du problème de classification en k classes en un problème de $k-1$ classifications en deux classes où la coupure qui divise les k classes en deux classes est uniquement faite entre des catégories adjacentes. Le découpage d'un noeud en deux parties dépend de la maximisation de la fonction de partition.

[Hothorn et al., 2006] proposent une version non biaisée des forêts aléatoires qui fournit la possibilité de prendre en compte l'information d'ordre lors de la construction de l'arbre. Un test statistique est réalisé pour évaluer l'association entre les prédicteurs et la réponse ordinale. Le prédicteur qui fournit la plus faible p-value est utilisé pour réaliser la coupure. Dans ce but, il faut attacher des scores à chaque catégorie de la variable réponse. Ces scores reflètent les distances entre les niveaux de la réponse.

Construction d'arbres par inférence conditionnelle
La méthode des forêts aléatoires est un outil de classification et de régression qui combine plusieurs arbres de décision. Un arbre individuel est modélisé en utilisant un échantillon aléatoire d'observations tiré dans l'échantillon d'origine. Pour chaque division au sein de l'arbre, *mtry* prédicteurs tirés aléatoirement sont testés pour être candidats pour la division et le prédicteur qui produit la meilleure division est choisi. Dans la méthode proposée par [Hothorn et al., 2006], des tests d'inférence conditionnelle sont réalisés pour sélectionner le meilleur découpage de façon non biaisée. Pour chaque division au sein d'un arbre, l'association de chaque prédicteur candidat, issu du sous-ensemble tiré aléatoirement, avec la réponse est testée, produisant une p-value. Le prédicteur avec la plus faible p-value est conservé, et pour ce prédicteur la meilleure partition est conservée. Cette méthodologie utilise un système de tests de permutations et est donc applicable aux problèmes où les prédicteurs et la réponse sont mesurés sur des échelles arbitraires, incluant les variables ordinales. Dans le cas d'une réponse ordinale, la réponse est transformée dans une échelle métrique en attribuant des scores aux niveaux de la réponse. La réponse transformée est alors utilisée pour tester l'association avec les prédicteurs candidats. Soit $s(j) \in \mathbb{R}$ le score pour la catégorie $j \in \{1, ..., G\}$ et Y_i la réponse ordinale de l'observation i avec les covariables X_{ir}, $r = 1, ..., p$, alors le test statistique qui est utilisé pour tester l'association entre la réponse ordinale et la variable prédictrice X_r sur l'échelle arbitraire est défini par :

$$T_j = \sum_{i=1}^{n} g_r(X_{ir}) s(Y_i).$$

Avec $g_j : X_r \to \mathbb{R}^{p_j}$, une transformation non aléatoire de la variable prédictrice X_r de l'espace vectoriel à une dimension à l'espace vectoriel à p_r dimensions. Pour une variable prédictrice numérique la transformation est habituellement la fonction identité. Pour une variable explicative numérique cette transformation est habituellement la fonction identité où $g_r(X_{ir}) = X_{ir}$ et $p_r = 1$ [Janitza et al., 2014].

On peut remarquer que la construction des arbres de régression ordinale a la même structure que celle des arbres de régression, mais que les prédictions sont différentes car les schémas d'aggrégation sont différents et correspondent à ceux des arbres de classification.

Prédiction par forêt aléatoire Une forêt aléatoire combine plusieurs arbres de décision individuels pour fournir une prédiction finale. Pour une observation qui n'a pas été utilisée pour construire la forêt aléatoire, chaque arbre de cette forêt fournit une prédiction. La prédiction finale est alors la moyenne de toutes les prédictions fournies par les arbres. Dans le cas des données ordinales ces prédictions moyennées sont difficiles à interpréter et il n'existe aucune procédure permettant d'obtenir les classes à partir de ces valeurs. Dans ce cas, on peut utiliser les probabilités d'appartenir à la classe, et classer les observations dans la classe à laquelle ils ont le plus de chance d'appartenir : $\widehat{Y} = j \Longleftrightarrow \widehat{p(Y = j)} = max_{l=1,...,G} \widehat{p(Y = l)}$.
La classe prédite correspond alors au mode de la distribution de probabilité de la classe

prédite.

Observations Out-of-bag Chaque arbre est construit à partir d'un échantillon aléatoire des données, les observations non-utilisées sont dites "Out-of-bag". L'erreur de classification doit donc être calculée à partir de ces données "Out-of-bag". Ces données peuvent aussi servir à mesurer l'importance des variables prédictrices.

Le package *party* [Strobl et al., 2008] propose des algorithmes permettant de faire des arbres et des forêts aléatoires.

3.2 Bootstrap aggregation

Cette méthode est un cas particulier des forêts aléatoires où l'on répète p fois l'étape *mtry* où sont tirés aléatoirement un nombre de prédicteurs qui sont testés pour être candidats pour la division.

En effet, il a été montré que les arbres de classification sont instables, c'est à dire que de petits changements de l'échantillon d'entrainement entrainent des changements pouvant être importants dans l'arbre final obtenu. L'aggrégation par bootstrap (ou bagging) permet d'améliorer la stabilité de la prédiction et donc la précision [Archer and Mas, 2009]. Le bagging consiste à générer des échantillons bootstrap des données originelles de façon à produire plusieurs versions d'un prédicteur. Ces prédicteurs sont ensuite combinés pour former un classifieur d'aggrégation. Quand on applique le bagging sur les arbres de classification, un échantillon bootstrap d'entrainement L_b est obtenu en échantillonnant avec remise l'échantillon d'entrainement L. Un arbre très large $T_{max,b}$ croît en utilisant les observations de l'échantillon d'entrainement bootstrap L_b et n'est pas élagué. Pour chaque noeud terminal de $T_{max,b}$ la classe prédite est obtenue. Ce processus d'extraire des échantillons bootstrap et d'en déduire un arbre de classification est répété B fois.

Le classifieur aggrégé est obtenu en combinant les prédictions des B arbres de classification. Dans un premier temps l'indicateur de prédiction spécifique à la classe est dérivé de la i-ème observation et du b-ième échantillon bootstrap d'apprentissage, et pour la classe g on obtient :

$$\hat{d}_{igb}(x_i, L_b) = \begin{cases} 1 \text{ si } \hat{g}_{igb} = g \\ 0 \text{ si } \hat{g}_{igb} \neq g \end{cases}$$

Pour chaque observation i, l'estimateur de bagging est constitué à partir des valeurs issues des B arbres qui prédisent la classe g pour x_i, où ces valeurs sont issues de $\hat{d}_{ig,bag}(x_i, L) = \sum_{b=1}^{B} \hat{d}_{igb}(x_i, L_b)$. La classe prédite du bagging pour la i-ème observation est la classe qui obtient la majorité des votes parmi les B arbres

$$\hat{D}_{i,bag}(x_i, L) = \text{argmax}_g \hat{d}_{ig,bag}(x_i, L).$$

Différents post-traitements ont ensuite été proposés dans la littérature pour prendre la décision finale comme : arrondir la valeur prédite pour obtenir la classe, utiliser la médiane ou le mode ou la moyenne ou arrondir au niveau de chaque noeud lors de la construction de l'arbre. Cette dernière méthode peut être adaptée si la réponse ordinale est basée sur la distribution d'une mesure quantitative. Parfois il n'y a aucun choix évident dans l'attribution de scores numériques à une variable ordinale, et chaque méthode de scoring conduit à des résultats différents. Il existe des méthodes alternatives d'induction d'arbres plus stables comme l'index de Gini ou le critère de *twoing* ordonné. [Archer and Mas, 2009] proposent un nouveau critère spécifique au problème de prédiction d'une réponse ordinale. Les propriétés du critère de partitionnement pour une variable réponse nominale

nécessitent des modifications quand notre objectif est de prédire une réponse ordinale. Plus précisement, le critère de partionnement doit récompenser les partitions pour les variables qui sont liées de façon monotone à la réponse ordinale dans la procédure de croissance de l'arbre. La question fondamentale pour formuler le critère de partition est obtenue en répondant à "Quelle distribution des cas entre deux noeuds fils maximise le critère de goodness of fit?". Les auteurs proposent une nouvelle version de la fonction d'impureté pour le cas ordinal qui ignore la différence entre les scores des classes ordinales :

$$\sum_{g=1}^{G} F_t(g)[1 - F_t(g)],$$

avec $F_t(g)$ la proportion d'observations du noeud t.

Cette formule fournit un index qui permet de mesurer la force de l'association entre une réponse ordinale et une covariable ordinale.

Le package *rpart* [Therneau and Atkinson, 2018] permet d'appliquer cette méthode.

3.3 Forêt ordinale

[Hornung, 2019a] propose une méthode qui, contrairement aux forêts aléatoires décrites précédemment, utilise l'estimation des erreurs "out-of-bag" pendant la construction de la forêt .

Une méthode simple de prédiction basée sur les forêts pour une variable réponse ordinale serait de considérer une forêt de régression en utilisant les valeurs des classes $1, ..., G$ de la variable réponse. Mais cette proposition est sous-optimale car la taille des classes peut différer d'une classe à l'autre. Dans le modèle proposé ici, la taille des classes correspond à la taille des G intervalles adjacents dans la plage d'une variable réponse sous-jacente continue. La seule hypothèse de ce modèle est que, sous-jacente à la variable ordinale observée y, il existe une variable continue raffinée y^* qui détermine les valeurs de la variable ordinale. La relation entre la variable continue y^* et y est telle que plus la valeur de y^* est grande pour une observation, plus la classe de la variable réponse ordinale est grande pour cette observation. Plus précisément, si y^* est dans le j-ième intervalle des G intervalles adjacents, y prendra la valeur j.

Dans le cas des forêts ordinales, les frontières des intervalles de y^*, correspondant aux différentes classes de y, sont estimées ou optimisées en maximisant le performance de prédiction des "out-of-bag" (OOB) des forêts de régression. L'utilisation des valeurs des scores qui correspondent à ces intervalles de classes optimisés, au lieu d'utiliser les valeurs des classes $1, ..., G$, permet d'améliorer la performance de prédiction.

Construction des règles de prédiction des Forêts Ordinales Supposons un échantillon $\{(x_1, y_1), ..., (x_n, y_n)\}$, où x_i, $i \in \{1, ..., n\}$ désigne le vecteur des covariables de l'observation i et $y_i \in \{1, ..., G\}$ la valeur de la classe de la variable réponse ordinale pour cette observation. La règle de prédiction est construite comme suit pour $b = 1, ..., B_{sets}$ (par exemple $B_{sets} = 1000$) :

- Tirer $G - 1$ cas d'une variable aléatoire distribuée selon une loi uniforme $U(0, 1)$ et trier ces valeurs alors notées $d_{b,2}, ..., d_{b,G}$. De plus, on fixe $d_{b,1} = 0$ et $d_{b,G+1} = 1$.

- Créer une variable réponse continue $z_b = z_{b,1}, ..., z_{b,n}$ en remplaçant chaque valeur de la classe j dans la variable réponse ordinale $y = y_1, ..., y_n$ par la j-ième valeur de l'ensemble des scores $s_b = \{s_{b,1}, ..., s_{b,G}\}$ où $s_{b,j} := \phi^{-1}(c_{b,j})$ et $c_{b,j} = (d_{b,j} + d_{b,j+1})/2$.

- Faire croitre une forêt de régression f_{s_b} avec $B_{ntreeprior}$ arbres (par exemple $B_{ntreeprior} = 100$) en utilisant z_b comme variable réponse.
- Obtenir les prédictions OOB $\hat{z}_{b,1}, ..., \hat{z}_{b,n}$ de $z_{b,1}, ..., z_{b,n}$.
- Obtenir les prédictions OOB de $y_1, ..., y_n$ comme suit : $\hat{y}_{b,i} = j$ si $\hat{z}_{b,i} \in]\phi^{-1}(d_{b,j}), \phi^{-1}(d_{b,j+1})]$, $i \in \{1, ..., n\}$.
- Assigner un score de performance $sc_b = g(y, \hat{y}_b)$ à f_{sb} où $\hat{y}_b = \hat{y}_{b,1}, ..., \hat{y}_{b,n}$ et g une fonction spécifique appelée fonction de performance. Le choix de fonction dépend du contexte. Il existe plusieurs variantes de la fonction de performance g. Dans la plupart des situations, on souhaite classer les observations de chaque classe avec la même précision, indépendemment de la taille des classes. Dans d'autres situations le but principal est de classer correctement le plus d'observations possibles, ce qui implique de prioriser les plus grandes classes au détriment des plus petites classes afin de favoriser la précision de classification. Parfois, certaines classes spécifiques ont une importance particulière, ce qui peut être priorisé par l'algorithme de forêt ordinale.

Soit S_{bset} l'ensemble des indices des $B_{bestsets}$ (par exemple $B_{bestsets} = 10$) forêts de régression construites à l'étape précédente qui inclut les plus grandes valeurs sc_b. Alors, pour chaque $j \in \{1, ..., J+1\}$, on calcule la moyenne des valeurs $d_{b,j}$ pour lesquelles $b \in S_{best}$. Ce qui résulte en un ensemble de $J+1$ valeurs notées $d_1, ..., d_{J+1}$. On forme ainsi une nouvelle variable réponse continue $z = z_1, ..., z_n$ en remplaçant chaque valeur de classe j de la variable réponse ordinale y par la j-ième valeur dans l'ensemble des scores optimisés $\{s_1, ..., s_J\}$, où $s_j = \phi^{-1}(c_j)$ et $c_j = (d_j + d_{j+1})/2$. On fait grandir une forêt de régression f_{final} avec B_{ntree} arbres (par exemple $B_{ntree} = 5000$) en utilisant z comme variable réponse.

Prédiction en utilisant la forêt ordinale Une prédiction de la valeur de la variable réponse d'une observation indépendante i^* basée sur son vecteur de covariable x_{i^*} est obtenue de la façon suivante pour $b = 1, ..., B_{ntree}$: on applique le b-ième arbre sur f_{final} pour l'observation i^* afin d'obtenir une prédiction $\hat{z}_{i^*,b}$. Puis on calcule une prédiction de la classe pour le b-ième arbre : $\hat{y}_{i^*,b} = j$ si $\hat{z}_{i^*,b} \in]\phi^{-1}(d_j), \phi^{-1}(d_{j+1})]$. Et enfin, on calcule la prédiction de la classe finale à partir de $\hat{y}_{i^*,1}, ..., \hat{y}_{i^*,B_{ntree}}$ par la majorité de vote.

[Hornung, 2019a] propose une formule permettant de mesurer l'importance d'une variable pour la forêt ordinale. Le package *ordinalForest* [Hornung, 2019b] permet l'utilisation de cet algorithme de partitionnement.

Chapitre 8

Application à la prédiction de la qualité du saut d'obstacles

L'objectif de notre étude est de classer les sauts d'après leur qualité à l'aide d'une échelle de notes allant de 1 (très mauvais) à 5 (excellent). Notre objectif est d'arriver à les caractériser en fonction des signaux collectés par la selle lors de parcours de saut d'obstacles et à prédire leurs valeurs lorsque de nouvelles données seront collectées. Il a ainsi été demandé à des juges amateurs d'attribuer une note à chaque phase de saut pour différents obstacles de parcours réalisés par différents chevaux et cavaliers de niveau professionnel.

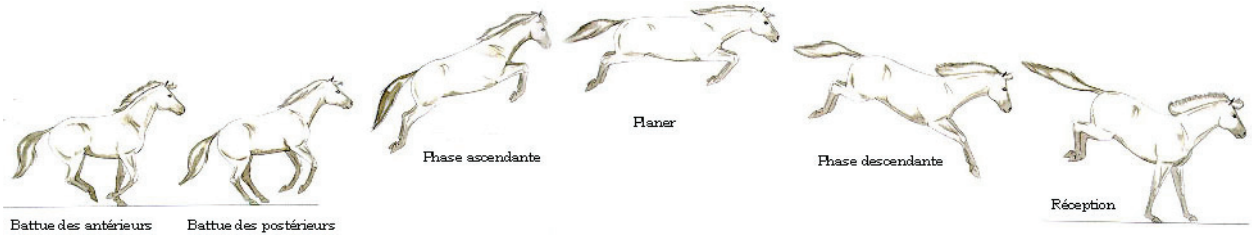


Figure 8.1: Les différentes phases de saut du cheval.

La base de données a été constituée en décomposant chaque saut en trois phases (cf. Figure 8.1). La phase d'abord correspond à deux foulées de galop avant la battue des antérieurs jusqu'à la battue des antérieurs. La phase de planer, entre la battue des antérieurs et la réception. Pour finir, la phase de réception qui va de la réception à deux foulées après. Chaque phase est ensuite ré-échantillonnée sur 2 secondes (donc 200 points pour chaque signal) pour avoir le même nombre de points pour chaque individu. A cela s'ajoute 26 paramètres calculés à partir des signaux collectés par l'IMU comme l'amplitude de certains pics du signal, le temps des phases et différentes énergies.

La répartition des notes est présentée à la Table 8.1. On peut voir que notre base de données de notes est totalement déséquilibrée : on a entre 4 fois et 25 fois plus de notes moyennes (3 et 4) que d'autres notes, ce qui risque de complexifier l'analyse.

Table 8.1: Répartition des notes dans la base de données générale

Note	1	2	3	4	5
Occurence	32	271	862	824	107

1 Etude de l'intégralité de la base de données

1.1 Analyse de la variance fonctionnelle

Avant de chercher à prédire la qualité du saut à partir des signaux collectés, nous souhaitons vérifier qu'il y a une différence significative entre les signaux collectés par la selle pour chaque note donnée par les juges. Pour cela, on applique une analye de la variance (anova) fonctionnelle [Cuevas et al., 2004] qui correspond à une version asymptotique de l'anova où une procédure de Monte Carlo est proposée pour calculer cette distribution asymptotique. Pour réaliser ces analyses, nous avons utilisé la fonction *anova.onefactor* du package fda.usc [Febrero-Bande and Oviedo de la Fuente, 2012].

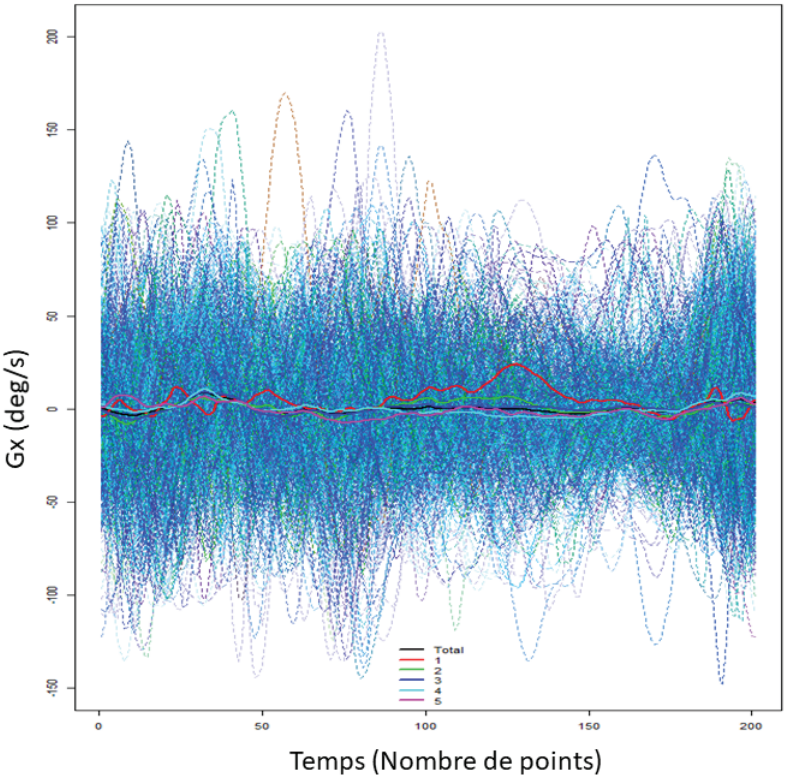


Figure 8.2: Fonctions moyennes de Gx pour chaque note. Trait noir : moyenne toutes notes confondues, rouge : note 1, vert : note 2, bleu : note 3, turquoise : note 4, rose : note 5

On obtient un effet significatif sur la note de toutes les accélérations sauf l'accélération Ay et les vitesses angulaires Gy et Gz (cf. Table 8.2 et Figures B.1,B.2, B.3, B.4, B.5, B.6 visibles en Annexe). Un exemple de fonction moyenne selon chaque note est représenté à la Figure 8.2 pour la vitesse angulaire Gx. On observe que la note 1 a un signal moyen très différent des signaux moyens des autres notes. Afin d'être sûr que ce ne soit pas ces très mauvais sauts qui induisent la différence significative, la même analyse est réalisée en retirant la note 1 de la base de données. De ce fait, on obtient les mêmes résultats sauf pour l'accélération Az qui devient non significative et la vitesse angulaire Gz qui devient significative. Ces variables avaient une probabilité critique proche de 0.05 lorsque l'on tenait compte de la note 1, ce qui peut expliquer cette inversion de significativité.

L'anova a donc permis de mettre en évidence qu'il y a bien des différences entre la plupart des signaux collectés pour chaque note.

Variable	P-value analyse globale	P-value sans la note 1
Ax	0	0
Ay	Non convergence	0.22
Az	0.04	0.06
Gx	0.04	0.02
Gy	0.3	0.08
Gz	0.06	0.02

Table 8.2: P-values associées à l'anova fonctionnelle avec toutes les notes (au centre) et sans tenir compte de la note 1 (à droite)

1.2 Analyse en composantes principales

Afin d'observer la répartition des individus en fonction des signaux collectés et des paramètres calculés, une ACP est réalisée sur l'ensemble de la base de données. L' ACP est réalisée à l'aide du package FactoMineR [Lê et al., 2008] avec la note en illustratif et les paramètres calculés et les variables mesurées pour chaque saut en actif en les prenant comme un vecteur de points multidimensionnel. On peut voir sur le graphe des individus (Figure 8.3) que l'on ne distingue pas de groupe net en fonction des notes, toutes les couleurs sont en effet mélangées. De plus, on peut noter que le pourcentage de variance expliqué par le premier plan factoriel est uniquement 22%, ce qui est plutôt faible pour un premier plan et qui laisse supposer que même si les signaux mesurés et les variables calculées apportent de l'information, les principales sources de variabilité ne seraient pas prises en compte dans notre analyse.

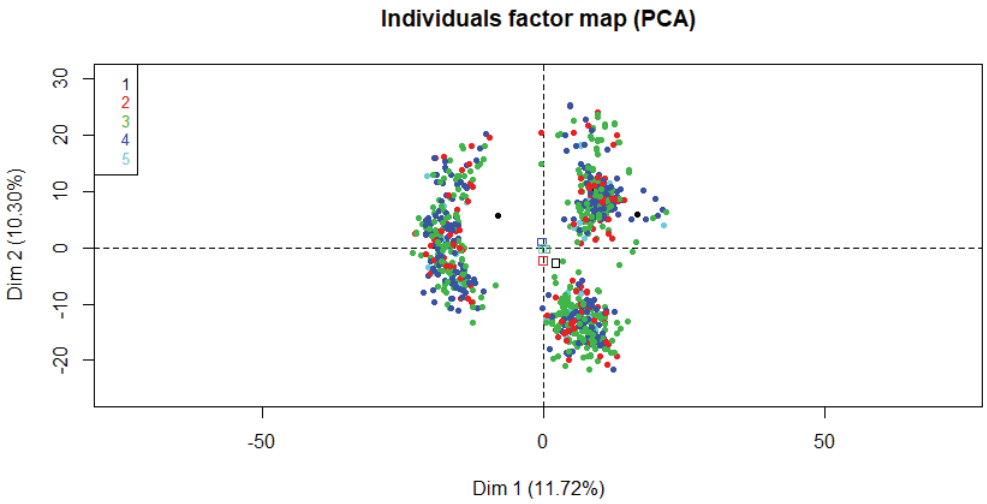


Figure 8.3: Graphique des individus de l'ACP coloré par note

La Figure 8.4 permet de mettre en évidence que les 10 variables qui ont le plus contribué à la construction du premier axe factoriel sont les derniers points du signal de vitesse angulaire Gy. Lorsque l'on croise ce graphe avec celui des individus, on peut noter

que le groupe d'individus à gauche se distingue des deux autres groupes par des fortes valeurs de vitesse angulaire Gy pour les derniers points du signal (156 à 164). Il serait intéressant de croiser ces résultats avec la vidéo du parcours afin de comprendre à quel évènement précis correspondent ces points pour chaque phase de saut.

Pour finir, la représentation des modalités de la variable Note avec leurs ellipses de confiance nous permet d'observer que la note 1 est celle avec la position la plus variable (avec la plus grande ellipse de confiance), ce qui est sans doute lié à la très faible proportion de cette modalité dans notre base de donnée. Les ellipses de toutes les notes se chevauchent, il y a donc une confusion totale entre toutes les notes.

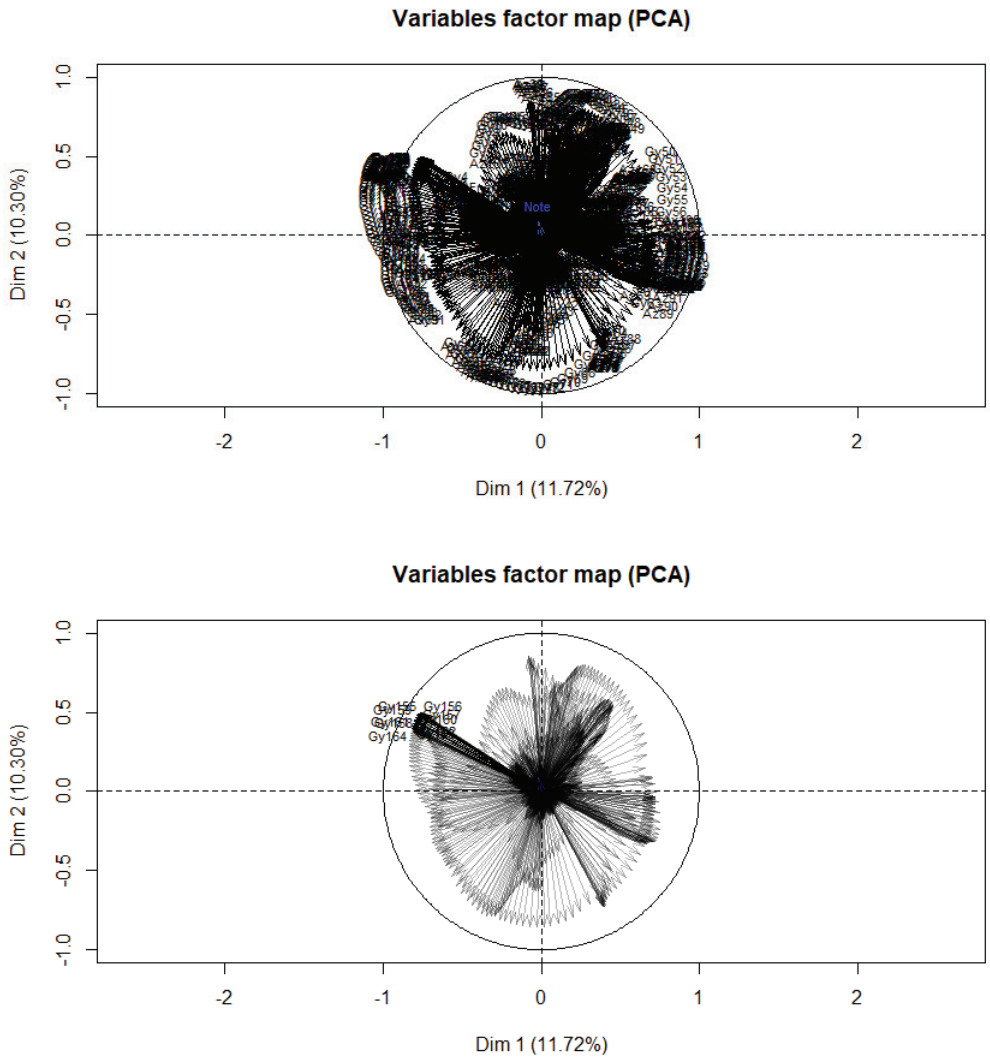


Figure 8.4: Graphique des variables actives de l'ACP (en haut) et graphique des 10 premières variables actives qui ont le plus contribué à la création du premier plan factoriel (en bas)

Pour conclure, il sera sans doute difficile de créer un modèle prédictif qui permettra une distinction plus fine que les très bons sauts versus les mauvais sauts. Or, ce type d'information est ressenti de manière évidente même pour un cavalier, même non averti, et cela limite donc l'intérêt d'un tel modèle. Les trois phases de saut représentant des évènements très différentes, et ayant donc des formes de signaux très différents, nous allons

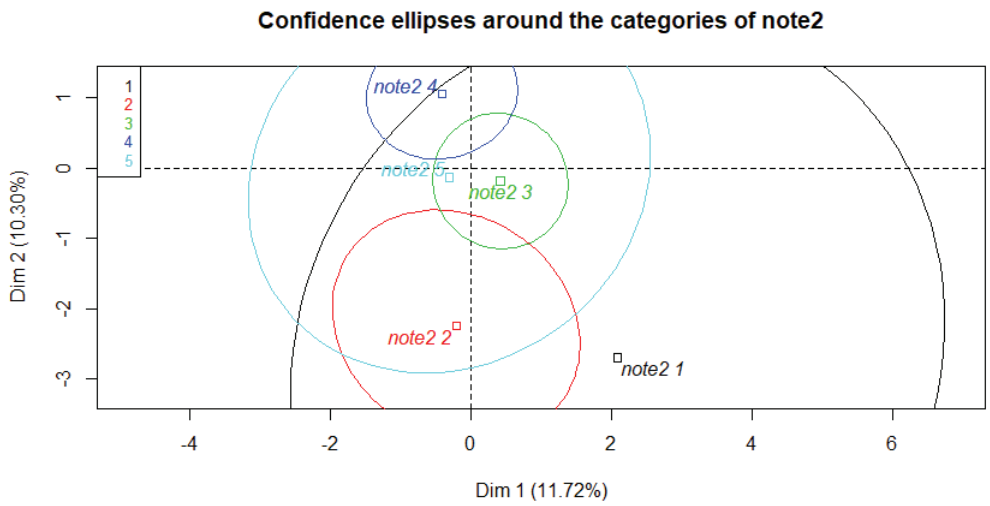


Figure 8.5: Représentation des modalités de la variable Note (appelée note2) avec leur ellipse de confiance

voir si l'analyse de chaque phase prise séparément permet de mieux mettre en évidence des différences entre les notes.

2 Analyse de chaque phase de saut séparément

Suite à l'analyse de la base de données au global, nous nous sommes interrogés sur l'impact de confondre les trois phases de saut alors qu'elles présentent chacune des signaux très caractéristiques. Nous allons donc réaliser les même analyses que précédemment mais sur chaque phase de saut prise séparemment afin de voir si cela va nous permettre de gagner en finesse d'analyse.

Les résultats détaillés de ces analyses sont visibles en Annexe C. Le découpage de la base de données par phase de saut n'a pas permis de mieux distinguer les notes que lors de l'analyse de l'intégralité de la base. C'est pourquoi nous ne détaillerons pas les résultats obtenus ici.

3 Comparaison des méthodes de prédiction

Afin de comparer les différentes méthodes présentées en Section 7, la base de données totale est divisée en deux : une base de données d'entrainement qui est composée d'un tirage aléatoire sans remise de 80% de la base de données, qui servira à construire le modèle, et les 20% restant constituent la base de données de test servant à valider le modèle. Le RMSE est calculé pour évaluer la performance de prédiction selon la formule suivante :

$$RMSE = \sqrt{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2/n},$$

avec y la note donnée par le juge, $\hat{y}$ la note estimée par le modèle et n le nombre de sauts de la base de test.

Dix simulations ont ainsi été réalisées pour chaque méthode et les moyennes et s.d. de RMSE obtenus pour chaque modèle sont résumés en Table 8.3. Plus la valeur du RMSE est faible, meilleure est la méthode.

On observe qu'en dehors de la régression logistique cumulative, toutes les méthodes ont un RMSE du même ordre de grandeur, légèrement inférieur à 1. On peut dire qu'en moyenne il y a un écart de 1 entre la note attribuée au saut par le juge et la note prédite par le modèle quelque soit le modèle prédictif utilisé. Cette valeur n'est pas satisfaisante car nous aurions au moins souhaité que cette erreur ne se réplique pas sur tous les individus et donc avoir un RMSE au moins inférieur à 0.5. Cela montre la difficulté qu'ont ces modèles à déterminer la qualité du saut à partir des signaux collectés par la selle et la nécessité de développer un modèle adapté à nos données.

Table 8.3: Moyenne (et s.d.) de RMSE pour 10 simulations de chaque méthode, le nom du package R utilisé est indiqué entre parenthèses

Méthode	Moyenne	s.d.
Régression logistique cumulative link model (ordinal)	1.23	0.04
Arbre de classification Rmc(T) taille optimisée (RpartScore)	0.87	0.02
Arbre de classification Rmc(T) taille base (RpartScore)	0.87	0.02
Arbre de classification Rmr(T) taille optimisée (RpartScore)	0.89	0.03
Arbre de classification Rmr(T) taille base (RpartScore)	0.88	0.02
Arbre de classif. quadratique Rmc(T) taille optimisée (RpartScore)	0.87	0.03
Arbre de classif. quadratique Rmc(T) taille base (RpartScore)	0.87	0.03
Arbre de classif. quadratique Rmr(T) taille optimisée (RpartScore)	0.91	0.05
Arbre de classif. quadratique Rmr(T) taille base (RpartScore)	0.88	0.03
Conditional inference trees (party)	0.78	0.03
Forêt Aléatoire (party)	0.96	0.05
Forêt ordinale (ordinalForest)	0.96	0.06
Recursive partitioning and regression tree (rpart)	0.87	0.03
Model-based gradient boosting (FDboost)	0.91	0.02

Chapitre 9

Conclusion et perspectives

Ce qui a motivé nos travaux est la recherche de paramètres objectifs d'analyse de la locomotion du cheval de saut d'obstacle. Nous avons développé, dans le Chapitre 5, des modèles de prédiction de la vitesse par foulée avec une précision de 0.6 m/s, et de la longueur de foulée avec une précision de 0.3 m. Puis, au Chapitre 8, a été développé l'étude de la qualité de saut à partir des signaux collectés par l'IMU et des paramètres calculés à partir de ces signaux (régularité à l'abord, énergies, etc.). Les limites et perspectives de ces études seront discutées dans cette dernière partie.

1 Discussion et perspectives applicatives

1.1 Estimation de la vitesse

Comme présenté en introduction, il existe trois grandes familles de méthodes classiquement utilisées pour estimer la vitesse à partir de données d'IMUs :

- Les méthodes basées sur un modèle physique comme le modèle du pendule inversé utilisé dans certains bracelets connectés pour la marche humaine. Aucun modèle physique ne permet de modéliser l'allure dissymétrique d'un cheval au galop, cette famille de méthodes ne peut donc pas être utilisée dans notre cas.

- Les méthodes basées sur le traitement du signal comme l'intégration simple des signaux collectés pour récupérer une vitesse ou la fusion de données issus de différents capteurs comme par exemple d'une IMU et d'un magnétomètre ou d'une IMU et d'un GPS.

- Les méthodes basées sur des modèles mathématiques comme celui que nous avons développé au Chapitre 5.

Nous allons, dans cette partie, comparer la performance de notre modèle SVM à deux autres méthodes de la famille du traitement du signal : l'intégration directe du signal et l'Overall Dynamic Body Acceleration [Wilson et al., 2006] qui a déjà été utilisée pour calculer la vitesse d'animaux en mouvement.

1.1.1 Méthode Overall Dynamic Body acceleration (ODBA)

Cette méthode proposée par [Wilson et al., 2006] est une méthode qui ne s'appuie pas sur une intégration directe du signal. Les auteurs proposent un paramètre appelé ODBA, calculé à partir des accélérations collectées dans les trois directions de l'espace, qui est très lié à la vitesse de déplacement d'un animal. Il se calcule en plusieurs étapes. Tout d'abord les signaux d'accélération sont soumis à un filtre passe-bas Butterworth d'ordre 4 avec une fréquence de coupure de 10 Hz. Ensuite, une correction d'angle est appliquée pour aligner l'axe Z de l'IMU avec le vecteur de gravité. Pour chaque axe, la moyenne du signal est soustraite aux données lissées. Ces valeurs sont ensuite passées à la valeur absolue et sommées. On obtient ainsi une valeur moyenne d'ODBA pour chaque foulée. Un modèle de régression linéaire est alors utilisé pour lier cette valeur moyenne d'ODBA pour une foulée à la vitesse de cette même foulée.

La régression linéaire est réalisée avec le logiciel R (v.3.4.0) [R Core Team, 2017a] et la fonction lm du package stats [R Core Team, 2017b].

1.1.2 Méthode d'intégration directe du signal

Les méthodes d'intégration directes du signal sont les approches les plus communément utilisées pour calculer le déplacement vertical d'un cheval à partir des signaux d'accélération [Pfau et al., 2005] [Bosch et al., 2018]. La principale difficulté de cette méthode est de prévenir la dérive induite par la double intégration. Cette dérive est d'autant plus importante que la période étudiée est longue. Cette méthode nécessite une première étape de pré-traitement des données. Tout d'abord, une rotation 3D est appliquée sur l'accéléromètre de façon à ce que l'axe Z soit aligné avec le vecteur de gravité. Ensuite, le signal d'accélération moyen est calculé sur trois foulées : la foulée précédente, la foulée actuelle et la suivante et est soustrait du signal d'accélération de la foulée actuelle afin d'en retirer la dérive. Enfin, le bruit est retiré à l'aide d'un filtre passe-bas de Buttterworth d'ordre quatre avec une fréquence de coupure de 10 Hz.

Le calcul de la vitesse à partir de l'accélération nécessite la connaissance de la constante d'intégration. Le choix de cette constante est complexe car nécessite de faire des hypothèses lourdes sur les données ou le processus d'acquisition : l'idéal est de distinguer un instant au sein de la foulée où la vitesse du capteur est nulle. Cette hypothèse est réaliste dans le cas où les chevaux se déplacent sur tapis roulant [Pfau et al., 2005] ou lorsque le capteur est positionné sur le membre. Dans notre cas où le capteur est positionné sur le garrot nous ne pouvons pas faire cette hypothèse. Nous allons donc proposer ici deux façons différentes de calculer la constante d'intégration. Une première méthode sans apport de données extérieures, pour se mettre dans la même situation que les méthodes SVM et ODBA où, une fois les modèles entrainés, seules les données de l'IMU seront nécessaires au calcul de la vitesse d'une foulée. Une seconde méthode basée sur la vitesse moyenne qui nécessiterait, pour son automatisation future, la présence d'un GPS pour fournir cette valeur. Dans chaque cas la première foulée considérée correspond à la première foulée du cheval rentrant dans le champ de mesure, lorsqu'il est donc déjà en mouvement.

Première proposition de constante d'intégration Lors de nos prises de mesures les chevaux ont été chronométrés sur des distances connues, ainsi une vitesse moyenne peut être calculée pour chaque passage d'un cheval selon la formule $vitesse = \frac{distance}{temps}$. Cette valeur de vitesse moyenne sera utilisée comme constante d'intégration pour l'intégration du signal d'accélération selon l'axe X pour chaque foulée. Cette approche correspond au cas où une estimation extérieure de la vitesse moyenne serait disponible.

Seconde proposition de constante d'intégration La constante d'intégration de la première foulée est calculée à partir du signal brut de cette même foulée où seule la correction par rapport au vecteur de gravité est appliquée. Ce signal est intégré deux fois (en considérant une constante d'intégration nulle) pour obtenir une distance et on obtient la vitesse correspondante à l'aide de la formule $vitesse = \frac{distance}{durée\ de\ la\ foulée}$. La durée de la foulée correspond au temps écoulé entre deux pics maximums sur l'axe Z. La valeur ainsi obtenue est utilisée comme constante d'intégration pour la première foulée. On fait ensuite l'hypothèse que la vitesse d'une foulée d'un cheval est très corrélée à la vitesse de la foulée précédente. A partir de la seconde foulée, on considère donc que la constante d'intégration est égale à la vitesse calculée pour la foulée précédente.

1.1.3 Critères de comparaison des différentes méthodes

Pour pouvoir comparer les différents modèles présentés entre eux, la base de données à notre disposition est coupée en deux : une base de données d'entrainement qui est composée d'un tirage aléatoire sans remise de 80% de la base et d'une base de données de test contenant les 20% de données restantes. Cet échantillonnage prévient le surajustement car une même foulée ne pourra pas être dans la base d'entrainement du modèle et dans la base de test en même temps.

Pour le modèle d'intégration directe du signal, qui ne nécessite pas d'entrainement, la précision de l'estimation sera évaluée sur la base de données de test. Pour les modèles ODBA et SVM, la base de données d'entrainement va servir à la construction du modèle puis sa précision sera évaluée sur la base de test.

Pour chacun des modèles précédents, le pourcentage d'erreur supérieur à 0.6 m/s sera calculé. Ce seuil est le seuil minimum satisfaisant pour que l'utilisation de ce paramètre ait un sens pour les professionnels du secteur équestre. Sa formule est la suivante :

$$\begin{aligned} \%\ \text{erreur} &= 100 \times \\ &\sum_i \frac{|\text{Vitesse mesurée pour la foulée i} - \text{Vitesse prédite pour la foulée i}| > 0.6}{\text{Nombre de foulées}}, \end{aligned}$$

avec i correspondant à chaque foulée de la base de test.

Les modèles seront aussi comparés à l'aide du graphe de Bland et Altman et son intervalle d'acceptation de 95% [Bland and Altman, 1999] qui permet une représentation graphique de l'écart entre deux méthodes utilisées sur les mêmes individus (ici les foulées). Dans ce cas, nous allons examiner les différences entre chacune de ces trois méthodes et le système de référence utilisé pour les mesures (le système de tracking 2D pour les lignes droites et le chronomètre à déclenchement automatique pour les lignes courbes). Pour cela, nous avons utilisé la fonction bland.altman.plot du package R BlandAltmanLeh [Lehnert, 2015].

1.1.4 Résultats

Table 9.1: Valeur moyenne, minimum et maximum de pourcentage d'erreur supérieur à 0.6 m/s et largeur moyenne (Sd) de l'intervalle de confiance (IC) de Bland et Altman pour 50 répétitions

	SVM	ODBA	Intégration 1[1]	Intégration 2[2]
Moyenne d'erreur > 0.6 m/s	10.9%	51.4%	86.2%	12.0%
Minimum d'erreur > 0.6 m/s	9.0%	47.8%	83.8%	10.1%
Maximum d'erreur > 0.6 m/s	14.0%	55.1%	89.0%	14.0%
Largeur moyenne de l'IC	1.7 m/s	3.9 m/s	5.4 m/s	1.8 m/s
Sd	0	0.1	0.1	0.1

Pour limiter la fluctuation des résultats liée au tirage aléatoire de la base d'entrainement et de test, le processus décrit précédemment sera répété 50 fois et les valeurs moyennes,

[1]Méthode d'intégration avec la constante calculée
[2]Méthode d'intégration avec la constante mesurée

minimales et maximales du pourcentage d'erreur seront présentées, ainsi que la valeur moyenne de la taille de l'intervalle de confiance de Bland et Altman.

La Table 9.1 présente les résultats moyens obtenus pour chaque modèle. Avec un pourcentage d'erreur moyen de 10.9% pour le SVM et 12.0% pour l'intégration avec la constante mesurée ces deux méthodes surpassent les méthodes ODBA et intégration avec constante calculée.

Le graphe de Bland et Altman pour une simulation du modèle SVM est présenté en Figure 9.1, où un point correspond à une foulée de la base de test. La vitesse prédite par le modèle et la vitesse mesurée à l'aide du système de référence sont comparés pour chaque foulée, le biais entre les deux méthodes est représenté sur l'axe des ordonnées. L'axe des abscisses présente la valeur moyenne de la vitesse prédite et mesurée pour une foulée. Les deux lignes pointillées extrêmes représentent les bornes de l'intervalle d'acceptation et la ligne pointillé centrale représente le biais moyen.

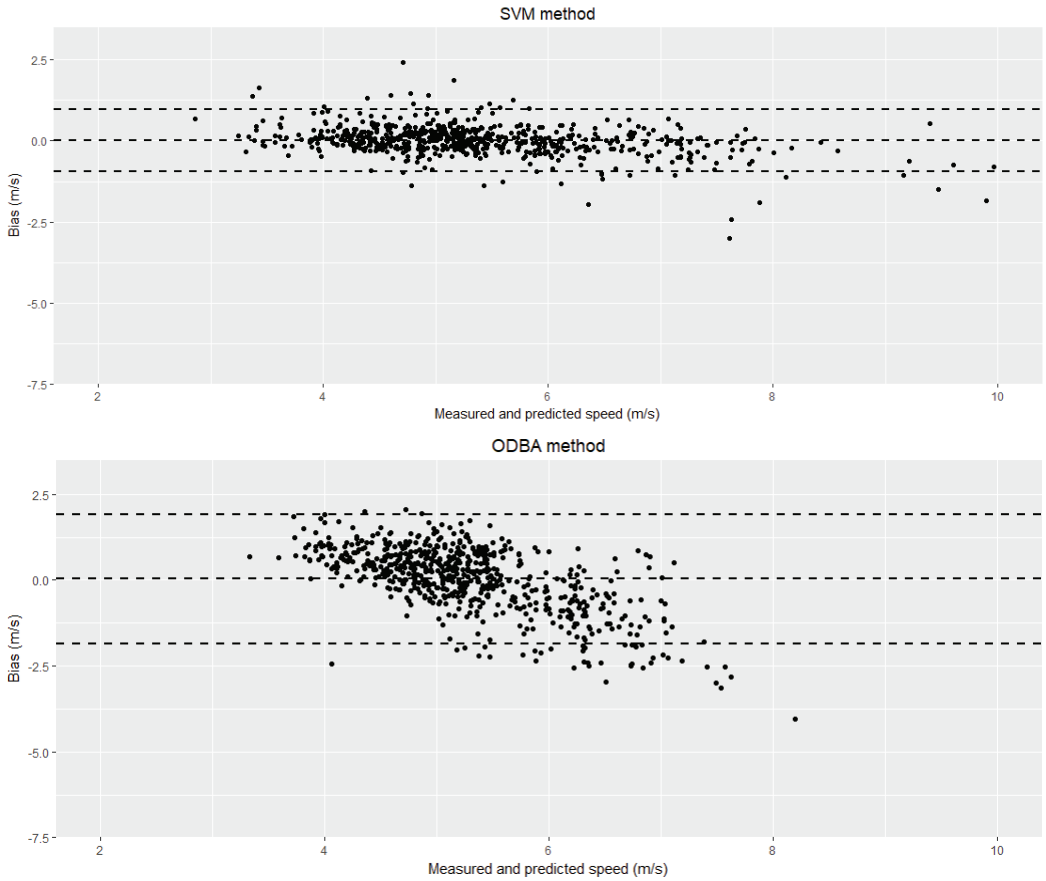


Figure 9.1: Graphe de Bland and Altman pour une simulation du modèle SVM avec son intervalle de confiance à 95% (en-haut) et méthode ODBA (en-bas)

Pour le modèle SVM, le biais moyen est 0, ce qui signifie qu'en moyenne les prédictions du modèle sont proches de la valeur mesurée par le système de référence. Si les prédictions du modèle étaient parfaites, tous les points seraient allignés sur la ligne du 0. Les points les plus éloignés de cette ligne correspondent aux plus mauvaises prédictions. On observe de plus que, pour des foulées de faible vitesse (inférieures à 5 m/s), le modèle SVM a tendance à surestimer leur valeur alors que pour certaines foulées de grande vitesse (supérieures à 5 m/s), le modèle a tendance à les sous-estimer. Mais 95% des foulées ont un biais inférieur à 1 m/s, que l'on considère satisfaisant compte tenu de notre objectif

d'atteindre une précision de 0.6 m/s.

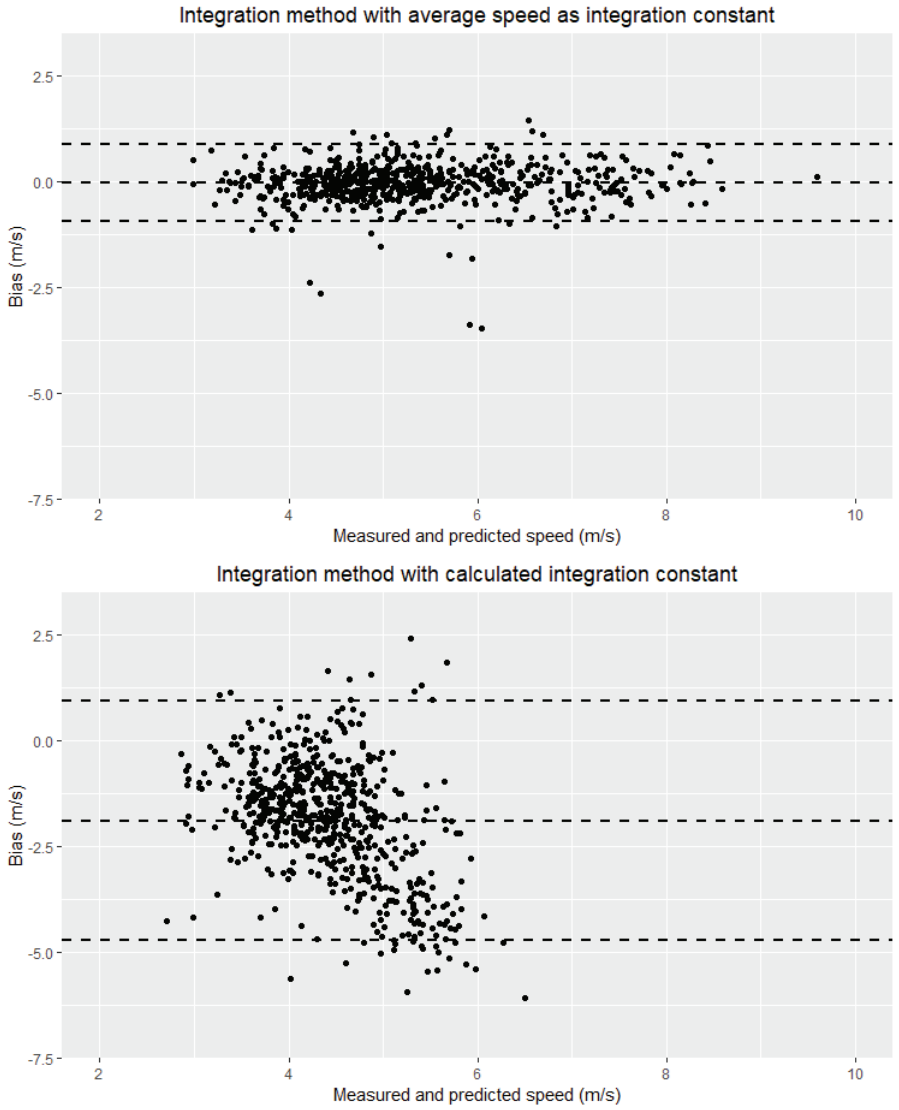


Figure 9.2: Graphe de Bland et Altman pour une simulation du modèle d'intégration 1 et son intervalle de confiance à 95% (en-haut) et du modèle d'intégration 2 (en-bas)

Dans le cas du modèle d'intégration avec la constante calculée, le biais moyen est de 2 m/s, ce qui est nettement supérieur aux autres méthodes (cf. Figure 9.2 en-bas). Cette méthode a tendance à globalement sous-estimer la vitesse du cheval. Les meilleurs résultats obtenus avec la méthode d'intégration avec la vitesse mesurée (cf. Figure 9.2 en-haut) montrent la difficulté de trouver la bonne constante d'intégration sans aucun apport d'information extérieure. En effet, pour la constante mesurée, la moyenne d'erreur de prédiction est légèrement supérieure à celle du SVM mais le biais moyen est identique. Nous pouvons tout de même noter que d'aussi bons résultats avec cette méthode d'intégration peuvent s'expliquer par la précision élevée de la vitesse moyenne. Ces résultats auraient été moins bons si la vitesse moyenne avait été estimée à l'aide d'un GPS de qualité moyenne. Pour finir, les valeurs estimées par la méthode ODBA sont plus variables que celles de la méthode SVM (cf. Figure 9.1 en-bas). Pour la méthode ODBA le biais moyen est aussi nul mais la taille de l'intervalle de confiance à 95% est le double de celle des SVMs.

1.1.5 Discussion et conclusion

L'objectif de notre étude est de développer un objet connecté qui peut fournir au cavalier des paramètres de locomotion du cheval à partir d'une IMU positionnée sur le garrot du cheval. Le nombre de capteurs est réduit à son minimum pour faciliter son utilisation quotidienne et l'absence de GPS est due à notre volonté de faire fonctionner cet outil aussi bien en intérieur qu'en extérieur.

Habituellement, les modèles utilisés pour l'estimation de la vitesse détectent dans un premier temps chaque foulée, puis découpent le signal selon ces foulées et, pour finir, appliquent un modèle de calcul [Bichler et al., 2012, Mannini and Sabatini, 2014, Zhao et al., 2016]. Ces méthodes standardisent la durée de la foulée et peu d'entre elles prennent en compte cette durée pour le calcul du paramètre de vitesse entrainant une perte d'information. Dans notre cas, nous avons choisi un pré-traitement différent des données où chaque foulée est détectée sur l'axe Z et 101 points sont conservés à partir du pic maximal. Cette modification permet de conserver, d'une certaine façon, l'information sur la durée de la foulée : pour les foulées de grande vitesse, le signal d'une "foulée" contiendra plus de cycles de galop que celui des "foulées" de plus faible vitesse. Comme on peut le voir à la Table 9.2, ce découpage permet un gain de précision avec notre modèle SVM lorsque l'on se compare au découpage classique d'une foulée ré-échantillonnée, le ré-échantillonnage permettant d'avoir le même nombre de points pour chaque foulée.

Table 9.2: Valeur moyenne, minimum et maximum de pourcentage d'erreur supérieur à 0.6 m/s et largeur moyenne et Sd de l'intervalle de confiance (IC) de Bland et Altman pour 50 répétitions

	SVM découpage 101 points	SVM découpage classique
Moyenne d'erreur > 0.6 m/s	10.9%	12.8%
Minimum d'erreur > 0.6 m/s	9.0%	10.9
Maximum d'erreur > 0.6 m/s	14.0%	15.5%
Largeur moyenne de l'IC	1.7 m/s	1.8 m/s
Sd	0	0.1

Pour évaluer la performance de notre modèle, nous l'avons comparé à deux méthodes qui ne nécessitent pas d'apport de données extérieures pour estimer la vitesse par foulée : l'ODBA et l'intégration avec une constante calculée. La performance du modèle SVM est nettement supérieure à ces deux méthodes. Les résultats médiocres de l'intégration du signal avec constante calculée peuvent s'expliquer par la difficulté de calculer la constante d'intégration quand le positionnement de l'IMU ne permet pas de faire l'hypothèse qu'à un instant de la foulée la vitesse est nulle, hypothèse classiquement utilisée chez l'humain ou quand l'IMU est positionnée sur le membre. La méthode d'intégration du signal avec constante mesurée donne des résultats légèrement moins précis que les SVM. Par contre, cette méthode nécessite la connaissance de la vitesse moyenne, et n'est donc pas applicable à notre objectif de prédire la vitesse du cheval de façon précise à partir d'un IMU uniquement.

La nouveauté de ce travail est de proposer un modèle pour l'estimation de la vitesse qui s'appuie sur une unique IMU. L'intégration d'un modèle de machine learning dans un outil pour les sports équestres est innovant lorsque l'on se compare aux systèmes déjà existants pour les sports équestres qui s'appuient sur un GPS, ou lorsque l'on se compare aux systèmes de suivi du mouvement pour l'homme qui s'appuient principalement sur une IMU avec magnétomètre intégré ou plusieurs IMUs [Filippeschi et al., 2017].

Pour conclure, les approches de type machine learning ont permis le développement d'un objet connecté qui ne repose pas sur un GPS pour l'estimation d'un phénomène physique, ici la vitesse du cheval à chaque foulée, avec une précision de 0.6 m/s. Cette précision répond aux attentes des professionnels de la discipline du saut d'obstacle, la précision étant leur principale préoccupation pour utiliser de tels objets connectés.

L'intégration d'un modèle de machine learning dans un outil pour les sports équestres est innovant car les outils proposés par nos concurrents principaux (Seaver, Equisense) s'appuyent sur un GPS pour calculer ce paramètre. Contrairement à eux, notre selle connectée peut fonctionner aussi bien en intérieur qu'en extérieur.

Nous n'avons pas pu mettre en compétition notre modèle avec d'autres travaux publiés réalisés sur les chevaux car personne ne fournit une estimation de vitesse par foulée à ce jour. En effet, [Pfau et al., 2005] et [Bosch et al., 2018] calculent la distance parcourue avec précision mais [Bosch et al., 2018] évoquent avoir pour objectif de travailler sur l'estimation de la vitesse dans des travaux futurs. Alors qu'en recherche humaine une littérature abondante est disponible à propos du calcul de vitesse d'un homme lors de la marche. On peut citer, par exemple, [Mannini and Sabatini, 2014] qui ont placé une IMU sur le pied et comparent deux méthodes d'estimation de la vitesse dont la précision est comprise entre 0.5 km/h (0.14 m/s) et 0.7 km/h (0.19 m/s) selon la vitesse de marche et la méthode utilisée. [Zihajehzadeh et al., 2016] ont développé un modèle pour la vitesse de marche basé sur un modèle de régression qui utilise les données d'un capteur inertiel porté au poignet. Dans leur travail, les bornes de l'intervalle d'acceptation de Bland et Altman sont inférieures à 0.2 m/s. Pour finir, [Sabatini and Mannini, 2016] estiment une vitesse instantanée qu'ils décomposent dans les trois directions de l'espace à partir de deux IMUs positionnées sur le pelvis et sur le mollet des sujets. La précision de leur méthode est du même ordre de grandeur que les précédentes. Ainsi, si l'on ramène ces erreurs à la vitesse du sujet, en considérant la vitesse de marche d'un homme de 3 km/h (0.8 m/s), l'erreur des trois modèles décrits précédement est d'environ 27% alors que pour un cheval de saut d'obstacle, qui galope à une vitesse de 350 m/min (5.8 m/s) l'erreur de notre modèle est de 10%. Ainsi, notre modèle est beaucoup plus performant que les travaux existants réalisés sur l'homme.

Pour continuer d'affiner la précision de notre modèle, plus de campagnes de mesures devraient être réalisées avec les deux systèmes de référence. En effet, un panel de 58 chevaux n'est pas suffisant pour modéliser le comportement de tous les chevaux compte tenu de la grande variabilité individuelle observée. Il est aussi nécessaire de réaliser plus de mesures sur des courbes de diamètre varié puisqu'il a été démontré que ce diamètre influence fortement le comportement du cheval, le signal collecté par l'IMU et donc la vitesse des foulées finale [Greve and Dyson, 2016].

Notre modèle présente aussi des limites, il ne peut pas être directement transposé à une autre discipline que le saut d'obstacle car le galop des chevaux de saut est spécifique à la discipline. En effet, les chevaux d'endurance ont un galop plutôt horizontal alors que les chevaux d'obstacle ont un galop avec plus de rebond vertical. Ainsi, pour transposer notre outil à d'autres disciplines sportives, équines ou humaines, il sera nécessaire de collecter des données spécifiques à ces nouvelles situations pour que le modèle soit tout aussi performant.

1.2 Estimation de la longueur de foulée

A l'initialisation du projet, la longueur de foulée était mesurée à l'aide du système 2D lorsque le cheval était en ligne droite et avec un odomètre dans le cas de lignes courbes. La précision voulue par l'entreprise pour ce paramètre était de 20 cm. Or, quel que soit le modèle mathématique utilisé le pourcentage d'erreur restait important, et nous avons conclu à une précision de 30 cm au Chapitre 5. En observant graphiquement les valeurs mesurées avec le système de référence versus les valeurs estimées par le modèle, nous nous sommes rendus compte que les foulées les plus mal estimées étaient celles en bord de champ des caméras du système 2D.

Nous avons donc réfléchi à un autre système de mesure de la longueur foulée qui nous permettrait de mesurer la longueur de foulée aussi bien en ligne droite qu'en courbe avec moins de dérive que le système 2D. Pour cela, nous avons utilisé l'odomètre (cf. Figure 1.6) dont la précision est de l'ordre du centimètre. Nous avons refait des campagnes de mesures où la longueur de foulée a été mesurée avec l'odomètre en ligne droite et en virage. Le sol a donc été ratissé entre chaque passage d'un cheval afin de repérer aisément le posé de l'antérieur directeur et mesurer la distance parcourue à l'aide de l'odomètre.

Lorsque l'on compare la précision du SVM appliqué sur les données mesurées avec le système 2D et celle du SVM appliqué sur les données mesurées à l'aide de l'odomètre (cf. Table 9.3), on observe que l'erreur de prédiction est divisée par trois. Ceci peut être dû au fait que, pour que la mesure de longueur de foulée avec le système 2D soit aussi précise, le cheval doit être parfaitement parallèle au champ des caméras, ce qui est rarement le cas en pratique.

Table 9.3: Valeur moyenne, minimum et maximum de pourcentage d'erreur supérieur à 0.3 et 0.2 m pour 50 simulations avec le modèle SVM

	Données issues du système 2D	Données odomètre
Moyenne d'erreur > 0.3 m	18.7%	**4.3**%
Minimum d'erreur > 0.3 m	16.0%	0%
Maximum d'erreur > 0.3 m	21.9%	13.7%
Moyenne d'erreur > 0.2 m	34.4%	**13.4**%
Minimum d'erreur > 0.2 m	30.5%	7.8%
Maximum d'erreur > 0.2 m	38.2%	25.5%

Ainsi, en se basant uniquement sur les données mesurées avec l'odomètre, les SVM permettent d'atteindre une précision de 0.2 m pour la longueur de foulée. La précision du modèle SVM pour prédire la longueur de foulée est maintenant limitée par la précision de l'outil de mesure. Pour pouvoir affiner encore la qualité de la prédiction, il faudrait trouver un outil de mesure plus précis que l'odomètre pour construire la base de données d'entrainement.

Par la suite, il pourrait aussi être intéressant de trouver un système permettant d'automatiser la mesure de la longueur de foulée, la mesure avec l'odomètre étant très chronophage car elle nécessite de ratisser la zone de mesure entre chaque passage d'un cheval pour faciliter le repérage du posé de l'antérieur directeur.

1.3 Estimation de la qualité du saut

Comme observé au Chapitre 8, la base de données de notes à notre disposition, qui a été constituée à l'aide de juges amateurs, est déséquilibrée pour les notes extrêmes. Nous

avons en effet beaucoup plus de notes moyennes (3-4), que de très bonnes (5), ou très mauvaises notes (1-2), rendant difficile la construction d'un modèle performant pour la prédiction de qualité du saut. Ceci peut aussi s'expliquer par le fait que les cavaliers filmés étaient tous des cavaliers professionnels avec peu de chevaux inexpérimentés, qui faisaient donc peu d'erreurs dans leur façon d'aborder les obstacles.

Afin de construire un modèle mathématique fiable, il est nécessaire de collecter des données à plus grande échelle et dans des conditions plus standardisées. C'est pourquoi un partenariat avec le Stud-Book des Selles Français est en cours de montage. Cette association évalue, à l'aide de juges professionnels, les jeunes chevaux de saut d'obstacle de leurs 2 ans à leurs 7 ans lors d'épreuves de saut en liberté et de saut monté. Ce partenariat nous permettrait de collecter les données des chevaux testés en les équipant de notre selle connectée et de récupérer en parallèle les notes des juges pour chaque cheval. Ces sessions de test ont lieu environ 4 fois par an et rassemblent à chaque fois plusieurs dizaines de chevaux qui sont ré-évalués tous les ans jusqu'à leurs 6 ans (7 ans pour les étalons).

Suite à la constitution de cette base de données, il pourra aussi être intéressant d'étudier l'évolution des notes d'un même cheval au cours de ses différentes années de test afin de voir si l'on arrive à lier les performances sportives du cheval (classement en concours) à ses paramètres de locomotion mesurés lors de ces scéances de test. L'objectif final serait de créer un modèle de détection de futurs champions ou un modèle d'évaluation de la valeur marchande du cheval compte tenu de ses paramètres de locomotion. En revanche, il faudra par la suite enrichir la base de données avec des chevaux d'autres races, autres que le selle français, présentes aussi dans la discipline du saut d'obstacle.

Figure 9.3: Visuel de l'application iJump avec l'affichage de la vitesse et de la longueur de foulée

Pour conclure, mes travaux ont permis l'intégration dans l'application du paramètre vitesse par foulée avec une précision de 0.6 m/s et longueur de foulée avec une précision de 0.2 m. Ils font donc partie des paramètres que le cavalier peut choisir de synchroniser avec son parcours (cf. Figure 9.3).

Dans un futur proche, il est souhaité d'étendre cet outil au travail sur le plat, pour le

cheval de saut d'obstacle dans un premier temps. En effet, pour le moment, l'algorithme calcule les paramètres de locomotion du cheval entre deux sauts, mais si aucun saut n'est détecté aucun paramètre n'est fourni au cavalier. On souhaite maintenant pouvoir mettre à la disposition du cavalier certains paramètres, comme la vitesse ou la régularité par exemple, tout au long de ses séances de travail sur le plat, séances qui font partie intégrante du travail du cheval de sport. Dans un second temps, et à plus long terme, cet outil pourra être étendu aux autres disciplines équestres pour leur fournir des paramètres de locomotion du cheval spécifiques à leur discipline. Mais ceci requière une collecte de données importante.

2 Discussion et perspectives mathématiques

L'objectif de cette thèse était de fournir des modèles précis de prédiction de la vitesse, de la longueur et de la qualité du saut du cheval de sport. Pour cela, nous avons étudié les données selon deux angles d'approches : l'approche fonctionnelle et l'approche statistique multidimensionnelle. Nous avons vu à la Partie 5 que c'est l'approche statistique multidimensionnelle qui nous a permis de répondre aux objectifs de précision de l'entreprise.

Néanmoins, cette thèse a permis de contribuer à l'avancée des méthodes d'analyse de données fonctionnelles.

Dans un premier temps, nous avons cherché à développer un modèle de prédiction de la vitesse du cheval à partir de données d'une IMU. Lors de nos premières analyses, nous nous sommes rendus compte que le découpage de la base de données en sous groupes plus homogènes avant l'application d'un modèle de prédiction, à l'aide d'un modèle de mélange de régression, permettait d'obtenir de meilleurs résultats que lorsque l'on appliquait des modèles de régression sur l'ensemble de la base de données. Ensuite, nous avons mis en avant que le modèle de régression fonctionnel non paramétrique appliqué sur nos données permettait d'obtenir de meilleurs résultats que des modèles de régression en grande dimension comme les modèlées PLS ou Lasso par exemple. Nous avons alors testé des méthodes de clustering fonctionnel pour découper la base de données, afin de tenir compte de la globalité des signaux collectés pour le découpage, avant l'application d'un modèle de prédiction adapté à chaque sous-groupe. Dans la littérature, il existe de nombreuses méthodes de clustering fonctionnel univarié qui sont aussi disponibles sous la forme de packages R, alors que les méthodes multivariées, qui nous permettraient de prendre en compte les différents signaux collectés, sont moins nombreuses. Nous avons ainsi choisi de développer un modèle de clustering fonctionnel multivarié. Le modèle que nous proposons s'appuie sur un modèle de mélange latent fonctionnel avec une première étape de lissage des données dans une base de fonctions. Puis, les données sont modélisées dans des sous-espaces fonctionnels spécifiques aux groupes à l'aide d'une analyse en composantes principales fonctionnelle multivariée. Les probabilités d'appartenance aux clusters étant inconnues, un algorithme EM est utilisé pour estimer ces probabilités et calculer les paramètres du modèle de mélange gaussien. Notre modèle, appelé funHDDC, se décline en 6 sous-modèles : le modèle principal $[a_{kj}b_kQ_kD_k]$ et 5 sous-modèles plus parcimonieux en forçant les valeurs propres a_{kj} et b_k à être communs au sein d'une même classe ou entre classes.

Nous avons ainsi proposé une méthode de clustering fonctionnel qui permet d'aller au delà des limitations de la méthode proposée par [Jacques and Preda, 2014b], en proposant aussi un modèle principal parcimonieux mais qui conserve toute l'information disponible dans les données en supposant que les dernières fonctions propres du cluster sont de

variances égales, ce qui permet de réduire drastiquement le nombre de paramètres du modèle à estimer sans retirer d'information. Ce modèle peut être vu comme l'extension de la méthode proposée par [Bouveyron and Jacques, 2011] au cas multivarié. La mise en compétition de notre algorithme, disponible sur le CRAN [Schmutz et al., 2018], avec les quelques méthodes multivariées disponibles sous la forme de packages ou fournies par les auteurs, a permis de mettre en évidence sa supériorité sur trois exemples de difficulté croissante. Ces résultats ont fait l'objet d'une publication actuellement en cours de révision.

L'algorithme ainsi développé n'a pas eu de résultats concluants pour la prédiction de la vitesse du cheval. La mise en compétition de notre algorithme de clustering fonctionnel, afin obtenir des sous-groupes de foulées plus homogènes avant l'application d'un modèle de régression adapté à chaque sous-groupe, avec les méthodes d'analyse de données en grande dimension et les méthodes de machine learning classiques nous a permis de mettre en évidence que les résultats sont équivalents à appliquer un SVM sur l'ensemble de la base de données. Ceci peut sans doute s'expliquer par le fait qu'à ce jour notre base de données a été constituée sur 58 chevaux, ce qui est un nombre relativement faible d'individus pour du clustering et pour qu'il y ait un réel apport du clustering dans l'homogénéisation des données. D'un point de vue simplicité d'implémentation et rapidité de calcul, nous avons donc choisi d'implémenter, dans l'application téléphone développée par l'entreprise, le modèle SVM pour fournir le paramètre de vitesse par foulée au cavalier. Ces travaux ont fait l'objet d'une publication actuellement en cours de révision dans le journal Mathematical Problems in Engineering.

Mais notre algorithme de clustering a donné de bons résultats sur un autre exemple de prédiction : la prédiction de la consommation d'électricité. En effet, funHDDC a été utilisé pour la prédiction de la demande d'électricité où il a été combiné à l'algorithme de prédiction Pattern Sequence Forecasting [Martinez Alvarez et al., 2011] afin de créer un algorithme capable de prédire des séries temporelles fonctionnelles, appelé funPSF. Cet algorithme fonctionne de la façon suivante : les données sont normalisées puis, funHDDC est appliqué pour différentes partitions K et le nombre de clusters est sélectionné à l'aide de plusieurs critères connus comme le BIC. L'algorithme funHDDC est ensuite appliqué sur les données, découpées en jour de mesure pour attribuer chaque jour de mesure à une classe, les données sont ainsi transformées en une séquence de labels. Une séquence spécifique de labels est ensuite cherchée et détectée dans les données avant l'application de l'algorithme de prédiction. Ce travail a fait l'objet d'une publication acceptée dans le journal Energies [F. Martínez-Álvarez, 2019].

Dans un second temps, notre modèle de clustering fonctionnel multivarié a été étendu au co-clustering fonctionnel multivarié. Cette méthode permet de découper une matrice de données fonctionnelle, non plus en groupes d'individus qui se ressemblent comme funHDDC, mais en groupes d'individus (lignes) et de variables (colonnes) qui se ressemblent. Si on prend comme exemple en ligne des foyers et en colonne des jours pour lesquels on a le suivi de consommation électrique toutes les jours, le résultat de cet algorithme est l'obtention de blocs de foyers et de jours qui se ressemblent du point de vue de leur consommation électrique. Or, dans le cas fonctionnel multivarié, à l'intersection d'une ligne et d'une colonne de la base de données on a plusieurs courbes, pour faciliter l'interprétation de ces données fonctionnelles multivariées, il peut être intéressant de découper les données en blocs d'individus et de jours qui se ressemblent d'un point de vue des signaux mesurés et d'étudier plus précisement ces sous-groupes plutôt que la base de données au global.

En pratique pour l'entreprise, cette méthode pourrait permettre de mettre en évidence des épreuves pour lesquelles la locomotion du cheval était "identique" et détecter ainsi des typologies de chevaux ou des inconforts selon le type d'épreuve. Ce développement méthodologique a permis de mettre à disposition le premier modèle de co-clustering pour données fonctionnelles multivariées. Pour la suite, ce modèle pourra être étendu de façon à proposer plusieurs matrices de variance-covariance permettant de rendre notre modèle plus flexible et plus adaptable à la diversité des données collectées. Ces travaux font l'objet d'une publication qui sera soumise prochainement et un package R a été développé, il sera prochainement disponible sur le CRAN.

Comme nous avons pu le voir tout au long de ce manuscrit la structure des données est complexe avec des dépendances multiples : la selle iJump collecte des données de parcours d'obstacles. Un même cavalier peut réaliser plusieurs parcours et donc se retrouver plusieurs plusieurs fois dans la base de données. Dans nos travaux, le but de notre modèle est de prédire des paramètres de locomotion pour de nouvelles foulées, nous avons négligé l'aspect cavalier car il nous semblait difficile à prendre en compte lorsqu'un nouveau cavalier utilise la selle. Mais nous pourrions envisager, utiliser notre modèle simplifié dans un premier temps puis, lorsque l'on a un nombre conséquent de parcours pour un même cavalier, réajuster le modèle de prédiction par cavalier.

De même il serait intéressant de prendre en compte la dépendance des foulées d'un même parcours. Dans le but de simplifier notre approche méthodologique nous avons dans un premier temps choisi de la négliger. Un développement futur pourrait être d'adapter nos modèles de clustering et de co-clustering aux données fonctionnelles dépendantes en s'appuyant sur les travaux déjà existants [Kokoszka, 2012, Chen and Müller, 2012].

Lors de cette thèse, nous nous sommes intéressés à des modèles non supervisés pour données fonctionnelles multivariées. Nous n'avons pas abordé les modèles supervisés, que ce soit dans un but de classification ou de régression. Or, le dernier objectif de cette thèse était de prédire une variable ordinale, la qualité de saut, à partir de prédicteurs fonctionnels, les signaux collectés par la selle. Dans le Chapitre 7, nous avons vu de nombreuses méthodes de statistique multidimensionnelle permettant de prédire une variable ordinale par des variables quantitatives dont aucune ne s'est démarquée pour la prédiction de la qualité du saut. En effet, toutes les méthodes testées avaient un RMSE moyen proche de 1, pouvant être interprété comme une erreur moyenne de 1 entre la note réelle attribuée au saut et celle estimée par le modèle de prédiction. De plus, nous avons vu qu'aucun modèle n'existe à ce jour pour réaliser une régression d'une réponse ordinale avec plusieurs prédicteurs fonctionnels. Récemment, des modèles de régression logistique fonctionnelle ont été développés [Peng Wei, 2014] [Mousavi and Sørensen, 2017], mais aucune de ces méthode n'est disponible sous la forme d'un package R. [Mousavi and Sørensen, 2017] compare trois méthodes de régression logistique fonctionnelle sur des données simulées et sur un exemple réel de détection de boiterie chez le cheval. Alors que sa variable réponse comporte trois modalités (normal, boiteux diagonal droit, boiteux diagonal gauche), il découpe sa question en trois scénarios binaires et applique la régression logistique fonctionnelle sur chaque scénario, ce qui n'est pas le plus adapté pour répondre à la question et augmente le risque de faux positifs. Il pourrait donc être intéressant de développer un modèle de régression fonctionnelle pour variable ordinale avec un prédicteur fonctionnel ainsi qu'un modèle de régression fonctionnelle pour variable ordinale avec plusieurs prédicteurs fonctionnels afin de voir l'impact de la prise en compte de plusieurs variables

fonctionnelles sur la qualité de la prédiction, et de les rendre disponibles sous la forme d'un package R. Ce qui pourrait faire l'objet d'un développement futur.

Communications réalisées au cours de la thèse

Posters :

- *Speed prediction of the sport horse from accelerometric and gyroscopic data*, Statlearn 2017, 5/04/17 au 07/04/17, Lyon, France.
- *Speed prediction of the sport horse from accelerometric and gyroscopic data,* IWFOS 2017, 15/06/17 au 17/06/17, La Corogne, Espagne.
- *Prédiction de la vitesse du cheval de sport à partir de données accélérométriques et gyroscopiques,* Journée de la Recherche Equine 2018, 15/03/2018, Paris, France.
- *FunHDDC : expansion of the R package for the clustering of multivariate functional curves*, Statlearn 2018, 4/04/18 au 6/04/18, Nice, France.

Communications orales :

- *Prédiction de la vitesse du cheval de sport à partir de données accélérométriques et gyroscopiques*, JDS 2017, 29/05/17 au 2/06/17, Avignon, France.
- *Données fonctionnelles multivariées issues d'objets connectés : une méthode pour classer les individus,* JDS 2018, 28/05/18 au 1/06/18, Saclay, France.
- *FunHDDC : extension du package R pour le clustering de courbes fonctionnelles multivariées*, Rencontres R 2018, 4/06/18 au 6/06/18, Rennes, France.
- *Co-clustering de courbes fonctionnelles multivariées,* JDS 2019, du 3/06/2019 au 7/06/2019, Nancy, France.
- *funHDDC, a R package to cluster univariate and multivariate functional data*, UseR!, du 9/07/2019 au 12/07/2019, Toulouse, France.
- *Co-clustering of multivariate functional curves,* IFCS 2019, du 26/08/2019 au 29/08/2019, Thessalonique, Grèce.

Publications :

Acceptée

- F. Martínez-Álvarez, A. Schmutz, G. Asencio-Cortés, J. Jacques (2019). A novel hybrid algorithm to forecast functional time series based on pattern sequence similarity with application to electricity demand, Energies, 12[1].

En révision

- A. Schmutz, J. Jacques, C. Bouveyron, L. Chèze, P. Martin (2018). Clustering multivariate functional data in group specific functional subspaces, Preprint HAL n°01652467v2.

Soumises

- A. Schmutz, L. Chèze, J. Jacques, P. Martin (2019). A method to estimate horse speed at canter from IMU data with Machine Learning. Mathematical Problems in Engineering.
- A. Schmutz, J. Jacques, C. Bouveyron, L. Chèze, P. Martin (2019). A model to co-cluster multivariate functional data.

Bibliographie

[Abraham et al., 2003] Abraham, C., Cornillon, P. A., Matzner-Løber, E., and Molinari, N. (2003). Unsupervised curve clustering using B-splines. *Scandinavian Journal of Statistics. Theory and Applications*, 30(3):581–595.

[Agresti, 2010] Agresti, A. (2010). *Analysis of Ordinal Categorical Data*. Wiley Series in Probability and Statistics. John Wiley and Sons, Hoboken, NJ.

[Archer and Mas, 2009] Archer, K. J. and Mas, V. R. (2009). Ordinal response prediction using bootstrap aggregation, with application to a high-throughput methylation data set. *Statistics in Medicine*, 28(29):3597–3610.

[Bichler et al., 2012] Bichler, S., Ogris, G., Kremser, V., Schwab, F., Knott, S., and Baca, A. (2012). Towards high-precision imu/gps-based stride-parameter determination in an outdoor runners' scenario. *9th Conference of the International Sports Engineering Association (ISEA)*, 34:592–597.

[Bishop, 2006] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

[Bland and Altman, 1999] Bland, J. M. and Altman, D. G. (1999). Measuring agreement in method comparison studies. *Statistical Methods in Medical Research*, 8(2):135–160.

[Bosch et al., 2018] Bosch, S., Bragança, F. S., Marin-Perianu, R., van der Zwaag, B., Voskamp, J., Back, W., van Weeren, R., and Havinga, P. (2018). Equimoves: A wireless network inertial measurement system for objective examination of horse gait. *Sensors*, 18(3).

[Boullé, 2012] Boullé, M. (2012). Functional data clustering via piecewise constant nonparametric density estimation. *Pattern Recognition*, 45(12):4389 – 4401.

[Bouveyron and Brunet, 2013] Bouveyron, C. and Brunet, C. (2013). Model-Based Clustering of High-Dimensional Data: A review. *Computational Statistics and Data Analysis*, 71:52–78.

[Bouveyron et al., 2015] Bouveyron, C., Come, E., and Jacques, J. (2015). The discriminative functional mixture model for the analysis of bike sharing systems. *Annals of Applied Statistics*, 9(4):1726–1760.

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

[Brandes et al., 2006] Brandes, M., Zijlstra, W., Heikens, S., van Lummel, R., and Rosenbaum, D. (2006). Accelerometry based assessment of gait parameters in children. *Gait Posture*, 24(4):482–486.

[Breiman et al., 1984] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA.

[Brzostowski, 2018] Brzostowski, K. (2018). Novel approach to human walking speed enhancement based on drift estimation. *Biomedical Signal Processing and Control*, 42:18–29.

[Cattell, 1966] Cattell, R. (1966). The scree test for the number of factors. *Multivariate Behaviour Research*, 1(2):245–276.

[Chen and Müller, 2012] Chen, K. and Müller, H.-G. (2012). Modeling repeated functional observations. *Journal of the American Statistical Association*, 107(500):1599–1609.

[Chiou and Li, 2007] Chiou, J.-M. and Li, P.-L. (2007). Functional clustering and identifying substructures of longitudinal data. *Journal of the Royal Statistical Society. Series B. Statistical Methodology*, 69(4):679–699.

[Christensen, 2018] Christensen, R. H. B. (2018). ordinal—regression models for ordinal data. R package version 2018.8-25. http://www.cran.r-project.org/package=ordinal/.

[Cornillon et al., 2008] Cornillon, P.-A., Guyader, A., Husson, F., Jégou, N., Josse, J., Kloareg, M., Matzner-Løber, E., and Rouviere, L. (2008). *Statistique avec R*. Presses Universitaires de Rennes.

[Cuevas et al., 2004] Cuevas, A., Febrero-Bande, M., and Fraiman, R. (2004). An anova test for functional data. *Computational Statistics & Data Analysis*, 47:111–122.

[Dempster et al., 1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B. Methodological*, 39(1):1–38.

[der Kruk and Reijine, 2018] der Kruk, E. V. and Reijine, M. (2018). Accuracy of human motion capture systems for sport applications; state of the art review. *European Journal of Sport Science*, 18(6):806–819.

[F. Martínez-Álvarez, 2019] F. Martínez-Álvarez, A. Schmutz, G. A.-C. J. J. (2019). A novel hybrid algorithm to forecast functional time series based on pattern sequence similarity with application to electricity demand. *Energies*, 12(1):94.

[Febrero-Bande and Oviedo de la Fuente, 2012] Febrero-Bande, M. and Oviedo de la Fuente, M. (2012). Statistical computing in functional data analysis: The R package fda.usc. *Journal of Statistical Software*, 51(4):1–28.

[Ferraty and Vieu, 2006a] Ferraty, F. and Vieu, P. (2006a). *Nonparametric functional data analysis*. Springer Series in Statistics. Springer, New York.

[Ferraty and Vieu, 2006b] Ferraty, F. and Vieu, P. (2006b). *Nonparametric Functional Data Analysis: Theory and Practice (Springer Series in Statistics)*. Springer-Verlag, Berlin, Heidelberg.

[Filippeschi et al., 2017] Filippeschi, A., Schmitz, N., Miezal, M., Bleser, G., Ruffaldi, E., and Stricker, D. (2017). Survey of motion tracking methods based on inertial sensors: A focus on upper limb human motion. *Sensors*, 17.

[Galimberti et al., 2012a] Galimberti, G., Soffritti, G., and Di Maso, M. (2012a). Classification trees for ordinal responses in r: The rpartscore package. *Journal of Statistical Software*, 47.

[Galimberti et al., 2012b] Galimberti, G., Soffritti, G., and Maso, M. D. (2012b). Classification trees for ordinal responses in R: The rpartScore package. *Journal of Statistical Software*, 47(10):1–25.

[Gastin et al., 2008] Gastin, P., K., N., and K., W. (2008). Determination of stride frequency using an accelerometer during equine locomotion. *Exercise and Nutrition Sciences*.

[Govaert and Nadif, 2013] Govaert, G. and Nadif, M. (2013). *Co-Clustering*. Wiley-IEEE Press, 1st edition.

[Greve and Dyson, 2016] Greve, L. and Dyson, S. (2016). Body lean angle in sound dressage horses in-hand, on the lunge and ridden. *Veterinary journal*, 217:52–57.

[Hastie et al., 2001] Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA.

[Hornung, 2019a] Hornung, R. (2019a). Ordinal forests. *Journal of Classification*.

[Hornung, 2019b] Hornung, R. (2019b). *ordinalForest: Ordinal Forests: Prediction and Variable Ranking with Ordinal Target Variables*. R package version 2.3-1.

[Hothorn et al., 2006] Hothorn, T., Hornik, K., and Zeileis, A. (2006). Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, 15(3):651–674.

[Ieva et al., 2011] Ieva, F., Paganoni, A., Pigoli, D., and Vitelli, V. (2011). *ECG signal reconstruction, landmark registration and functional classification*. Padova.

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

[Jacques and Preda, 2014b] Jacques, J. and Preda, C. (2014b). Model based clustering for multivariate functional data. *Computational Statistics and Data Analysis*, 71:92–106.

[James and Sugar, 2003] James, G. and Sugar, C. (2003). Clustering for sparsely sampled functional data. *Journal of the American Statistical Association*, 98(462):397–408.

[James et al., 2014] James, G., Witten, D., Hastie, T., and Tibshirani, R. (2014). *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated.

[Janitza et al., 2014] Janitza, S., Tutz, G., and Boulesteix, A.-L. (2014). Random forests for ordinal response data: Prediction and variable selection.

[Johansson, 2009] Johansson, E. (2009). *Choice of resting place and shelter by suckled cows kept outside during winter time.* PhD thesis, Swedish University of Agricultural Sciences, Uppsala.

[Kampakis, 2013] Kampakis, S. (2013). Comparison of machine learning methods for predicting the recovery time of professional football players after an undiagnosed injury. In *MLSA@PKDD/ECML*, volume 1969 of *CEUR Workshop Proceedings*, pages 58–68. CEUR-WS.org.

[Kayano et al., 2010] Kayano, M., Dozono, K., and Konishi, S. (2010). Functional Cluster Analysis via Orthonormalized Gaussian Basis Expansions and Its Application. *Journal of Classification*, 27:211–230.

[Kokoszka, 2012] Kokoszka, P. (2012). Dependent functional data. *International Scholarly Research Network*, 2012:30.

[Lê et al., 2008] Lê, S., Josse, J., and Husson, F. (2008). FactoMineR: A package for multivariate analysis. *Journal of Statistical Software*, 25(1):1–18.

[Lehnert, 2015] Lehnert, B. (2015). *BlandAltmanLeh: Plots (Slightly Extended) Bland-Altman Plots.* R package version 0.3.1.

[Leisch, 2004] Leisch, F. (2004). FlexMix: A general framework for finite mixture models and latent class regression in R. *Journal of Statistical Software*, 11(8):1–18.

[Lopez et al., 2011] Lopez, J., Holmbak-Petersen, R., Santiago, I., Gomez de Segura, I., and Barrey, E. (2011). Gait analysis using 3d accelerometry in horses sedated with xylazine. *Veterinary journal (London, England : 1997)*, 193:212–6.

[Mannini and Sabatini, 2014] Mannini, A. and Sabatini, A. (2014). Walking speed estimation using foot-mounted inertial sensors: comparing machine learning and strap-down integration methods. *Medical engineering and physics*, 36(10):1312–1321.

[Martin, 2015] Martin, P. (2015). *Saddle In Motion : back biomechanics of the ridden horse : analysis of the interactions between the saddle and the back, and application to the development of news prototypes of saddles.* Theses, Université Claude Bernard - Lyon I.

[Martinez Alvarez et al., 2011] Martinez Alvarez, F., Troncoso, A., Riquelme, J. C., and Aguilar Ruiz, J. S. (2011). Energy time series forecasting based on pattern sequence similarity. *IEEE Transactions on Knowledge and Data Engineering*, 23(8):1230–1243.

[Moe-Nilssen and Helbostad, 2004] Moe-Nilssen, R. and Helbostad, J. (2004). Estimation of gait cycle characteristics by trunk accelerometry. *Journal of biomechanics*, 37:121–6.

[Mousavi and Sørensen, 2017] Mousavi, S. and Sørensen, H. (2017). Functional logistic regression: a comparison of three methods. *Journal of Statistical Computation and Simulation*, 88:1–19.

[Mousavi and Sørensen, 2018] Mousavi, S. N. and Sørensen, H. (2018). Functional logistic regression: a comparison of three methods. *Journal of Statistical Computation and Simulation*, 88(2):250–268.

[Murphy et al., 2010] Murphy, J., Carr, H., and O'Neill, M. (2010). Animating horse gaits and transitions. In *Eurographics UK Symposium on Theory and Practice of Computer Graphics*.

[Peng and Müller, 2008] Peng, J. and Müller, H.-G. (2008). Distance-based clustering of sparsely observed stochastic processes, with applications to online auctions. *The Annals of Applied Statistics*, 2(3):1056–1077.

[Peng Wei, 2014] Peng Wei, Hongwei Tang, D. L. (2014). Functional logistic regression approach to detecting gene by longitudinal environmental exposure interaction in a case-control study. *Genet Epidemiol.*, 38(7):638–651.

[Pfau et al., 2016] Pfau, T., Boultbee, H., Davis, H., Walker, A., and Rhodin, M. (2016). Agreement between two inertial sensor gait analysis systems for lameness examinations in horses. *Equine Veterinary Education*, 28(4):203–208.

[Pfau et al., 2005] Pfau, T., Witte, T., and Wilson, A. (2005). A method for deriving displacement data during cyclical movement using inertial sensor. *Journal of Experimental Biology*, 208:2503–2514.

[Piccarreta, 2001] Piccarreta, R. (2001). A new measure of nominal-ordinal association. *Journal of Applied Statistics*, 28(1):107–120.

[R Core Team, 2017a] R Core Team (2017a). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

[R Core Team, 2017b] R Core Team (2017b). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

[Ramsay et al., 2009] Ramsay, J. O., Hooker, G., and Graves, S. (2009). *Functional Data Analysis with R and MATLAB*. Springer Publishing Company, Incorporated, 1st edition.

[Ramsay and Silverman, 2005] Ramsay, J. O. and Silverman, B. W. (2005). *Functional data analysis*. Springer Series in Statistics. Springer, New York, second edition.

[Sabatini and Mannini, 2016] Sabatini, A. M. and Mannini, A. (2016). Ambulatory assessment of instantaneous velocity during walking using inertial sensor measurements. *Sensors*, 16(12).

[Schmutz et al., 2018] Schmutz, A., Jacques, J., and Bouveyron, C. (2018). *funHDDC: Univariate and multivariate model-based clustering in group-specific functional subspaces*. R package version 2.2.0.

[Schwarz, 1978] Schwarz, G. (1978). Estimating the dimension of a model. *Ann. Statist.*, 6:461–464.

[Strobl et al., 2008] Strobl, C., Boulesteix, A.-L., Kneib, T., Augustin, T., and Zeileis, A. (2008). Conditional variable importance for random forests. *BMC Bioinformatics*, 9(307).

[Tan et al., 2008] Tan, H., Wilson, A., and Lowe, J. (2008). Measurement of stride parameters using a wearable gps and inertial measurement unit. *Journal of biomechanics*, 41(7):1398–1406.

[Tarpey and Kinateder, 2003] Tarpey, T. and Kinateder, K. (2003). Clustering functional data. *Journal of Classification*, 20(1):93–114.

[Therneau and Atkinson, 2018] Therneau, T. and Atkinson, B. (2018). *rpart: Recursive Partitioning and Regression Trees*. R package version 4.1-13.

[Wilson et al., 2006] Wilson, R., White, C., Quintana, F., Halsey, L., Liebsch, N., Martin, G., and Butler, P. (2006). Moving towards acceleration for estimates of activity-specific metabolic rate in free-living animals: the case of the cormorant. *Journal of Animal Ecology*, 75(5):1081–1090.

[Wing et al., 2005] Wing, M., Eklund, A., and Kellogg, L. (2005). Consumer-grade global positioning system (gps) accuracy and reliability. *Journal of Forestry*, 103(4):169.

[Zandbergen, 2009] Zandbergen, P. (2009). Accuracy of iphone locations: a comparison of assisted gps, wifi and cellular positionning. *Transactions in GIS*, 13(1):5–26.

[Zhang and Ye, 2008] Zhang, H. and Ye, Y. (2008). A tree-based method for modeling a multivariate ordinal response. *Statistics and Its Interface*, 1:169–178.

[Zhao et al., 2016] Zhao, Y., Brahms, M., Gerhard, D., and Barden, J. (2016). Stance phase detection for walking and running using an imu periodicity-based approach. In Chung, P., Soltoggio, A., Dawson, C. W., Meng, Q., and Pain, M., editors, *Proceedings of the 10th International Symposium on Computer Science in Sports (ISCSS)*, pages 225–232, Cham. Springer International Publishing.

[Zihajehzadeh et al., 2016] Zihajehzadeh, S., Loh, D., Lee, T. J., Hoskinson, R., and Park, E. J. (2016). A cascaded kalman filter-based gps / mems-imu integration for sports applications.

Appendix A

Codes R du chapitre 2

Codes utilisés pour générer les résultats des exemples

Import des données

```
library(readxl)
brut_sec<-read_excel(path = "Z:/SCHMUTZ Amandine/Data/vitesse et longueur foulée/All_Brut_par_sec.xlsx"
                     sheet = 1, col_names =TRUE,col_types = c(rep("text",4),rep("numeric",609)))

az<-brut_sec[,c(210:310)]
dim(az)
```

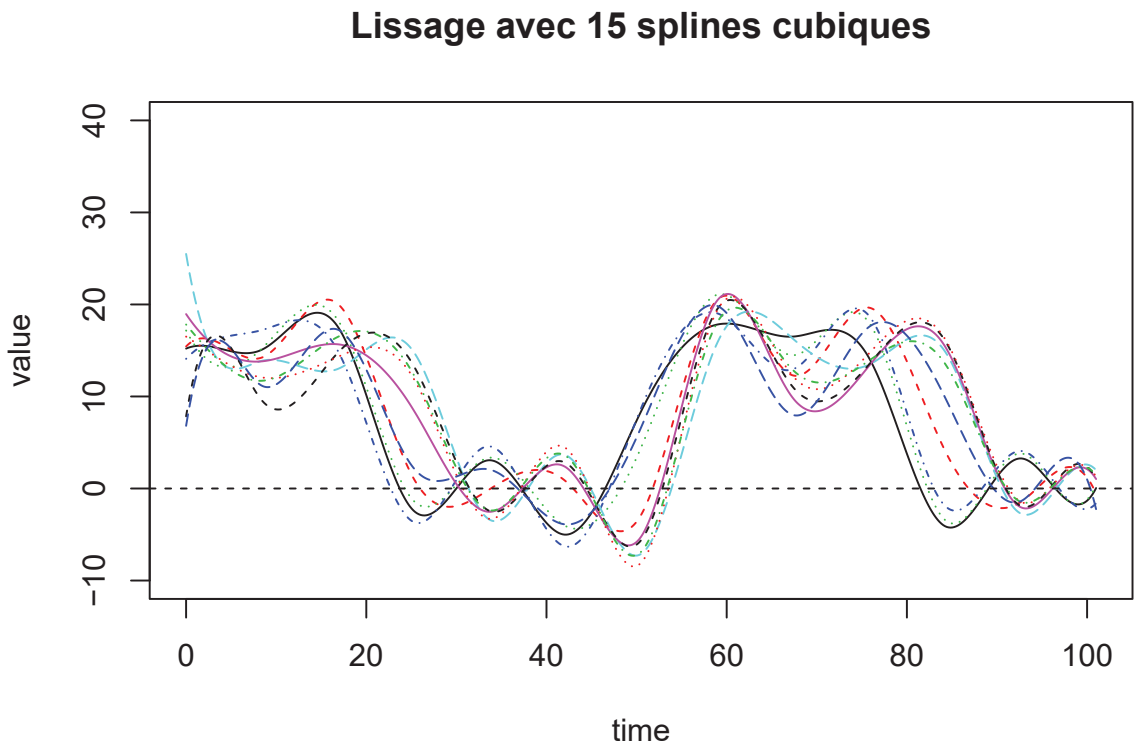
```
## [1] 2905  101
```

Lissage des données

```
library(fda)
```

Lissage avec 15 Bsplines cubiques.

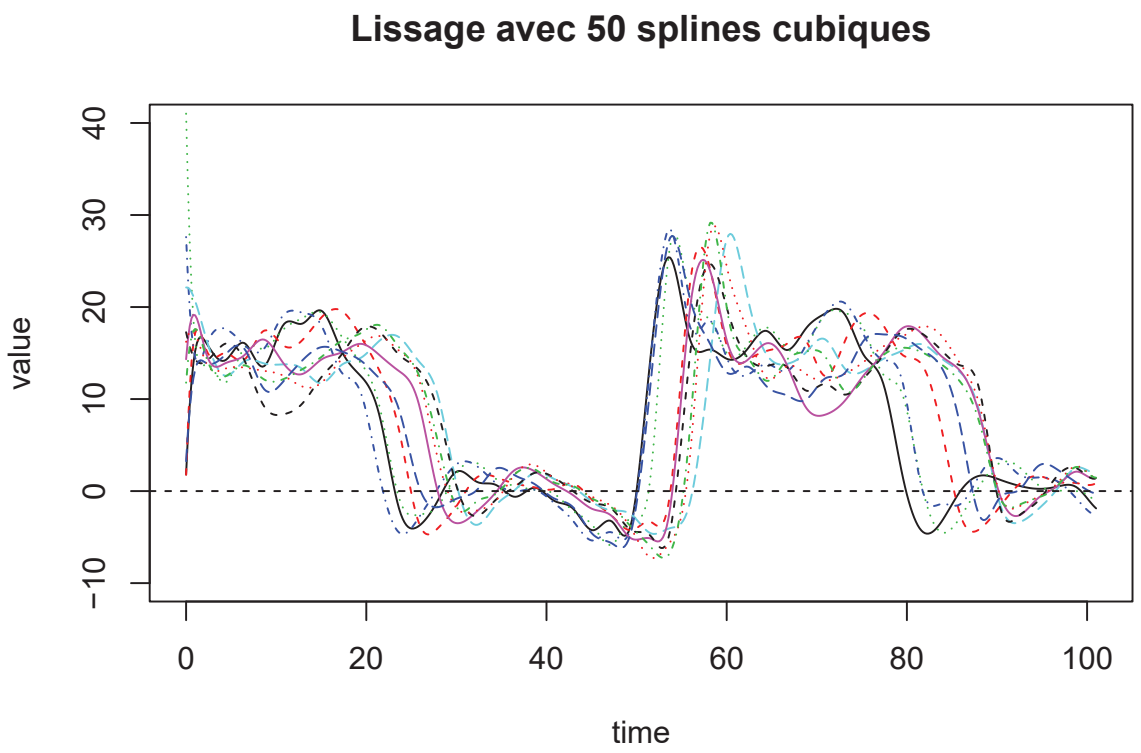
```
basis<-create.bspline.basis(rangeval=c(0,101), nbasis=15,norder=4)
az_smooth_select<-smooth.basis(y=t(az[1:10,]),fdParobj =basis)
plot(az_smooth_select,main="Lissage avec 15 splines cubiques",ylim=c(-10,40))
```



```
## [1] "done"
```

Lissage avec 50 Bsplines cubiques

```
basis<-create.bspline.basis(rangeval=c(0,101), nbasis=50,norder=4)
az_smooth_select<-smooth.basis(y=t(az[1:10,]),fdParobj =basis)
plot(az_smooth_select,main="Lissage avec 50 splines cubiques",ylim=c(-10,40))
```




```
## [1] "done"
```

Le lissage des données avec 50 Bsplines permet de conserver des variations plus fines des courbes que le lissage avec 15 Bsplines qui est plus grossier.

Alignement des données

Alignement des courbes par rapport à la courbe moyenne

```
# Specify smoothing weight
lambda.gr2.3 <- .03
# Specify what to smooth, namely the rate of change of curvature
Lfdobj<- 2
# Set up a B-spline basis for smoothing the discrete data
npoints <-101
norder <- 4
```

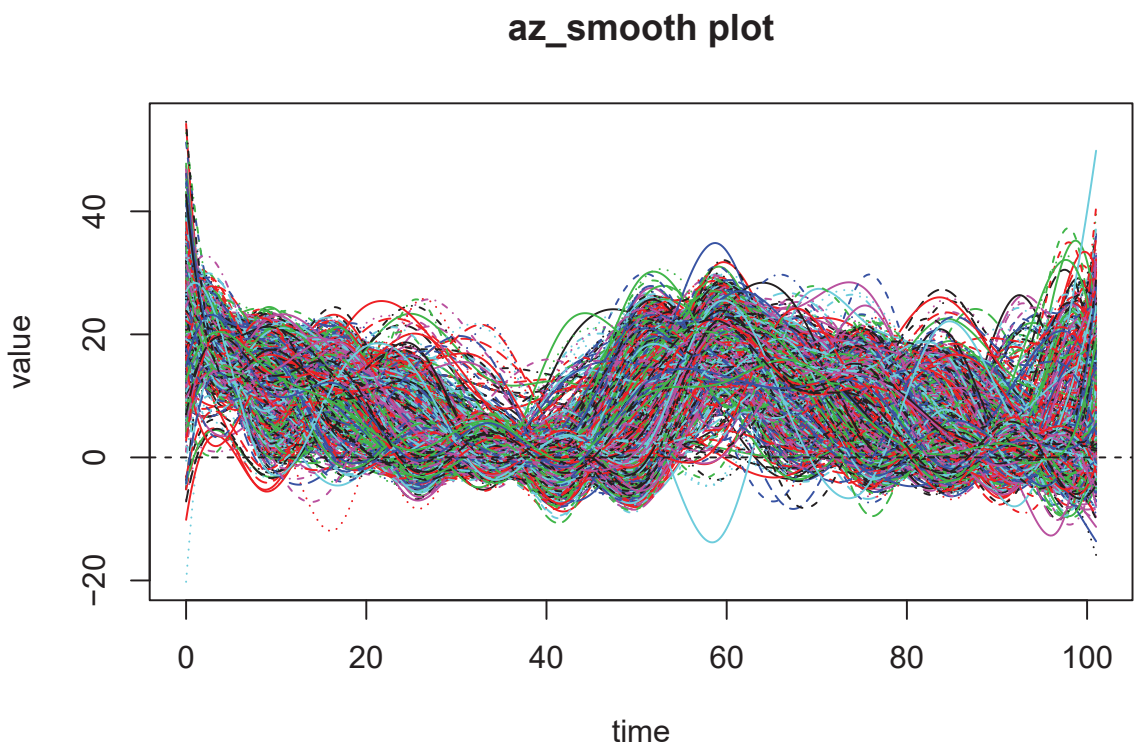
```r
nbasis <- npoints + norder - 2
rng <- range(1,101)
breaks1<-seq(1,101,by=1)
wbasis <- create.bspline.basis(rangeval=rng,
                                    nbasis=nbasis, norder=norder,
                                    breaks=breaks1)

cvec0 <- matrix(0,nbasis,1)
Wfd0  <- fd(cvec0, wbasis)
growfdPar2.3 <- fdPar(Wfd0, Lfdobj, lambda.gr2.3)
total_az2<-as.data.frame(t(az))
names(total_az2)<-c(seq(1,101,by=1))
total_az2<-fd(data.matrix(total_az2))

smBv <- total_az2

nInd <- 2905
#  Define the target function as the mean of the individuals curves
smBv0 = mean.fd(smBv[1:nInd])
```

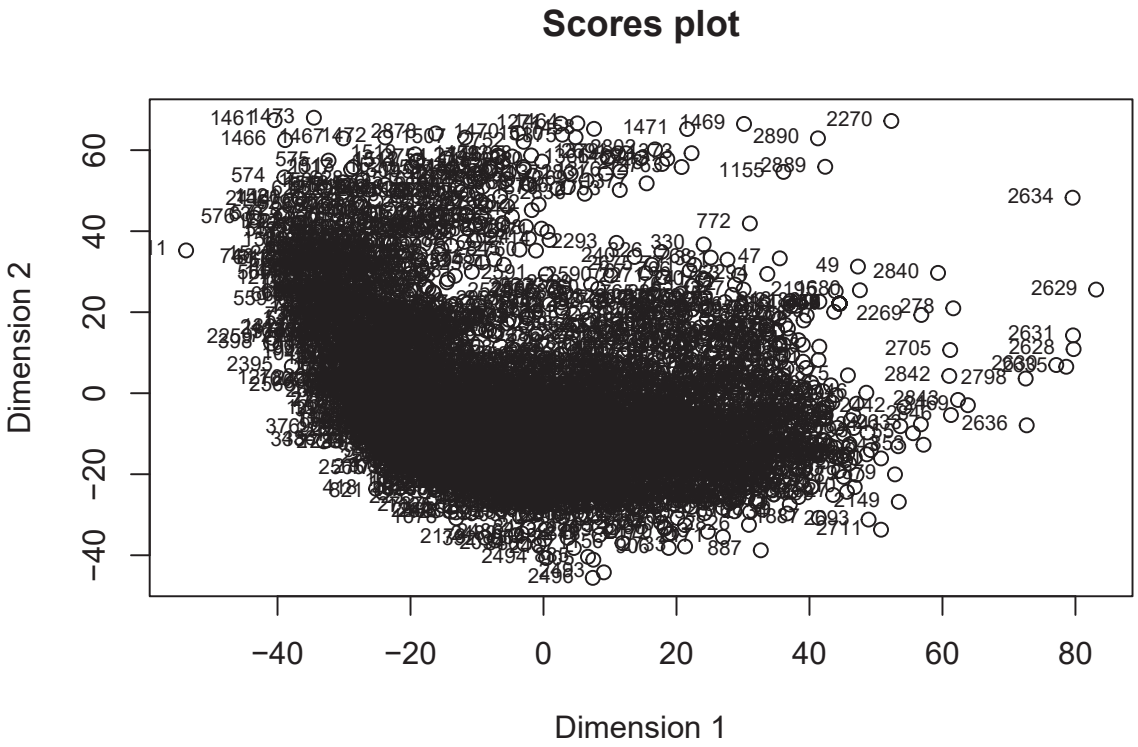
```r
smB.reg.0 <- register.fd(smBv0, smBv[1:nInd])
az_recal<-smB.reg.0$regfd$coefs

plotreg.fd(smB.reg.0)
```

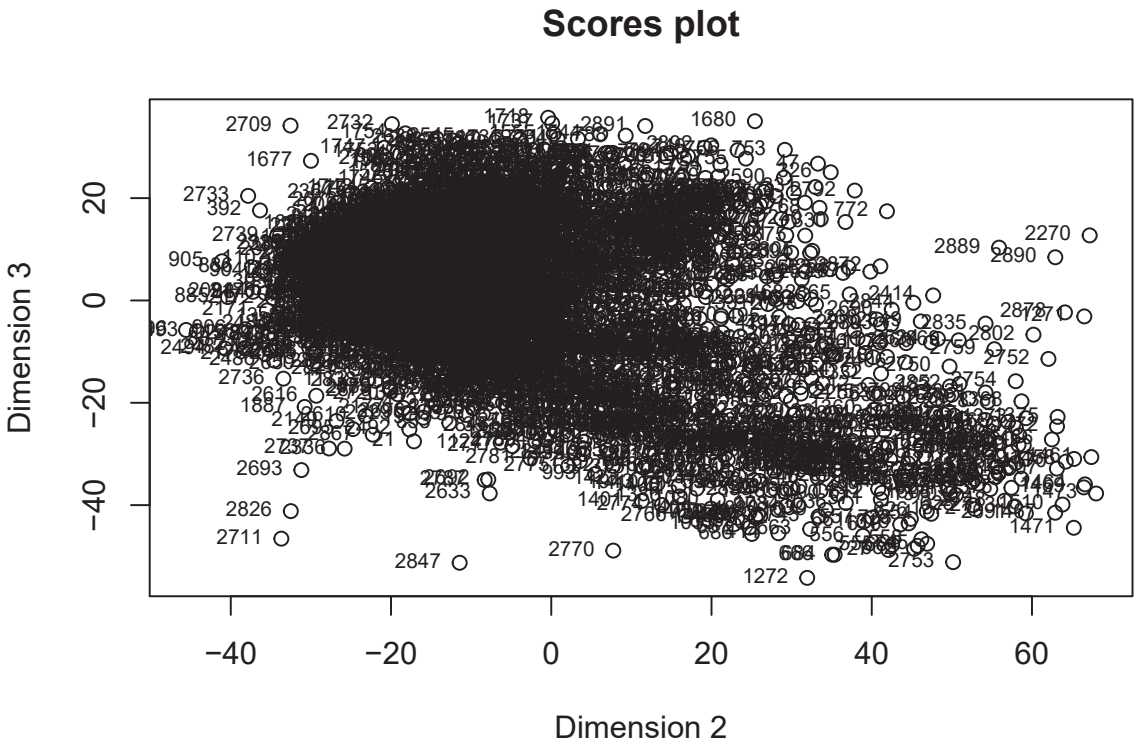
ACP univariée selon Az

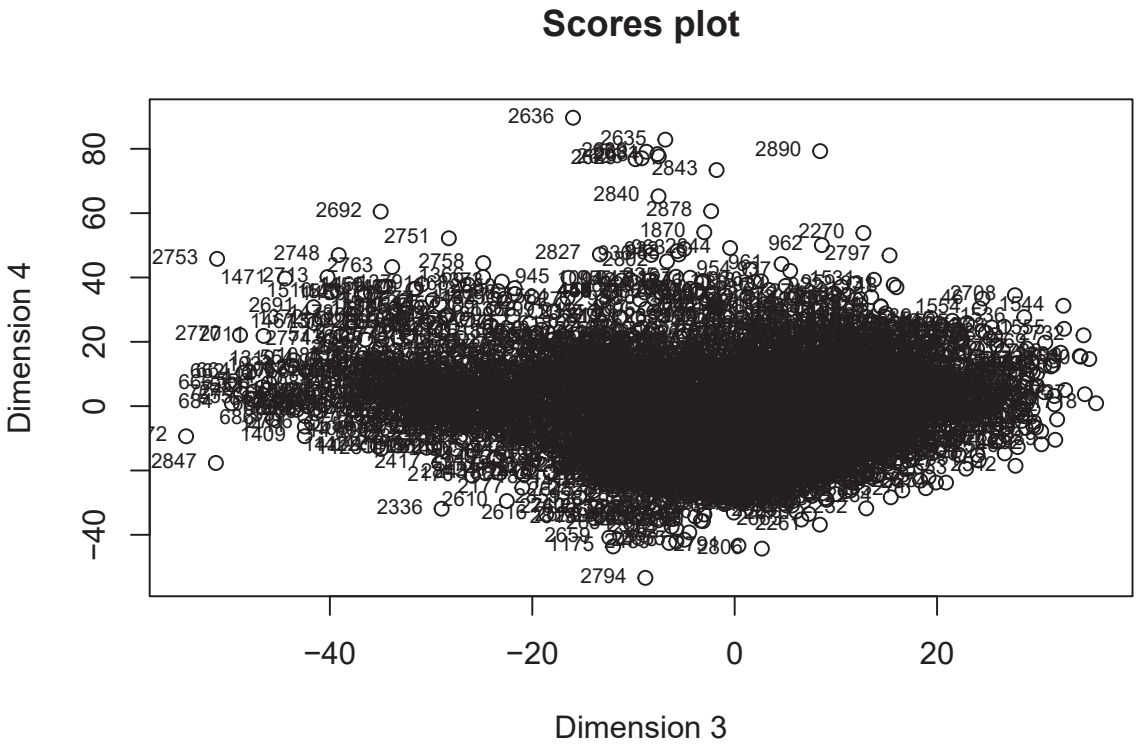
```r
library(funHDDC)
```

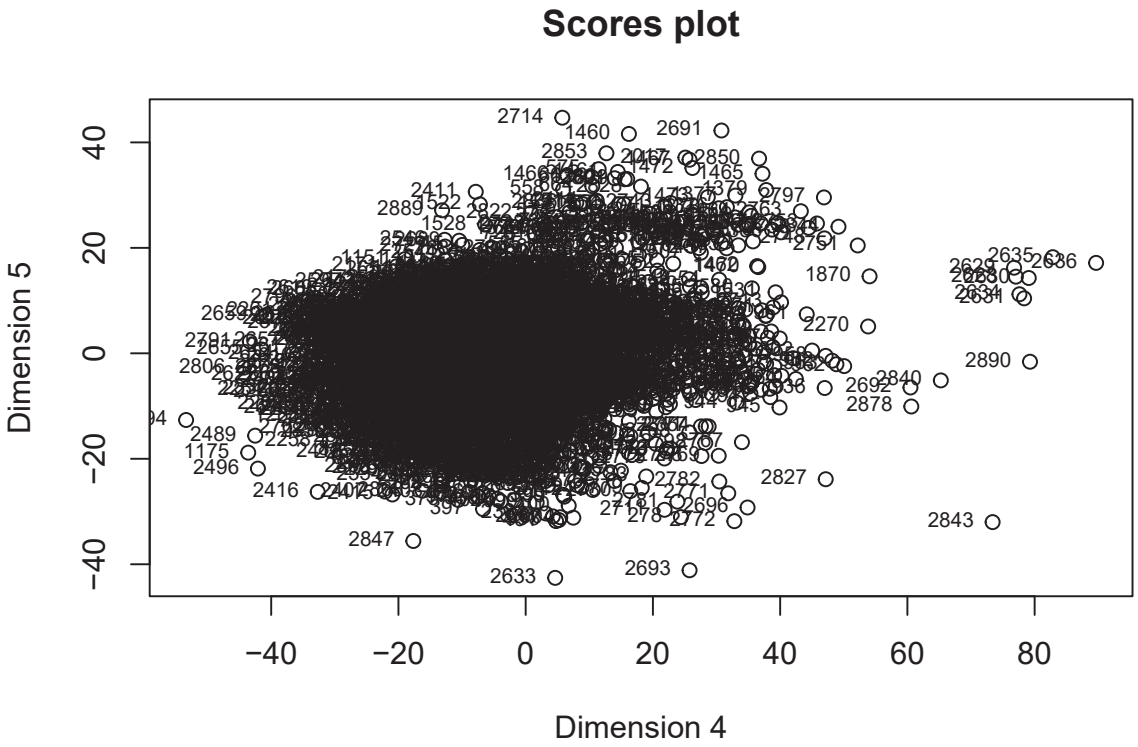
```r
basis<-create.bspline.basis(rangeval=c(0,101), nbasis=15,norder=4)
az_smooth<-smooth.basis(y=t(az),fdParobj =basis)$fd
res<-mfpca(az_smooth)
plot(res,nharm=4)
```

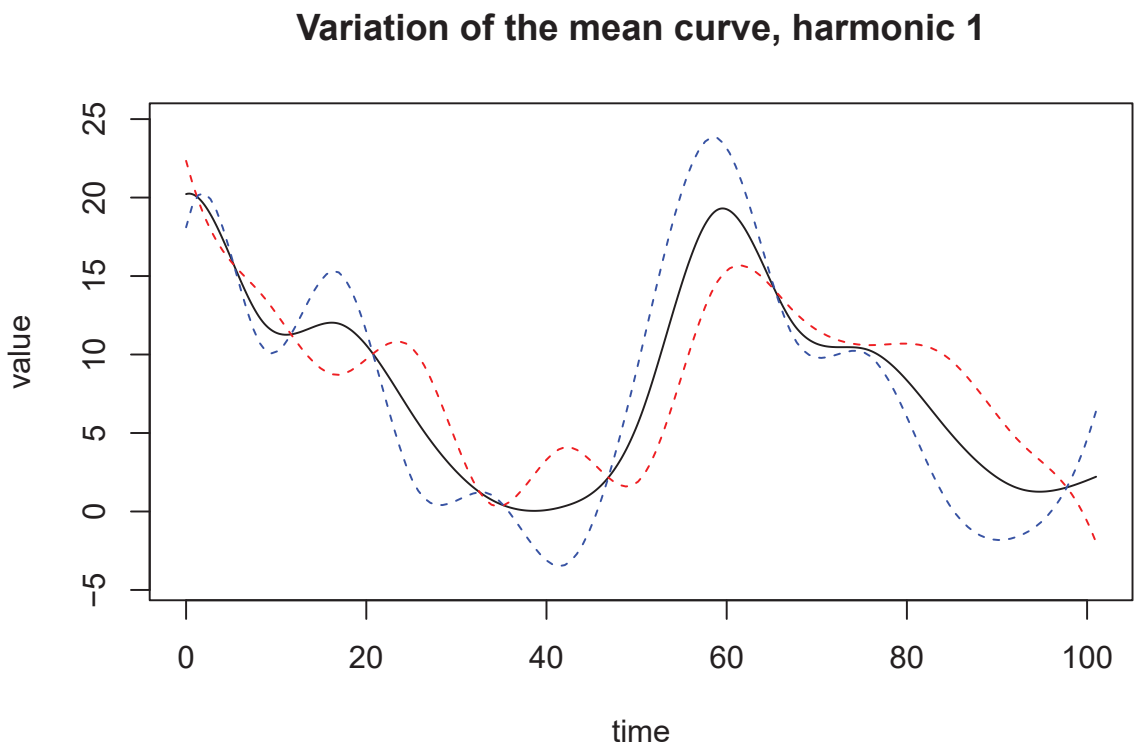

az_smooth plot
value
40
20
0
−20
0
20
40
60
80
100
time

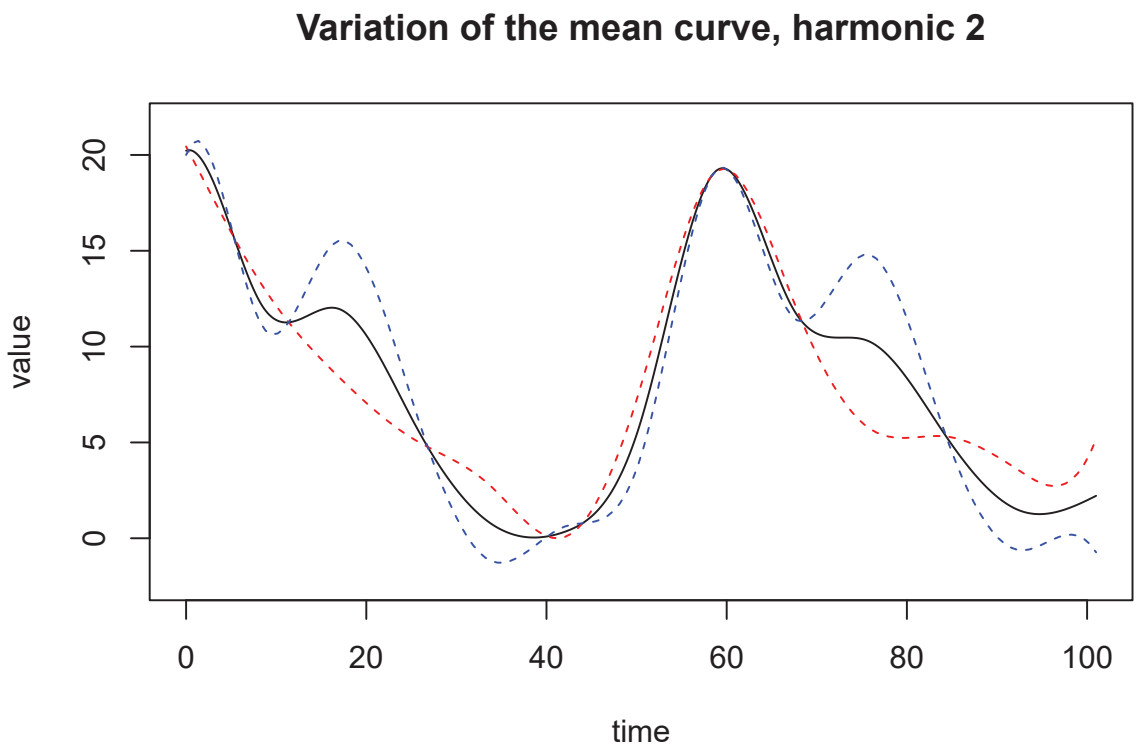

Scores plot
Dimension 1
Dimension 2
−40
−20
0
20
40
60
80

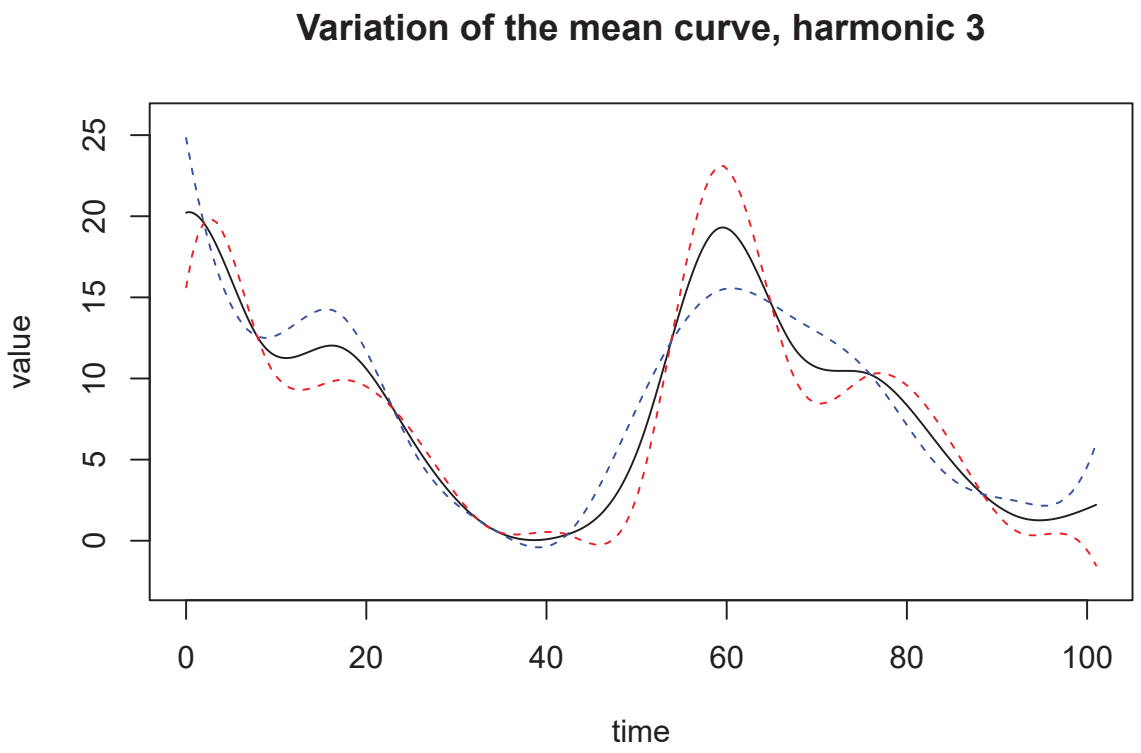

Scores plot
Dimension 3
Dimension 2
20
0
−20
−40
−40
−20
0
20
40
60
2709
2732
1680
1677
2733
392
2270
2889
2890
905
2826
2711
2693
2847
2770
1272
2753
1471
2736
2616
2835
2802
2752
2754
2633

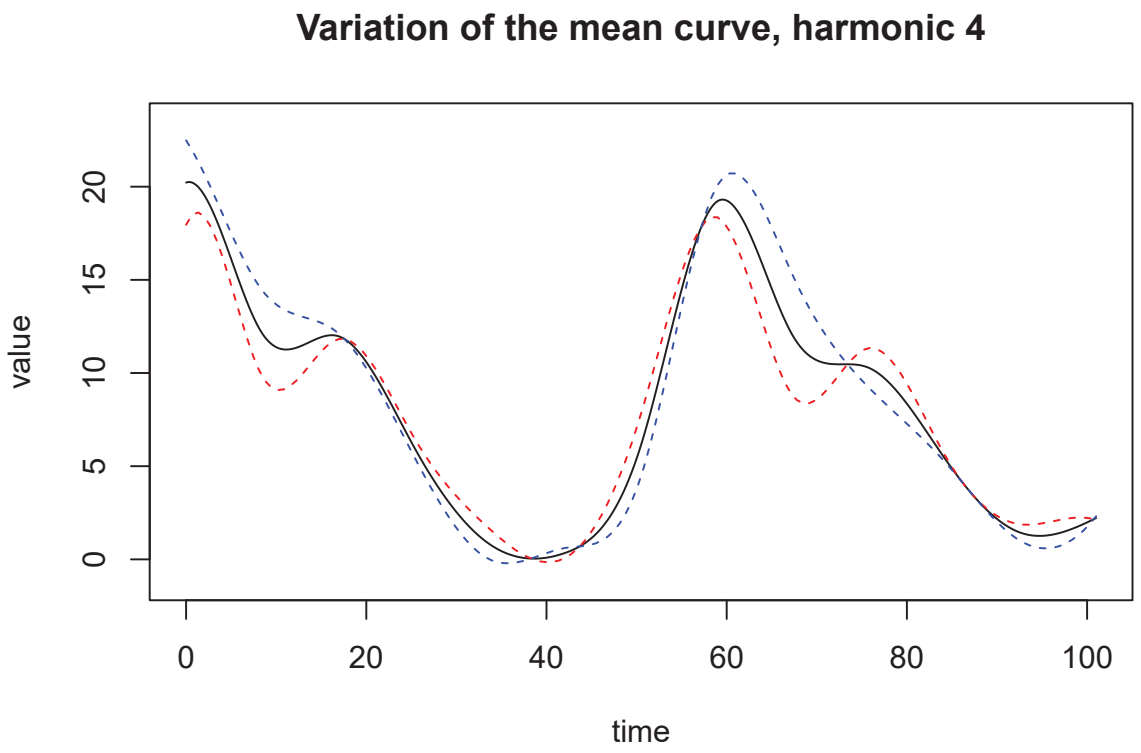

Scores plot
Dimension 4
Dimension 3
80
60
40
20
0
−40
−40
−20
0
20
2636
2635
2890
2843
2840
2878
2692
2751
1870
2270
962
2797
2753
2748
2763
2758
2827
945
961
2691
1544
2847
2336
2610
2794
1409
1175

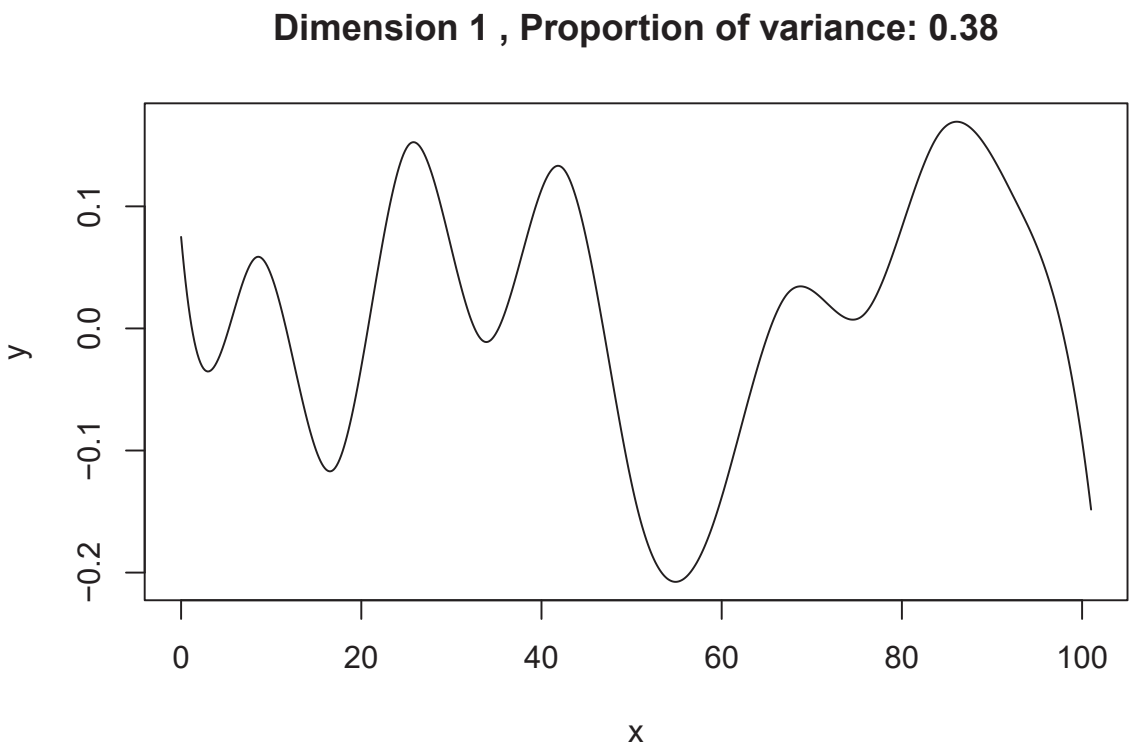

Scores plot
Dimension 5
Dimension 4
40
20
0
−20
−40
−40
−20
0
20
40
60
80
2714
1460
2691
2853
2850
1465
2797
2411
1528
1870
2635
2636
2270
2890
2840
2692
2878
2489
1175
2496
2416
2827
2843
2847
2633
2693

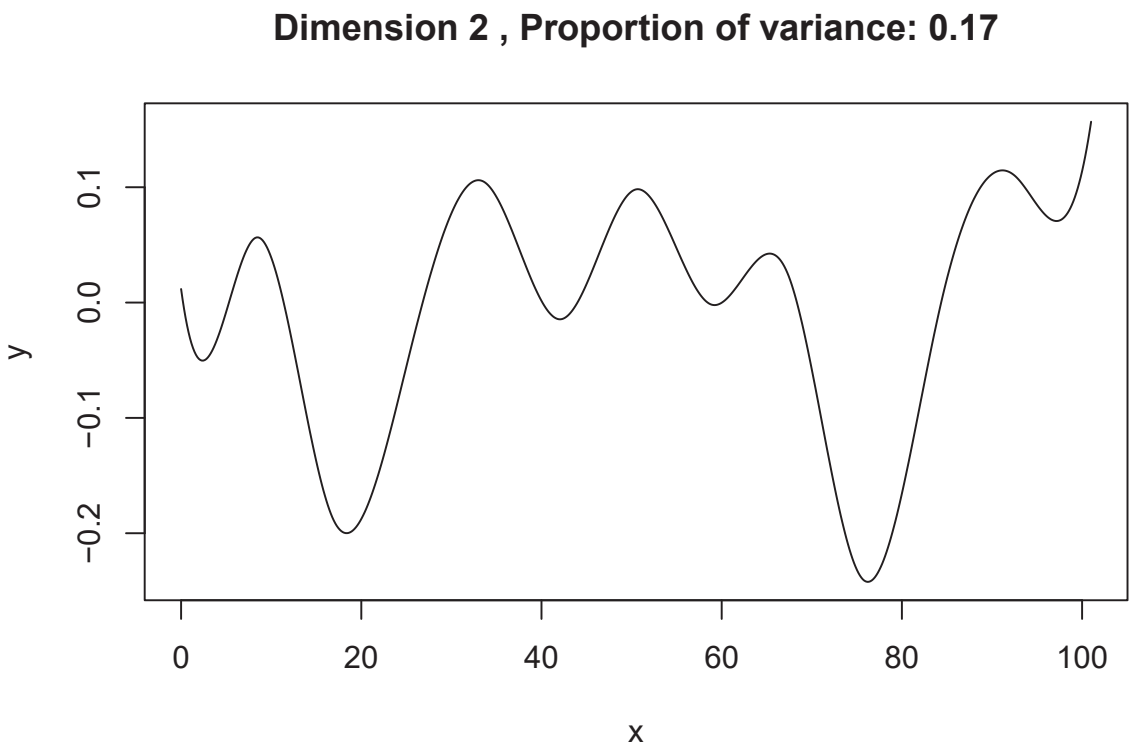

Variation of the mean curve, harmonic 1
value
time
25
20
15
10
5
0
−5
0
20
40
60
80
100

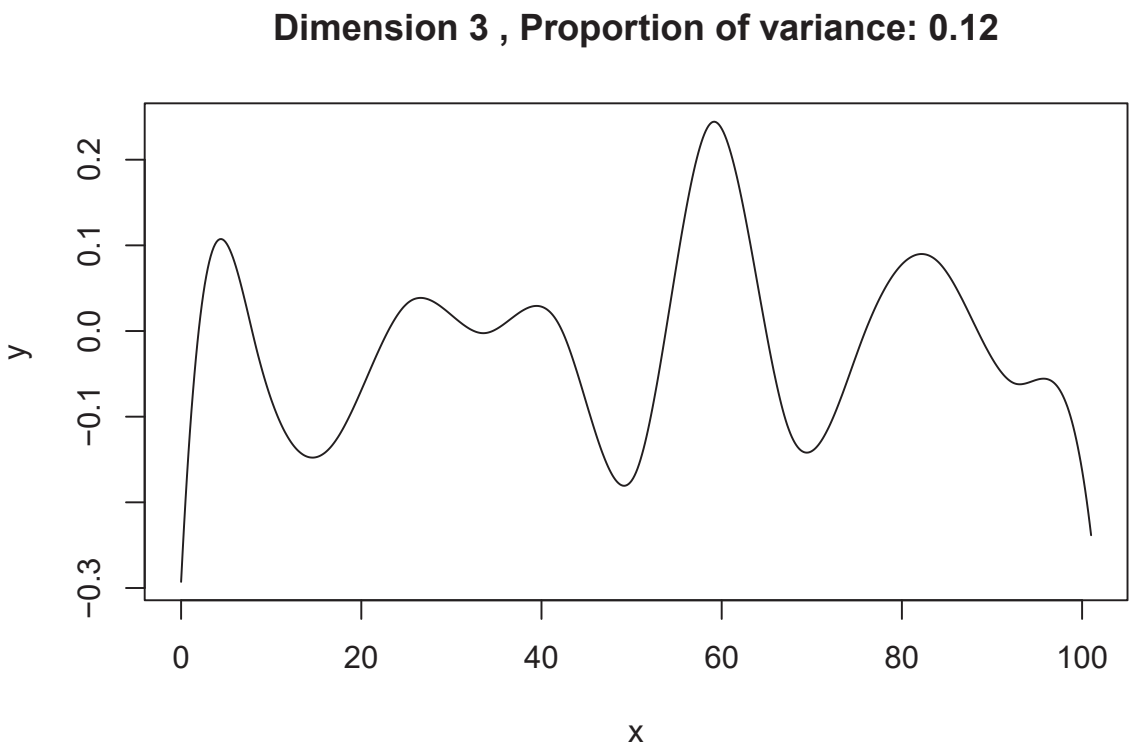

Variation of the mean curve, harmonic 2
value
0
5
10
15
20
0
20
40
60
80
100
time

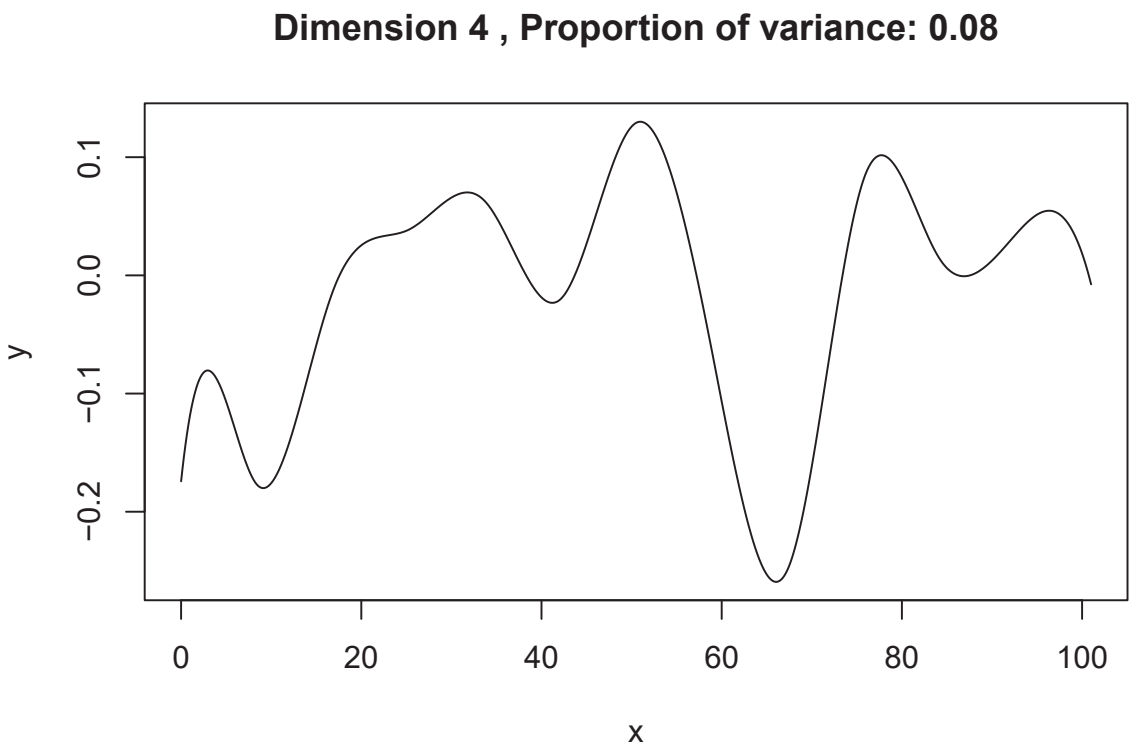

Variation of the mean curve, harmonic 3
value
0
5
10
15
20
25
0
20
40
60
80
100
time


Variation of the mean curve, harmonic 4
value
0
5
10
15
20
0
20
40
60
80
100
time


Dimension 1 , Proportion of variance: 0.38
y
0.1
0.0
−0.1
−0.2
0
20
40
60
80
100
x


Dimension 2 , Proportion of variance: 0.17
y
0.1
0.0
−0.1
−0.2
0
20
40
60
80
100
x


Dimension 3 , Proportion of variance: 0.12
y
0.2
0.1
0.0
−0.1
−0.3
0
20
40
60
80
100
x



Le premier graphe correspond aux courbes lissées. Les scores pour les 4 premières harmoniques sont ensuite représentés. Puis les variations des courbes moyennes et enfin les variations des premières fonctions propres. ##Clustering

```
library(fda)
library(funHDDC)
library(funFEM)
```

```
basis<-create.bspline.basis(rangeval=c(0,101), nbasis=15,norder=4)
az_smooth<-smooth.basis(y=t(az),fdParobj =basis)$fd
res.funfem<-funFEM(az_smooth,K=2:15,disp=TRUE)
```

```
## >> K = 2

## The "ward" method has been renamed to "ward.D"; note new "ward.D2"

## AkjBk    :    bic = -155459
##
## >> K = 3

## The "ward" method has been renamed to "ward.D"; note new "ward.D2"

## AkjBk    :    bic = -150827.2
##
## >> K = 4
```

```
## The "ward" method has been renamed to "ward.D"; note new "ward.D2"

## AkjBk    :    bic = -148585.2
##
## >> K = 5

## The "ward" method has been renamed to "ward.D"; note new "ward.D2"

## AkjBk    :    bic = -147082.7
##
## >> K = 6

## The "ward" method has been renamed to "ward.D"; note new "ward.D2"

## AkjBk    :    bic = -145455.8
##
## >> K = 7

## The "ward" method has been renamed to "ward.D"; note new "ward.D2"

## AkjBk    :    bic = -144120.2
##
## >> K = 8

## The "ward" method has been renamed to "ward.D"; note new "ward.D2"

## AkjBk    :    bic = -143605.5
##
## >> K = 9

## The "ward" method has been renamed to "ward.D"; note new "ward.D2"

## AkjBk    :    bic = -142232.3
##
## >> K = 10

## The "ward" method has been renamed to "ward.D"; note new "ward.D2"

## AkjBk    :    bic = -141695.3
##
## >> K = 11

## The "ward" method has been renamed to "ward.D"; note new "ward.D2"

## AkjBk    :    bic = -141180.9
##
## >> K = 12

## The "ward" method has been renamed to "ward.D"; note new "ward.D2"
```

```
## AkjBk    :    bic = -140930.4
##
## >> K = 13

## The "ward" method has been renamed to "ward.D"; note new "ward.D2"

## AkjBk    :    bic = -140186.6
##
## >> K = 14

## The "ward" method has been renamed to "ward.D"; note new "ward.D2"

## AkjBk    :    bic = -139547.1
##
## >> K = 15

## The "ward" method has been renamed to "ward.D"; note new "ward.D2"

## Error in D[k, ((d + 1):p), ((d + 1):p)] <- diag(rep(bk, p - d)) :
##   le nombre d'objets à remplacer n'est pas multiple de la taille du remplacement
##
## The best model is AkjBk with K = 14 ( bic = -139547.1 )
```

```
res.funhddc<-funHDDC(az_smooth,K=2:6)
```

```
## funHDDC:
##       model K threshold complexity         BIC             comment
## 1 AKJBKQKDK 2       0.2        117 -441,458.59
## 2 AKJBKQKDK 3       0.2        223 -536,495.96
## 3 AKJBKQKDK 4       0.2       <NA>        -Inf pop<min.individuals
## 4 AKJBKQKDK 5       0.2       <NA>        -Inf pop<min.individuals
## 5 AKJBKQKDK 6       0.2       <NA>        -Inf pop<min.individuals
##
## SELECTED: model  AKJBKQKDK  with  2  clusters.
## Selection Criterion: BIC.
```

Régression paramétrique univariée

```
library(fda)
```

Régression paramétrique univariée sur les données lissées avec 15 Bsplines. Calcul de l'erreur de prédiction par rapport à la vitesse de référence (mesurée à l'aide du système de tracking vidéo).

```
vitesse<-brut_sec$Vitesse_2D
basis<-create.bspline.basis(rangeval=c(0,101), nbasis=15,norder=4)
az_smooth<-smooth.basis(y=t(az),fdParobj =basis)$fd

model<-fRegress(vitesse~az_smooth)
test<-predict(model)
error<-abs(test-vitesse)
mean(error)
```

```
## [1] 0.691341
```

```r
sd(error)
```

```
## [1] 0.5792948
```

Augmentation du nombre de bases de lissage pour voir l'impact sur la qualité de la prédiction du modèle de régression.

```r
vitesse<-brut_sec$Vitesse_2D
basis<-create.bspline.basis(rangeval=c(0,101), nbasis=50,norder=4)
az_smooth<-smooth.basis(y=t(az),fdParobj =basis)$fd

model<-fRegress(vitesse~az_smooth)
test<-predict(model)
error<-abs(test-vitesse)
mean(error)
```

```
## [1] 0.6720104
```

```r
sd(error)
```

```
## [1] 0.5757848
```

On peut observer que dans le cas de ces données, l'augmentation du nombre de bases de fonctions ne permet pas de réduire de façon importante l'erreur de prédiction.

Régression non paramétrique univariée

Chargement des fonctions disponibles sur le site https://www.math.univ-toulouse.fr/staph/npfda/.

```r
semimetric.pca <- function(DATA1, DATA2, q)
{
  ################################################################
  # Computes between curves a pca-type semimetric based on the
  # functional principal components analysis method.
  #    "DATA1" matrix containing a first set of curves stored row by row
  #    "DATA2" matrix containing a second set of curves stored row by row
  #    "q" the retained number of principal components
  # Returns a "semimetric" matrix containing the semimetric computed
  # between the curves lying to the first sample and the curves lying
  # to the second one.
  ################################################################
  if(is.vector(DATA1)) DATA1 <- as.matrix(t(DATA1))
  if(is.vector(DATA2)) DATA2 <- as.matrix(t(DATA2))
  testfordim <- sum(dim(DATA1)==dim(DATA2))==2
  twodatasets <- T
  if(testfordim) twodatasets <- sum(DATA1==DATA2)!=prod(dim(DATA1))
  qmax <- ncol(DATA1)
  if(q > qmax) stop(paste("give a integer q smaller than ", qmax))
```

```
  n <- nrow(DATA1)
  COVARIANCE <- t(DATA1) %*% DATA1/n
  EIGENVECTORS <- eigen(COVARIANCE, sym = T)$vectors[, 1:q]
  COMPONENT1 <- DATA1 %*% EIGENVECTORS
  if(twodatasets) {
    COMPONENT2 <- DATA2 %*% EIGENVECTORS
  }
  else {
    COMPONENT2 <- COMPONENT1
  }
  SEMIMETRIC <- 0
  for(qq in 1:q)
    SEMIMETRIC <- SEMIMETRIC + outer(COMPONENT1[, qq], COMPONENT2[,
                                                                  qq], "-")^2
  return(sqrt(SEMIMETRIC))
}

funopare.knn.lcv <- function(Response, CURVES, PRED, ..., kind.of.kernel = "quadratic",
                             semimetric = "deriv")
{
  ################################################################
  # Performs functional prediction (regression) of a scalar response
  # from a sample of curves via the functional kernel estimator.
  # A local bandwidth (i.e. local number of neighbours) is selected
  # by a cross-validation procedure.
  #    "Response" vector containing the observations of the scalar
  #               response
  #    "CURVES" matrix containing the curves dataset (row by row)
  #             used for the estimating stage
  #    "PRED" matrix containing new curves stored row by row
  #           used for computing predictions
  #    "..." arguments needed for the call of the function computing
  #          the semi-metric between curves
  #    "kind.of.kernel" the kernel function used for computing of
  #                     the kernel estimator; you can choose
  #                     "indicator", "triangle" or "quadratic (default)
  #    "semimetric" character string allowing to choose the function
  #                 computing the semimetric;  you can select
  #                 "deriv" (default), "fourier", "hshift", "mplsr",
  #                 and "pca"
  # Returns a list containing:
  #    "Estimated.values" vector containing estimated reponses for
  #                        each curve of "CURVES"
  #    "Predicted.values" if PRED different from CURVES, this vector
  #                       contains predicted responses for each
  #                       curve of PRED
  #    "Bandwidths" vector containing the local data-driven bandwidths
  #                 for each curve in the matrix "CURVES"
  #    "Mse" mean squared error between estimated values and
  #          observed values
  ################################################################
  Response <- as.vector(Response)
  if(is.vector(PRED)) PRED <- as.matrix(t(PRED))
```

```r
testfordim <- sum(dim(CURVES)==dim(PRED))==2
twodatasets <- T
if(testfordim) twodatasets <- sum(CURVES==PRED)!=prod(dim(CURVES))
sm <- get(paste("semimetric.", semimetric, sep = ""))
if(semimetric == "mplsr")
  SEMIMETRIC1 <- sm(Response, CURVES, CURVES, ...)
else SEMIMETRIC1 <- sm(CURVES, CURVES, ...)
kernel <- get(kind.of.kernel)
n1 <- ncol(SEMIMETRIC1)
step <- ceiling(n1/100)
if(step == 0)
  step <- 1
Knearest <- seq(from = 10, to = n1 %/% 2, by = step)
kmax <- max(Knearest)
# the vector Knearest contains the sequence of the
# k-nearest neighbours used for computing the optimal bandwidth
Response.estimated <- 0
Bandwidth.opt <- 0
Knn1 <- 0
for(i in 1:n1) {
  Norm.diff <- SEMIMETRIC1[, i]
  # "norm.order" gives the sequence k_1, k_2,... such that
  # dq(X_{k_1},X_i) < dq(X_{k_2},X_i) < ...
  Norm.order <- order(Norm.diff)
  # "zz" contains dq(X_{k_2},X_i), dq(X_{k_3},X_i),...,
  # dq(X_{j_{kamx+2}},X_i)
  zz <- sort(Norm.diff)[2:(kmax + 2)]
  # Bandwidth[l-1] contains (dq(X_{j_l},X_i) +
  # dq(X_{j_l},X_i))/2 for l=2,...,kmax+2
  Bandwidth <- 0.5 * (zz[-1] + zz[ - (kmax + 1)])
  z <- zz[ - (kmax + 1)]
  ZMAT <- matrix(rep(z, kmax), nrow = kmax, byrow = T)
  UMAT <- ZMAT/Bandwidth
  KMAT <- kernel(UMAT)
  KMAT[col(KMAT) > row(KMAT)] <- 0
  Ind.curves <- Norm.order[2:(kmax + 1)]
  Ind.resp <- Response[Ind.curves]
  YMAT <- matrix(rep(Ind.resp, kmax), nrow = kmax, byrow = T)
  Hat.resp <- apply(YMAT[Knearest,  ] * KMAT[Knearest,  ], 1, sum
  )/apply(KMAT[Knearest,  ], 1, sum)
  Criterium <- abs(Hat.resp - Response[i])
  index <- order(Criterium)[1]
  Knn1[i] <- Knearest[index]
  Response.estimated[i] <- Hat.resp[index]
  Bandwidth.opt[i] <- Bandwidth[index]
}
Mse.estimated <- sum((Response.estimated - Response)^2)/n1
if(twodatasets) {
  if(semimetric == "mplsr")
    SEMIMETRIC2 <- sm(Response, CURVES, PRED, ...)
  else SEMIMETRIC2 <- sm(CURVES, PRED, ...)
  Bandwidth2 <- 0
  n2 <- ncol(SEMIMETRIC2)
```

```
    for(k in 1:n2) {
      Sm2k <- SEMIMETRIC2[, k]
      Sm2k.ord <- order(SEMIMETRIC2[, k])
      knn <- Knn1[Sm2k.ord[1]]
      Bandwidth2[k] <- sum(sort(Sm2k)[knn:(knn+1)])*0.5
    }
    KERNEL <- kernel(t(t(SEMIMETRIC2)/Bandwidth2))
    KERNEL[KERNEL < 0] <- 0
    KERNEL[KERNEL > 1] <- 0
    Denom <- apply(as.matrix(KERNEL), 2, sum)
    RESPKERNEL <- KERNEL * Response
    Response.predicted <- apply(as.matrix(RESPKERNEL), 2, sum)/
      Denom
    return(list(Estimated.values = Response.estimated,
                Predicted.values = Response.predicted, Bandwidths =
                  Bandwidth.opt, Mse = Mse.estimated))
  }
  else {
    return(list(Estimated.values = Response.estimated, Bandwidths
                = Bandwidth.opt, Mse = Mse.estimated))
  }
}

symsolve <- function(Asym, Bmat)
{
  #  Performed by J.O. Ramsay and available on its
  #  website http://ego.psych.mcgill.ca/misc/fda which contains a lot
  #  of functions for analyzing functional data in a different way:
  #   Solves the system ASYM X = BMAT for X where ASYM is symmetric
  #   Returns X
  n <- ncol(Asym)
  if(max(abs(Asym - t(Asym)))/max(abs(Asym)) > 1e-10)
    stop("Argument not symmetric.")
  Lmat <- chol(Asym, T)
  if(attr(Lmat, "rank") < n)
    stop("Argument singular.")
  Lmatinv <- solve(Lmat[, order(attr(Lmat, "pivot"))])
  Xmat <- Lmatinv %*% t(Lmatinv) %*% Bmat
  return(Xmat)
}

quadratic <- function(u)
{
  #  quadratic kernel
  1 - (u)^2
}
```

Réalisation de la régression non paramétrique fonctionnelle univariée, prédiction de la vitesse à partir du signal d'accélération selon Z.

```
az<-data.matrix(brut_sec[,c(209:309)])
indice<-sample(1:2905,round(0.2*2905),replace=FALSE)
resul.add1<-funopare.knn.lcv(Response=as.vector(brut_sec$Vitesse_2D[-indice]),
```

```r
                              CURVES=az[-indice,], PRED=az[indice,],semimetric = "pca",15)
error<-abs(brut_sec$Vitesse_2D[indice]-resul.add1$Predicted.values)
mean(error)
```

```
## [1] 0.4434359
```

```r
sd(error)
```

```
## [1] 0.4781474
```

On peut voir que la moyenne d'erreur de prédiction est plus faible que dans le cas de la régression paramétrique.

L'avantage de cette méthode est que l'on peut facilement booster les prédictions en ajoutant les effets des autres variables fonctionnelles mesurées (cf. Ferraty et Vieu, 2009).

```r
gy<-data.matrix(brut_sec[,c(411:511)])

y<-brut_sec[-indice,5]-resul.add1$Estimated.values
resul.add2<-funopare.knn.lcv(Response=as.vector(t(y)),CURVES=gy[-indice,],
                              PRED=gy[indice,],semimetric = "pca",15)
error<-abs(brut_sec$Vitesse_2D[indice]-(resul.add2$Predicted.values+resul.add1$Predicted.values))
mean(error)
```

```
## [1] 0.4058982
```

```r
sd(error)
```

```
## [1] 0.4301072
```

Ainsi la prise en compte de la vitesse angulaire selon y permet de diminuer l'erreur de prédiction.

Appendix B

Graphes de l'analyse de la variance fonctionnelle réalisée sur l'ensemble de la base de données à la partie II

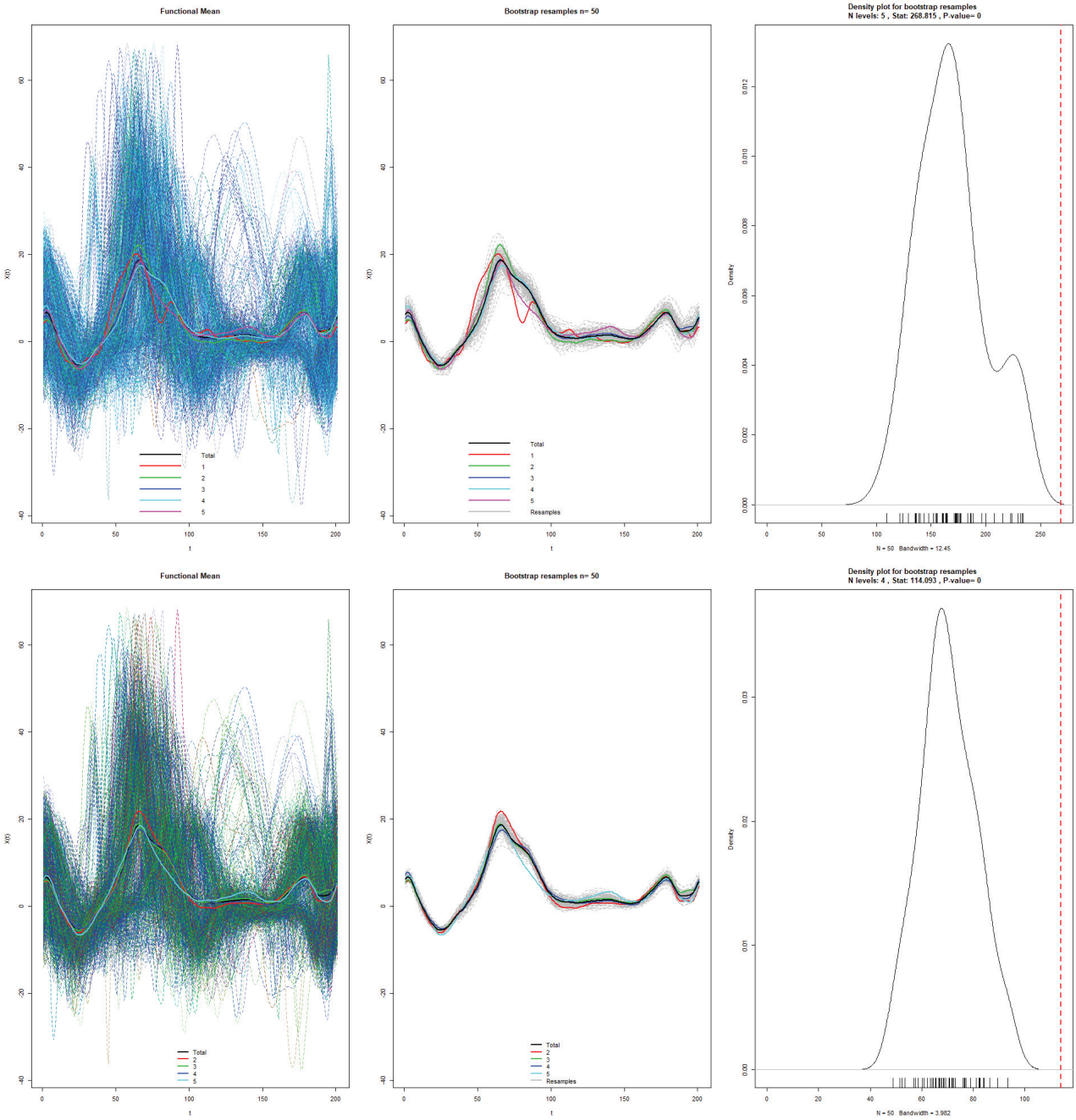


Figure B.1: Anova de l'accélération selon l'axe antéro-postérieur (Ax) sur les 5 notes (en haut) et sur les données sans la note 1 (en bas). Le trait noir correspond à la moyenne globale toutes notes confondues, les traits de couleur correspondent à la fonction moyenne par note.

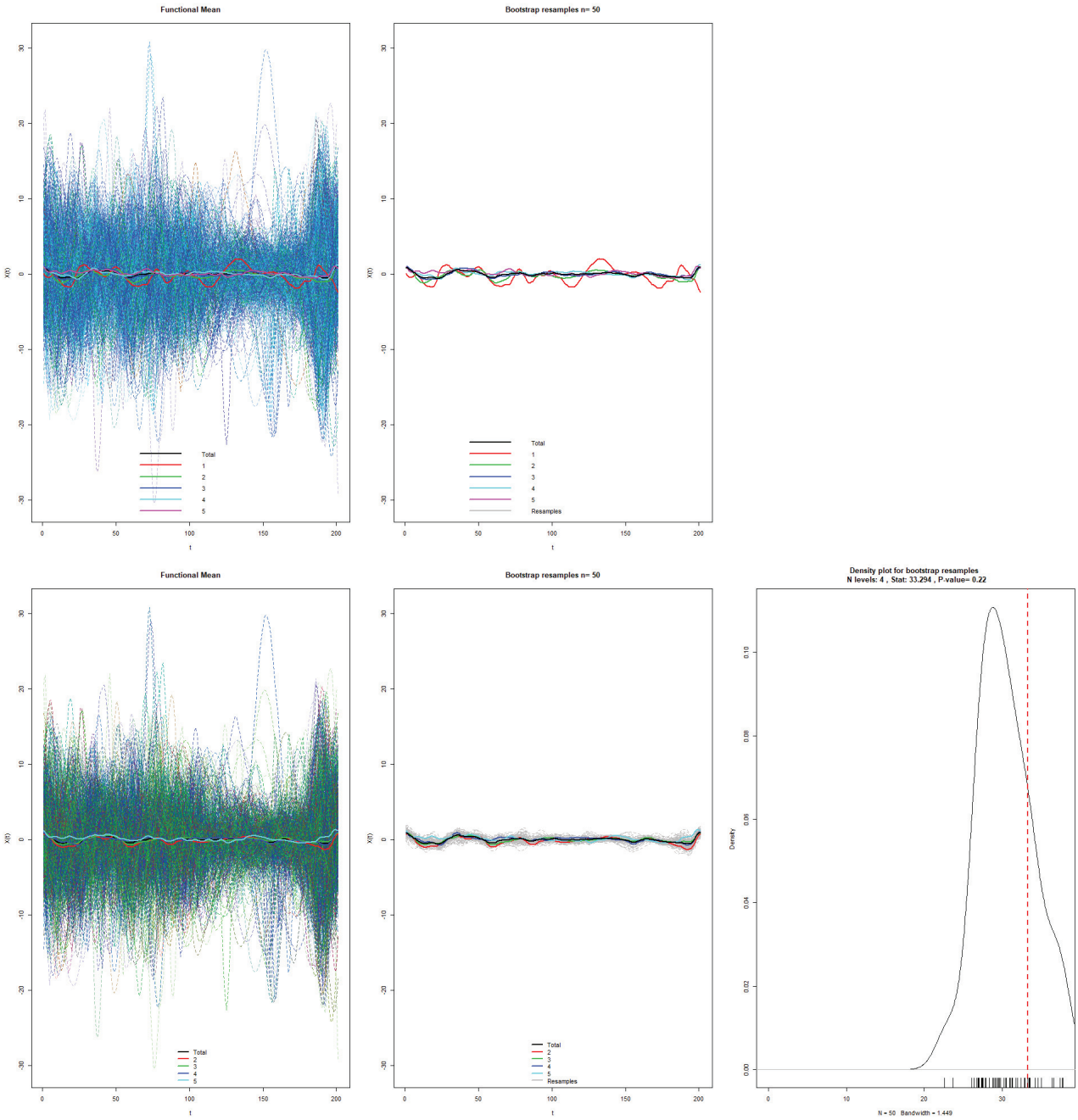


Figure B.2: Anova de l'accélération selon la direction medio-latérale (Ay) sur les 5 notes (en haut) et les données sans la note 1 (en bas). Le trait noir correspond à la moyenne globale toutes notes confondues, les traits de couleur correspondent à la fonction moyenne par note.

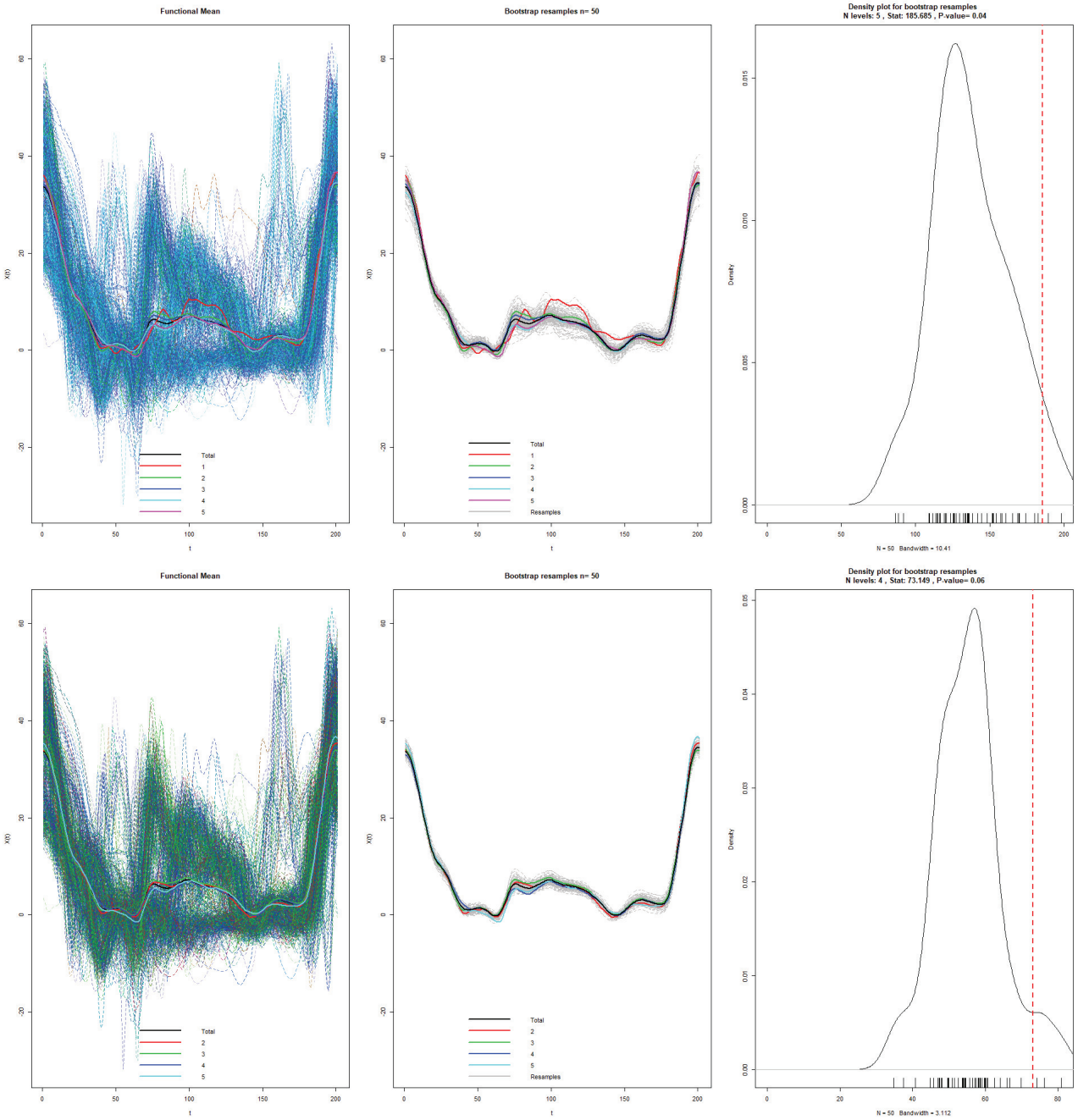


Figure B.3: Anova de l'accélération selon la direction dorso-ventrale (Az) sur les 5 notes (en haut) et les données sans la note 1 (en bas). Le trait noir correspond à la moyenne globale toutes notes confondues, les traits de couleur correspondent à la fonction moyenne par note.

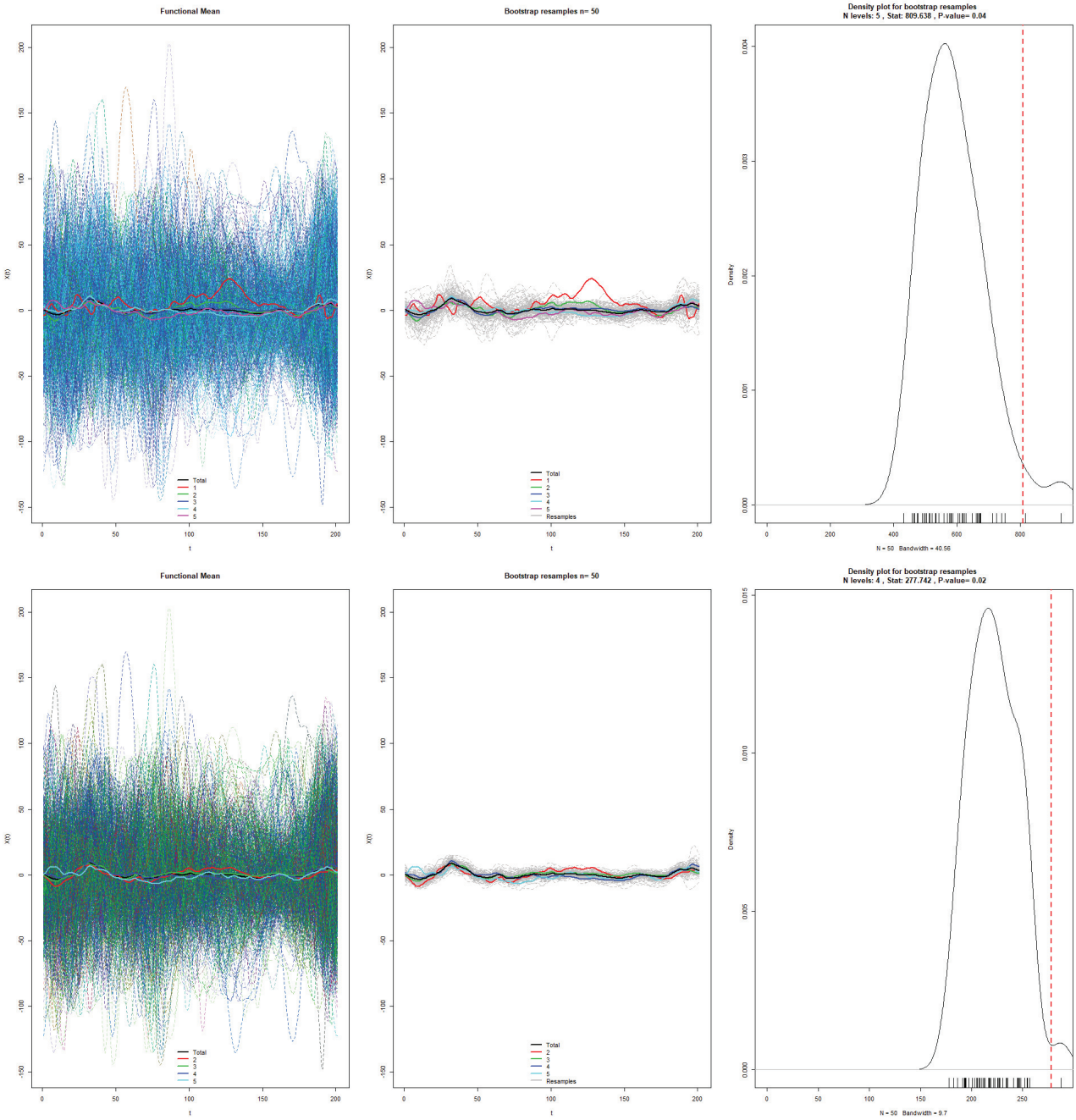


Figure B.4: Anova de la vitesse angulaire selon la direction antéro-postérieure (Gx) et qualité du saut sur les 5 notes (en haut) et la base de données sans la note 1 (en bas). Le trait noir correspond à la moyenne globale toutes notes confondues, les traits de couleur correspondent à la fonction moyenne par note.

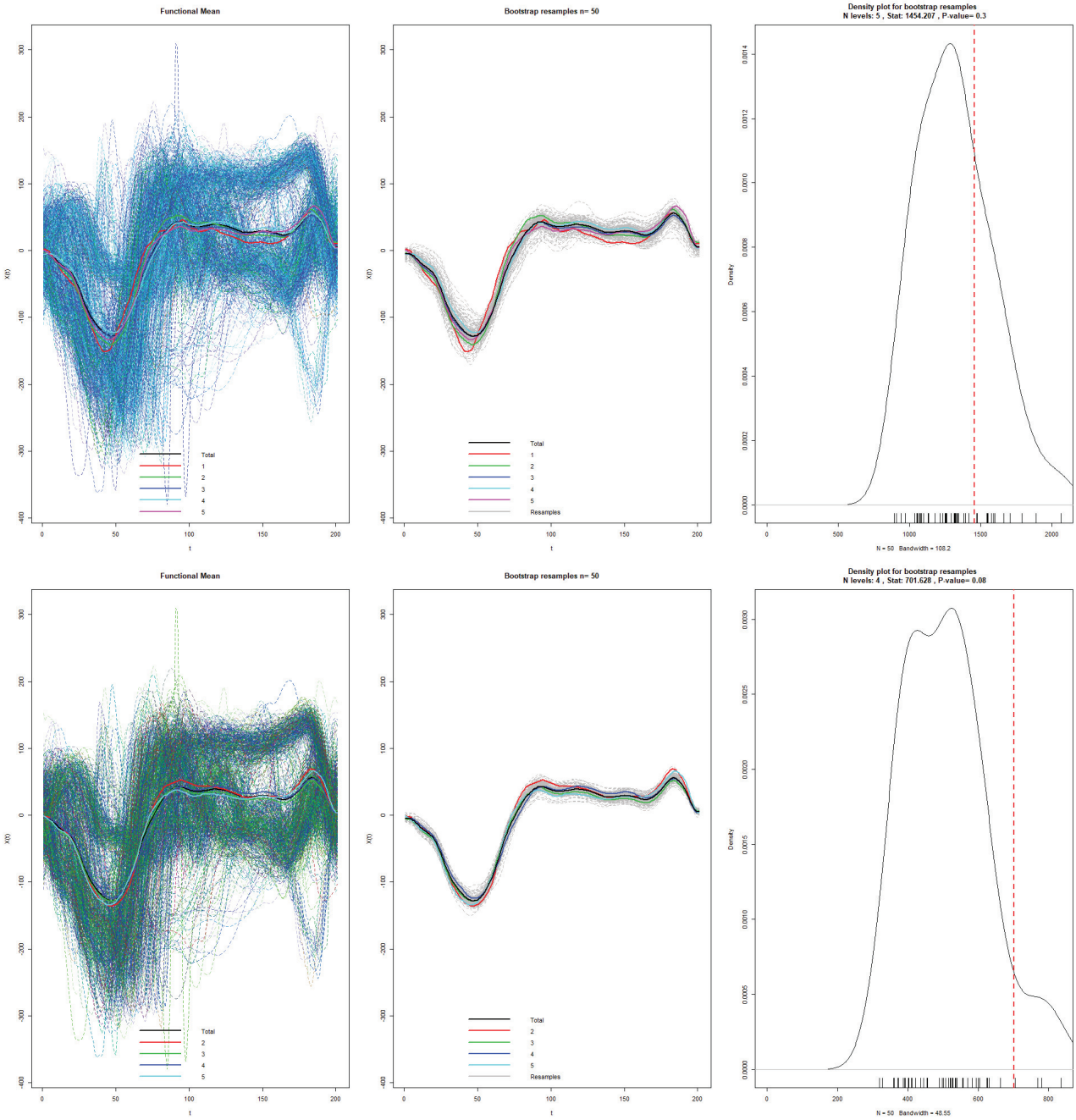


Figure B.5: Anova de la vitesse angulaire selon la direction médio-latérale (Gy) et qualité du saut sur les 5 notes (en haut) et la base de données sans la note 1 (en bas). Le trait noir correspond à la moyenne globale toutes notes confondues, les traits de couleur correspondent à la fonction moyenne par note.

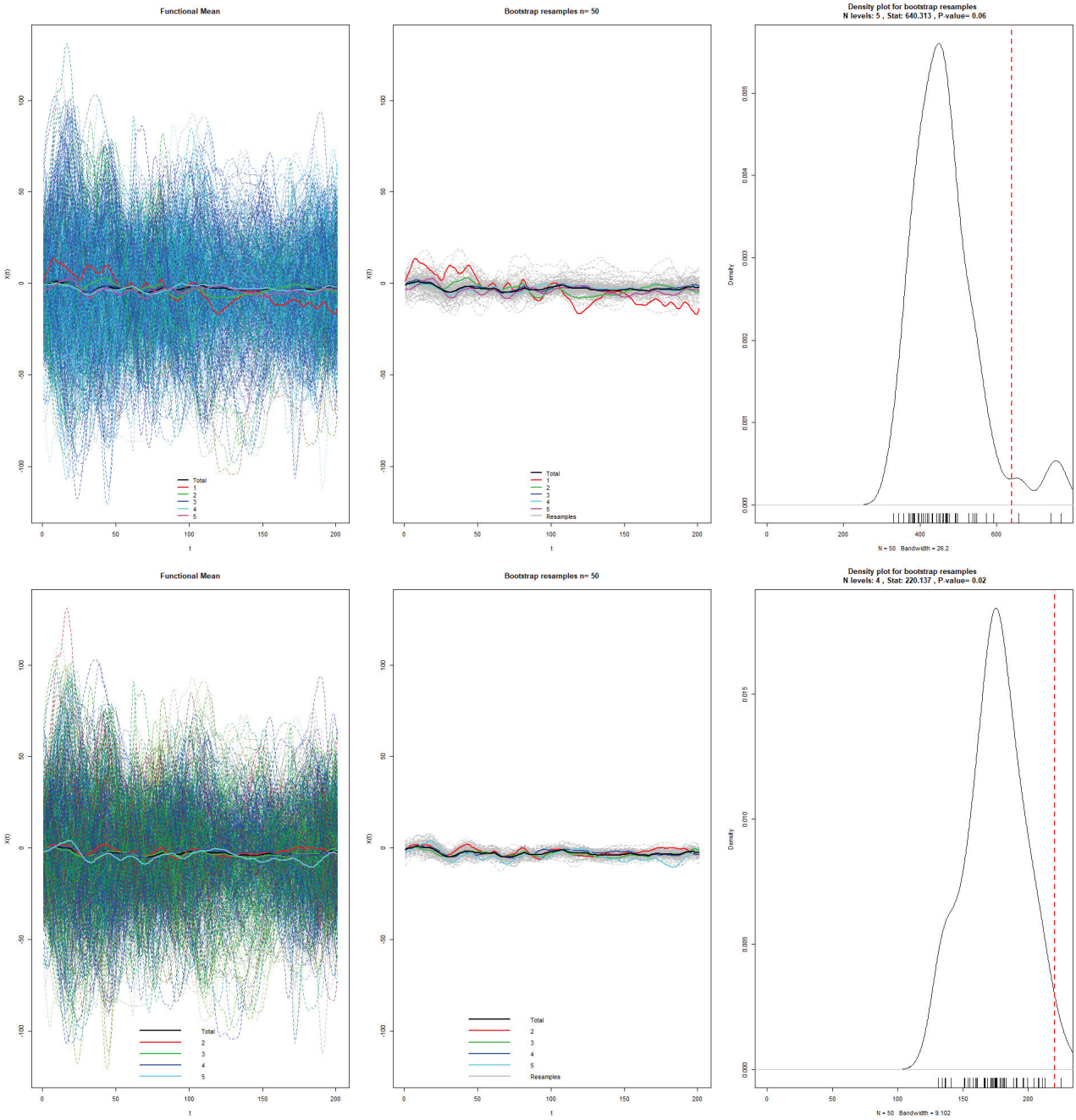


Figure B.6: Anova de la vitesse angulaire selon la direction dorso-ventrale (Gz) et qualité du saut sur les 5 notes (en haut) et la base de données sans la note 1 (en bas). Le trait noir correspond à la moyenne globale toutes notes confondues, les traits de couleur correspondent à la fonction moyenne par note.

Appendix C

Analyse de chaque phase de saut prise séparément, partie II

1 Analyse de la variance des données d'abord

L'analyse de la variance réalisée sur chacun des signaux et l'ensemble des 5 notes met en évidence des différences entre les courbes moyennes pour chaque signal, mais ces différences sont significatives uniquement pour les trois accélérations (cf. Table C.1 Figures C.1,C.2,C.3,C.4, C.5,C.6).

Variable	P-value analyse globale
Ax	0.04
Ay	0.03
Az	0.01
Gx	0.2
Gy	0.09
Gz	0.1

Table C.1: P-values associées à l'anova fonctionnelle

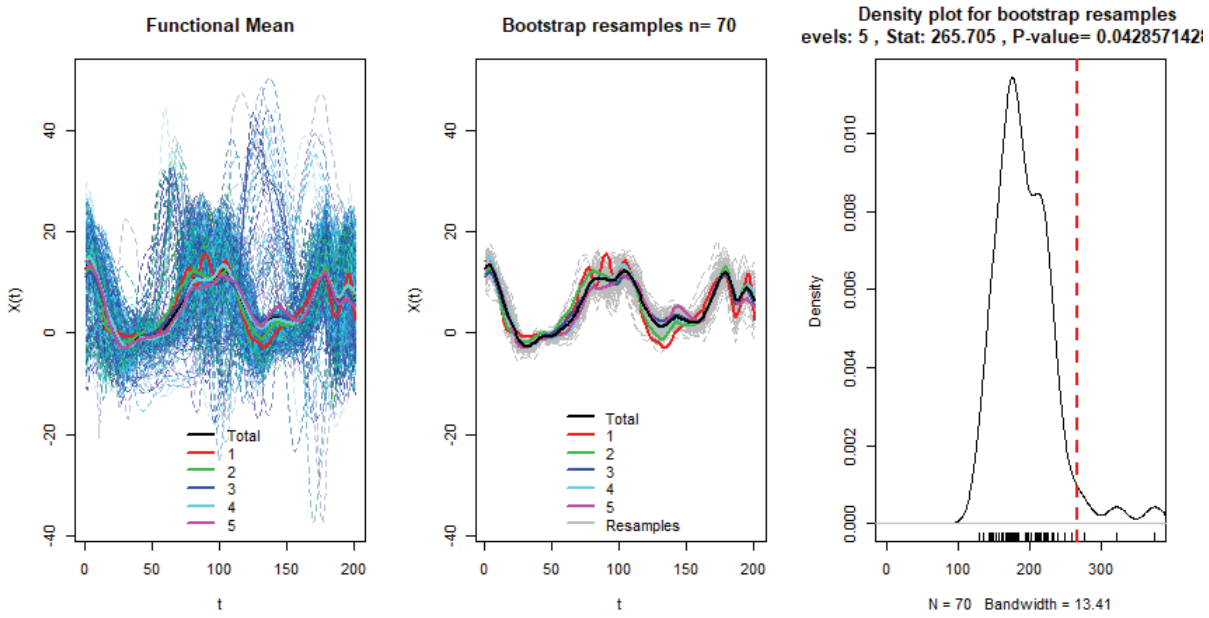


Figure C.1: Anova variable Ax et qualité du saut sur les 5 notes

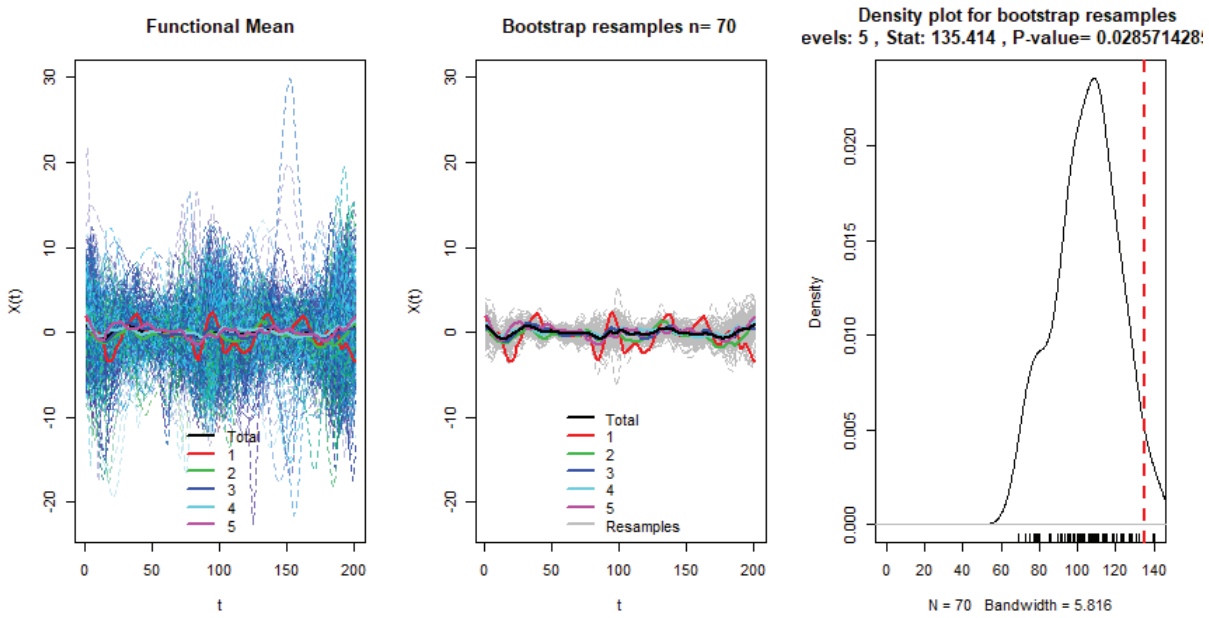


Figure C.2: Anova variable Ay et qualité du saut sur les 5 notes

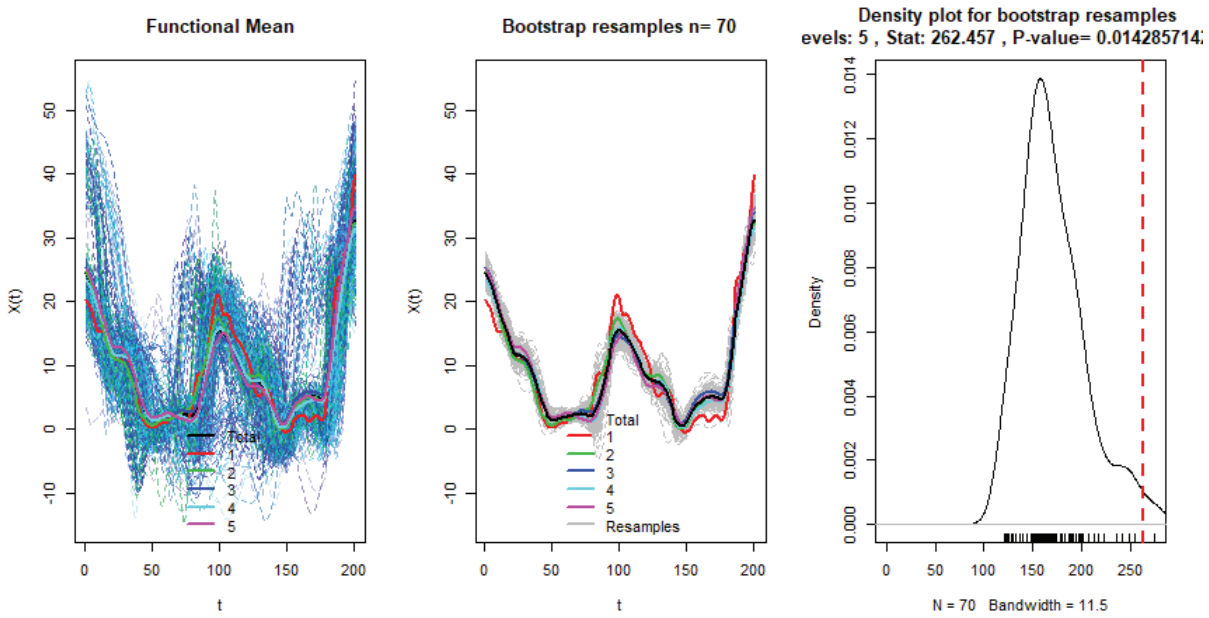


Figure C.3: Anova variable Az et qualité du saut sur les 5 notes

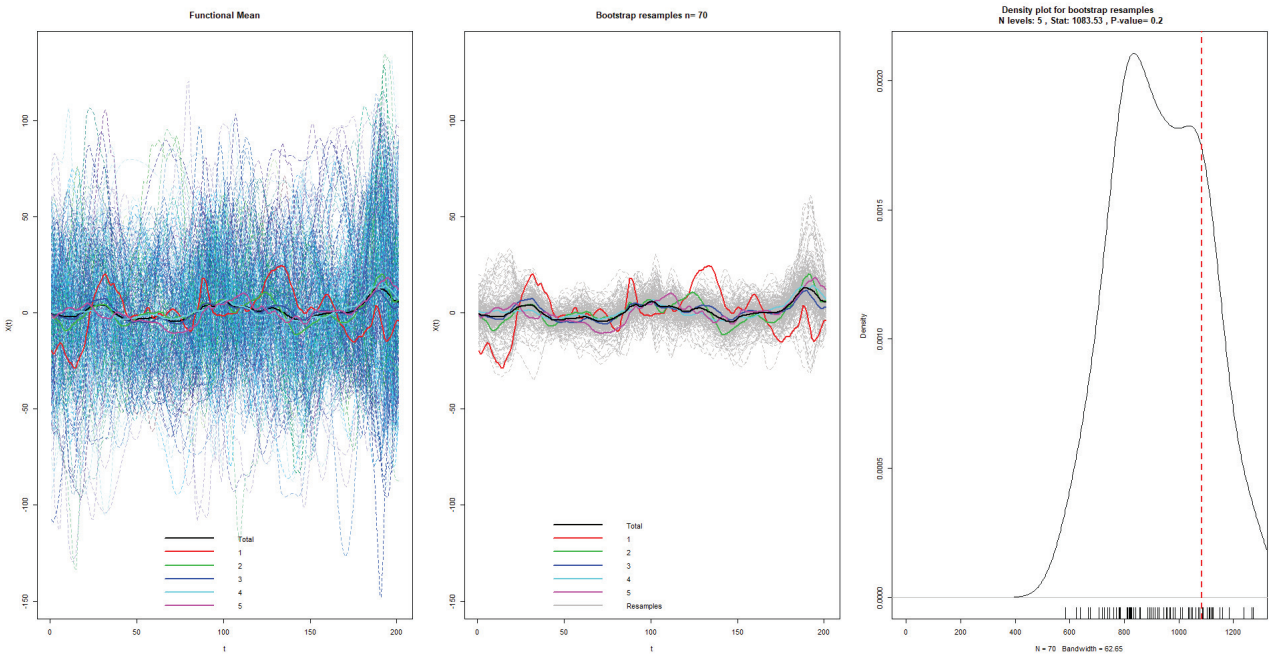


Figure C.4: Anova variable Gx et qualité du saut sur les 5 notes

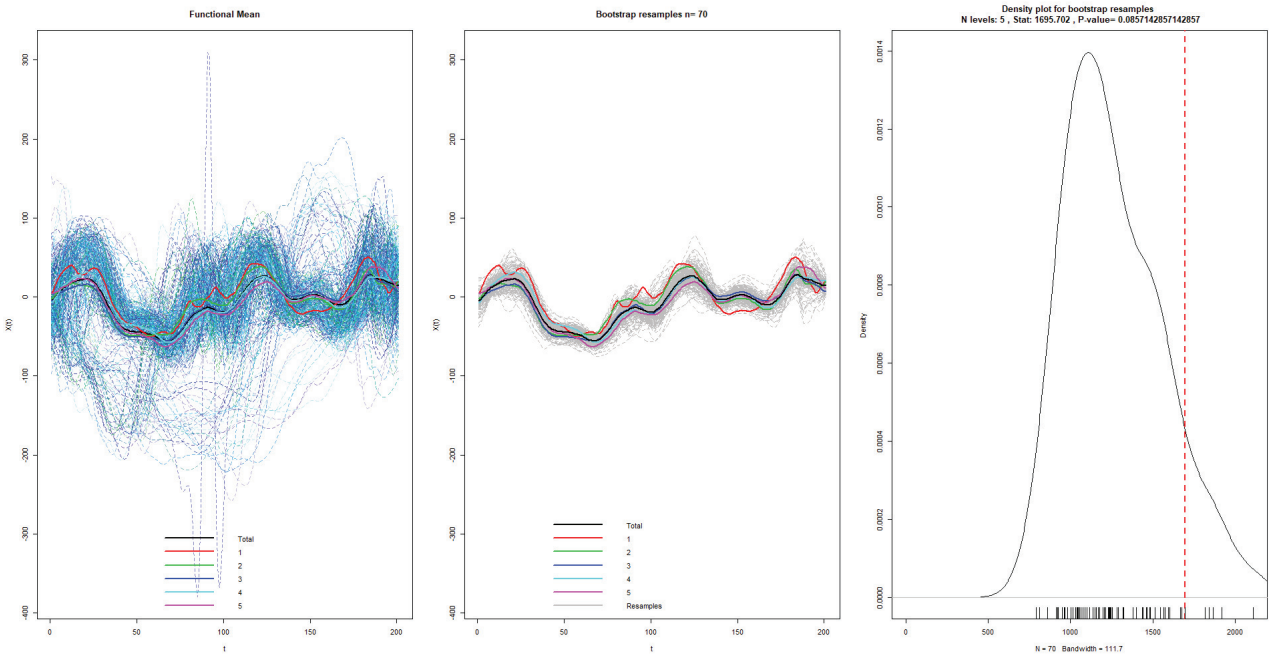


Figure C.5: Anova variable Gy et qualité du saut sur les 5 notes

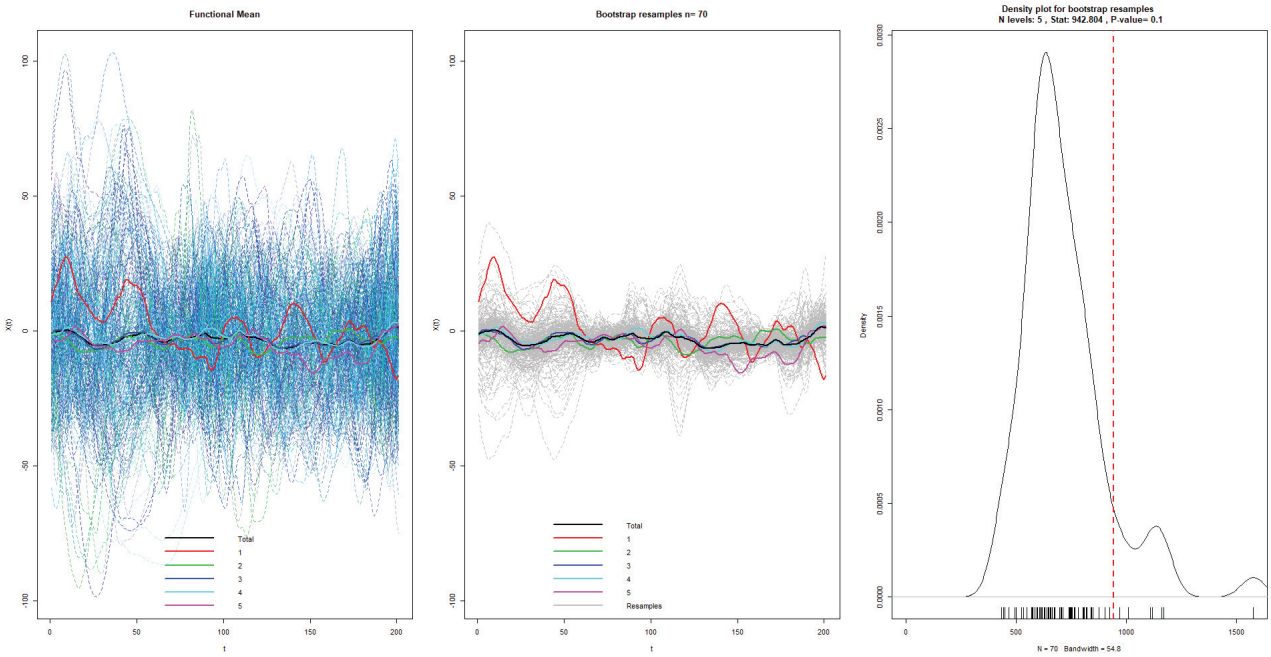


Figure C.6: Anova variable Gz et qualité du saut sur les 5 notes

2 Analyse de la variance des données de planer

L'analyse de la variance réalisée sur chacun des signaux et l'ensemble des 5 notes met en évidence des différences significatives entre les courbes moyennes de chaque signal pour les accélérations Ax, Az et la vitesse angulaire Gx (cf. Table C.2 et Figures C.7,C.8,C.9,C.10, C.11,C.12 en Annexe).

Variable	P-value analyse globale
Ax	0.04
Ay	0.07
Az	0
Gx	0
Gy	0.09
Gz	0.09

Table C.2: P-values associées à l'anova fonctionnelle

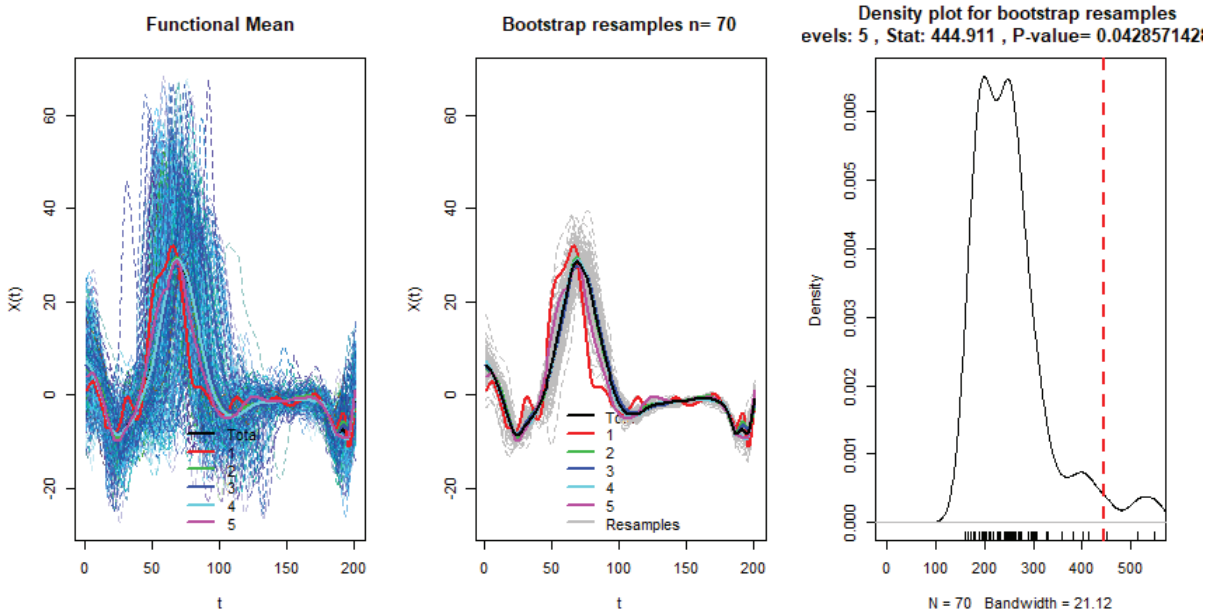


Figure C.7: Anova variable Ax et qualité du saut sur les 5 notes

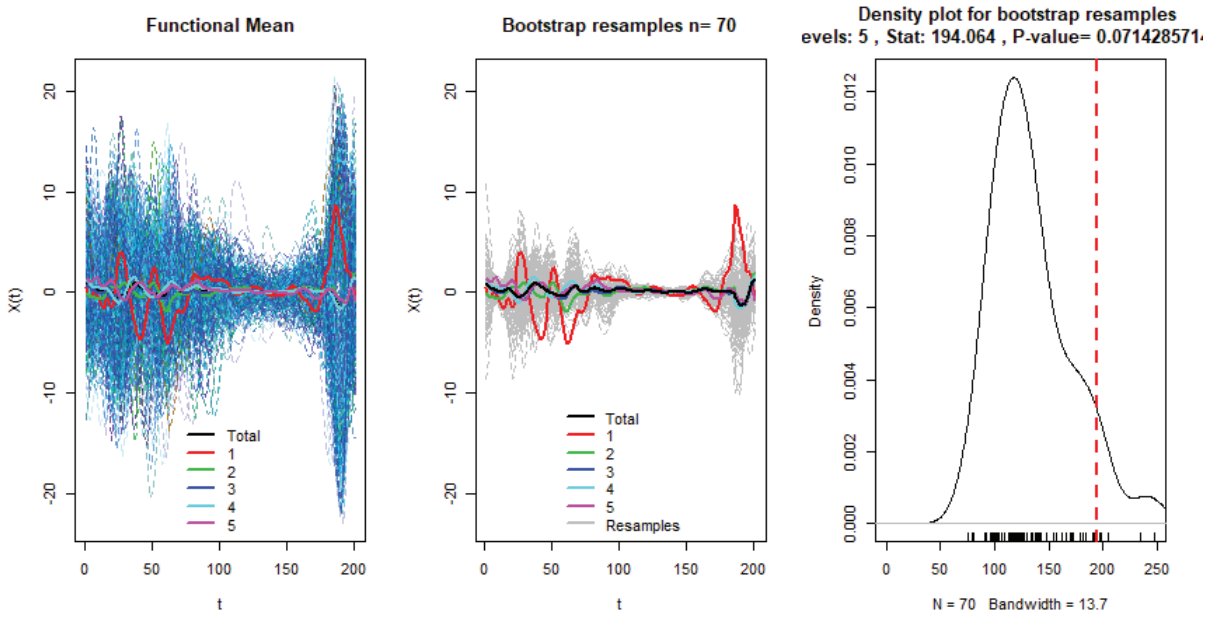


Figure C.8: Anova variable Ay et qualité du saut sur les 5 notes

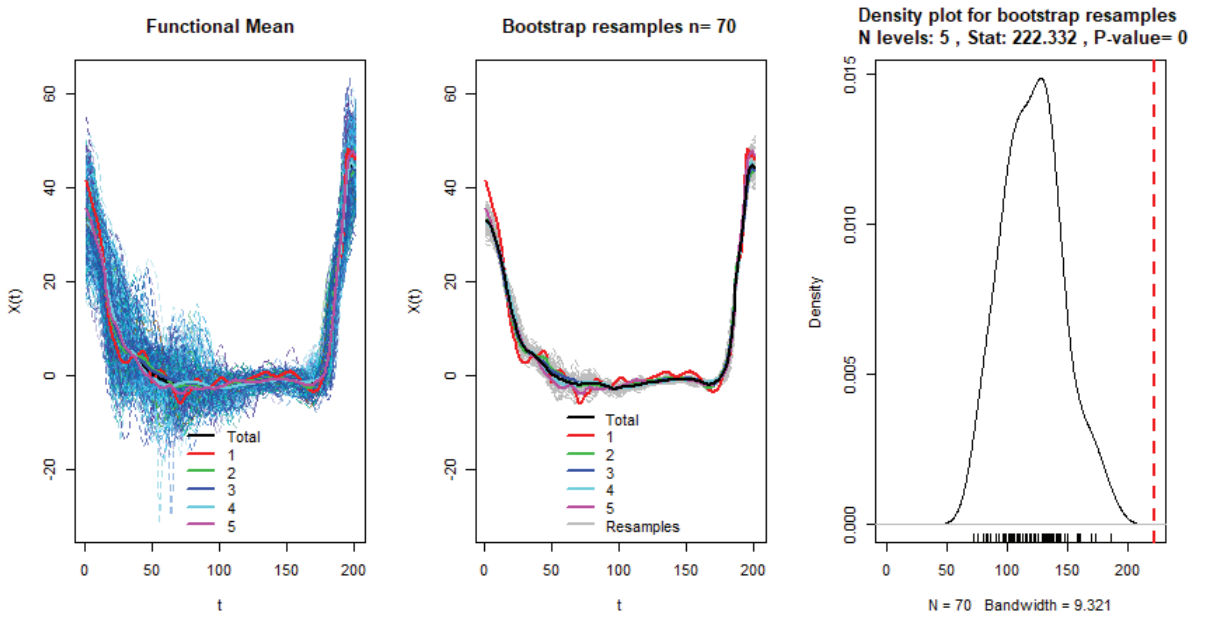


Figure C.9: Anova variable Az et qualité du saut sur les 5 notes

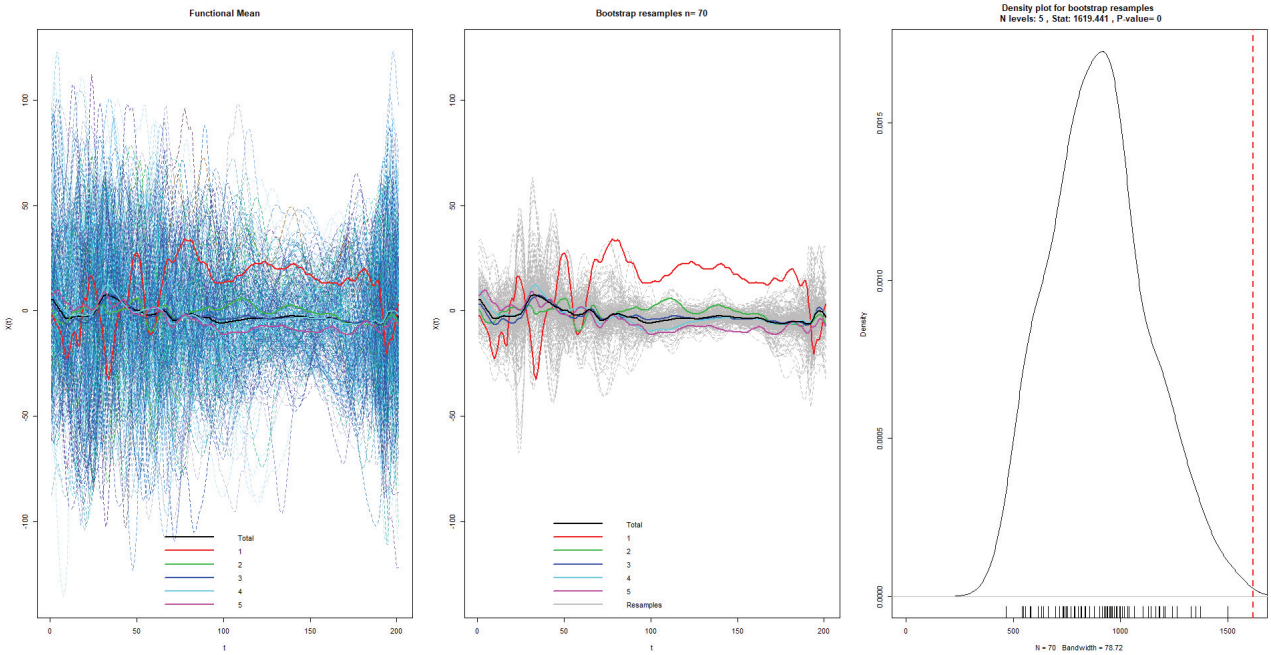


Figure C.10: Anova variable Gx et qualité du saut sur les 5 notes

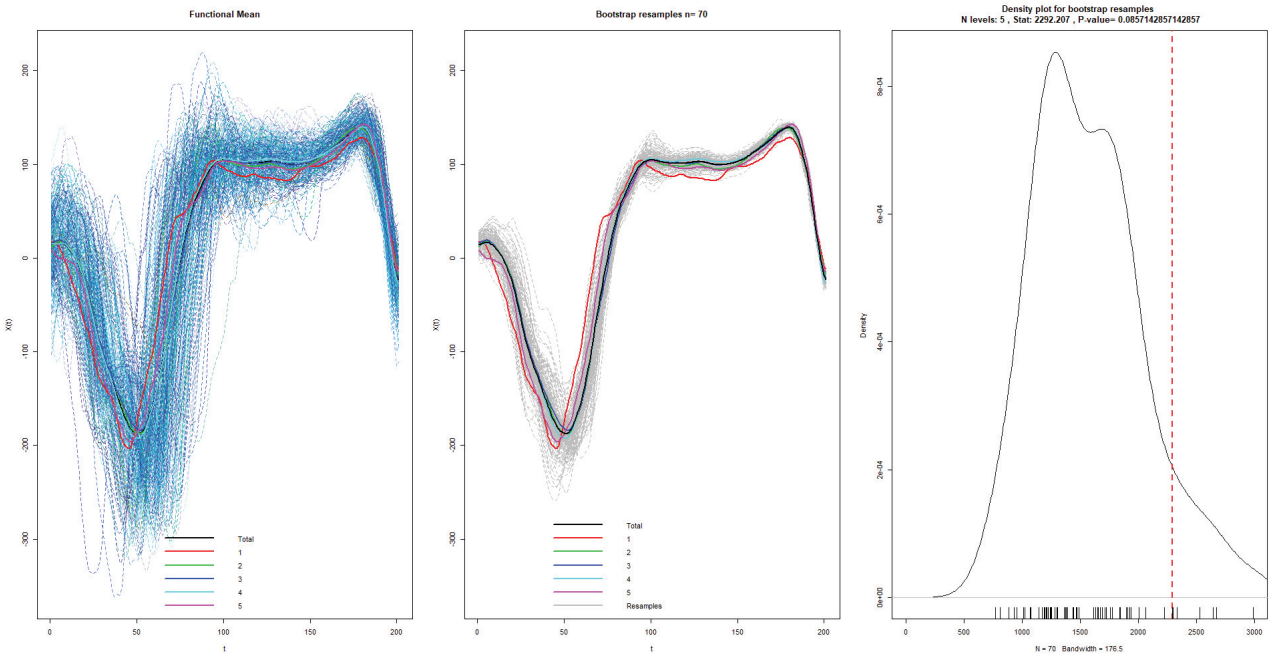


Figure C.11: Anova variable Gy et qualité du saut sur les 5 notes

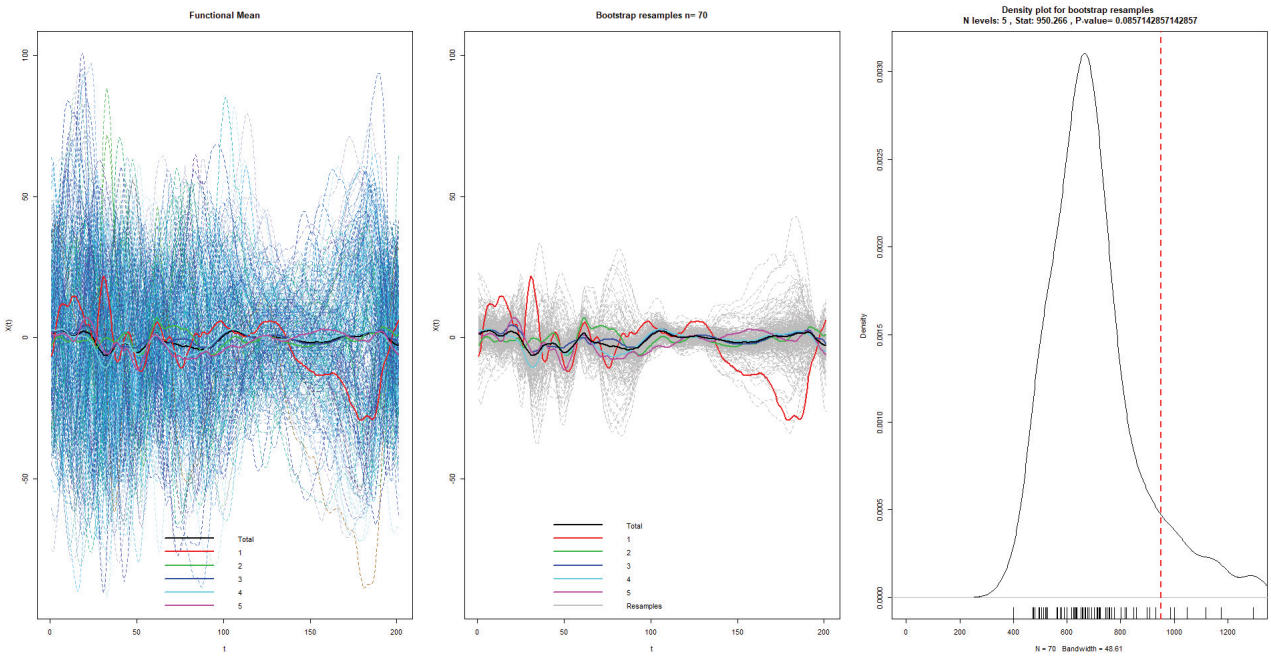


Figure C.12: Anova variable Gz et qualité du saut sur les 5 notes

3 Analyse de la variance des données de réception

L'analyse de la variance réalisée sur chacun des signaux et l'ensemble des 5 notes ne met en évidence aucune différence entre les courbes moyennes de chaque signal. (cf. Table C.3 et Figures C.13,C.14,C.15,C.16, C.17,C.18 en Annexe).

Variable	P-value analyse globale
Ax	0.1
Ay	0.1
Az	0.06
Gx	0.09
Gy	0.09
Gz	0.26

Table C.3: P-values associées à l'anova fonctionnelle

Pour conclure, la séparation des données en trois phases de saut ne permet pas de mettre en avant plus de différences que lorsque l'on étudie la base de données au globlal.

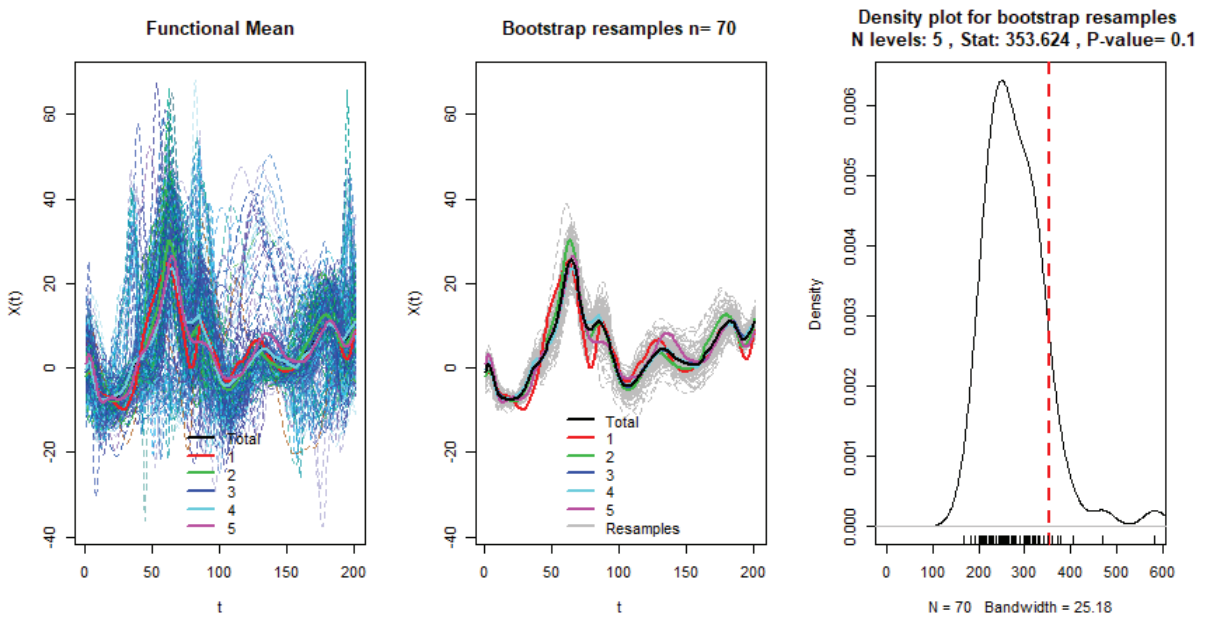


Figure C.13: Anova variable Ax et qualité du saut sur les 5 notes

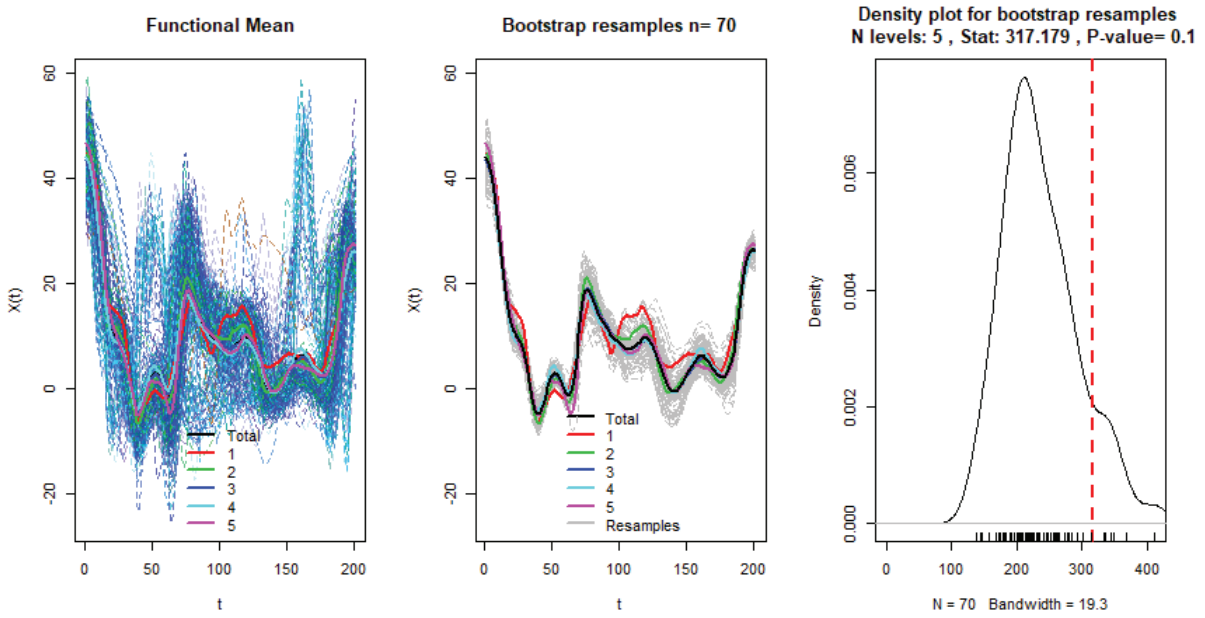


Figure C.14: Anova variable Ay et qualité du saut sur les 5 notes

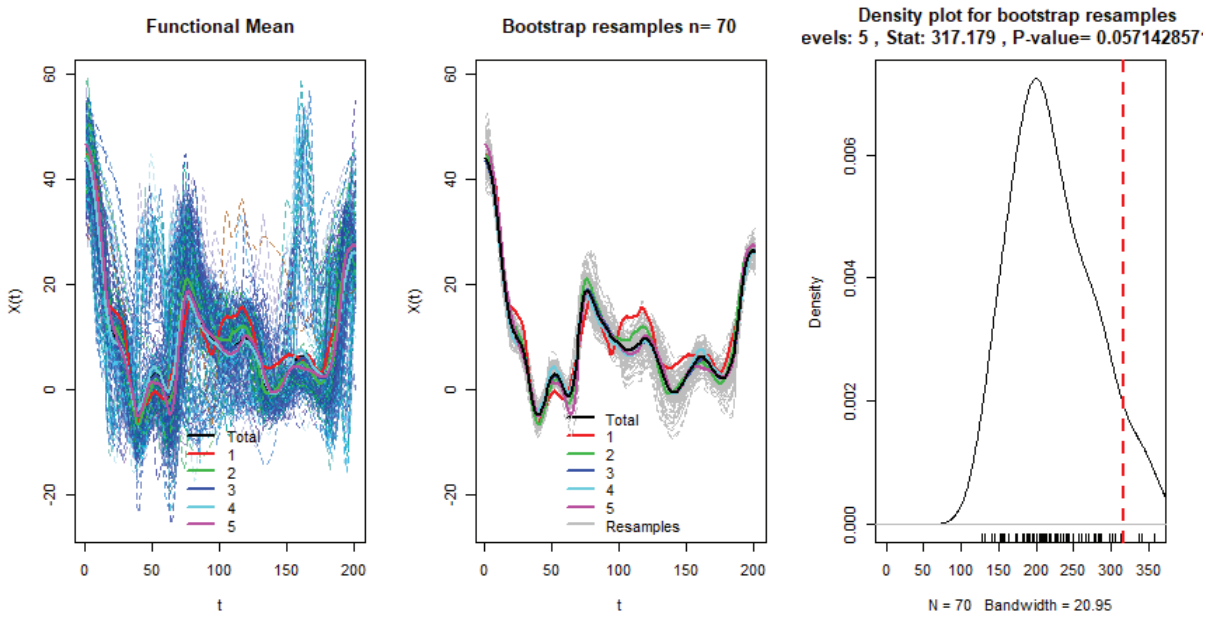


Figure C.15: Anova variable Az et qualité du saut sur les 5 notes

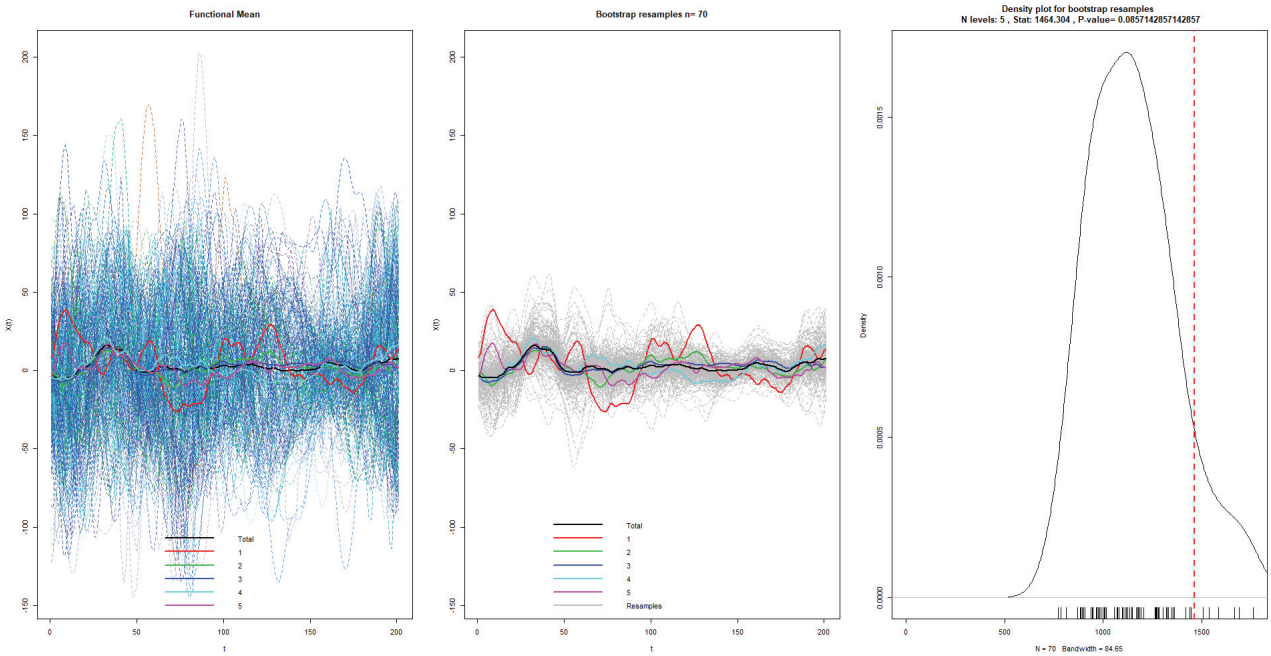


Figure C.16: Anova variable Gx et qualité du saut sur les 5 notes

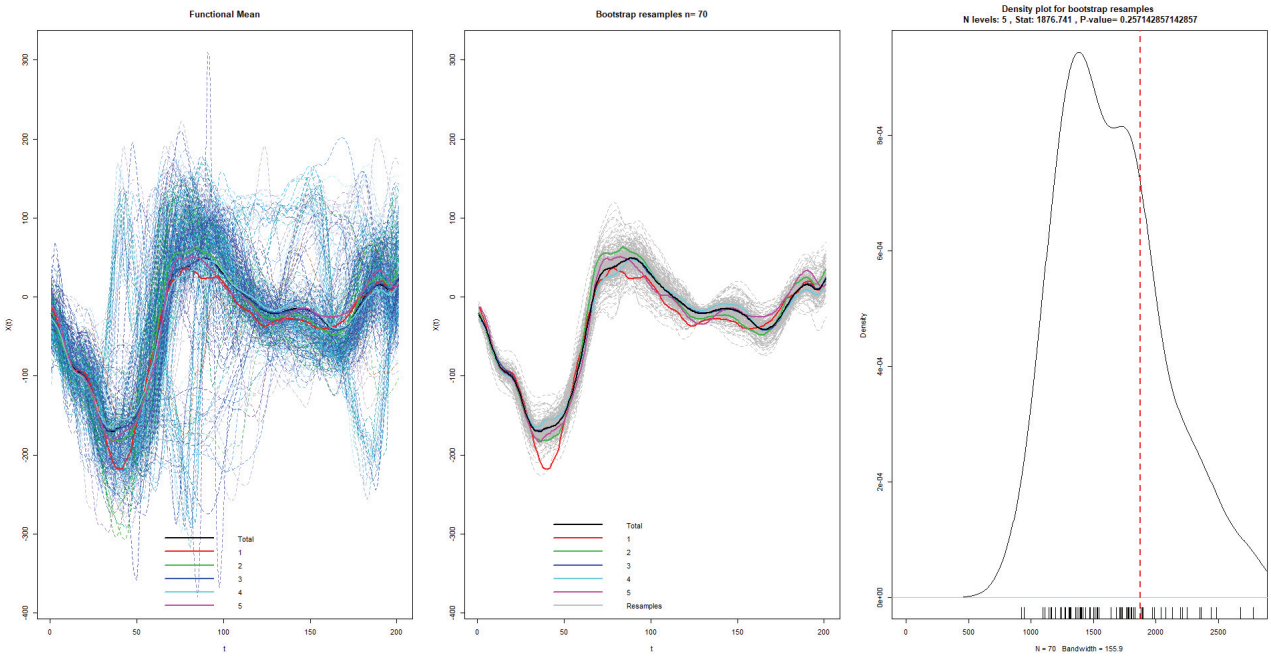


Figure C.17: Anova variable Gy et qualité du saut sur les 5 notes

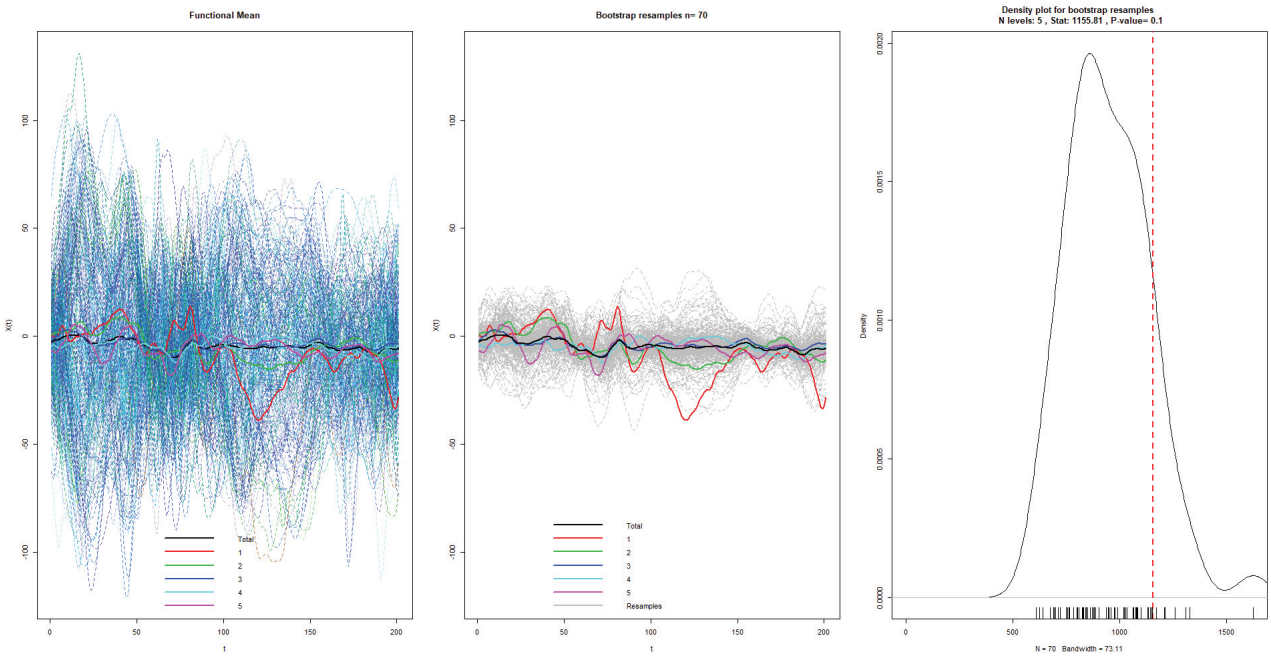


Figure C.18: Anova variable Gz et qualité du saut sur les 5 notes

4 ACP sur les données d'abord

La base de données est composée de 699 notes dont la répartition est résumée à la Table C.4. Cette répartition est très déséquilibrée avec 82% des des notes situées entre 3 et 4. Daprès le graphe des individus (Figure C.19), on ne distingue pas de groupes nets de sauts par rapports aux notes, toutes les couleurs sont mélangées. En revanche, ce graphe des individus met en évidence un premier groupe, composé de la majorité des sauts, proche du centre du graphe, donc du profil moyen et deux autres groupes de sauts qui se distinguent, l'un sur l'axe vertical, et l'autre sur l'axe horizontal. Lorsque l'on couple ces résultats avec le graphes des 10 variables ayant le plus contribué à la construction du premier plan factoriel (Figure C.20), on observe que le groupe de sauts sur l'axe horizontal se distingue des autres groupes de sauts par un pic présent au milieu du signal d'accélération selon X (au niveau des points 128 à 138 sur 202).

Table C.4: Répartition des notes

Note	1	2	3	4	5
Occurence	10	76	279	296	38

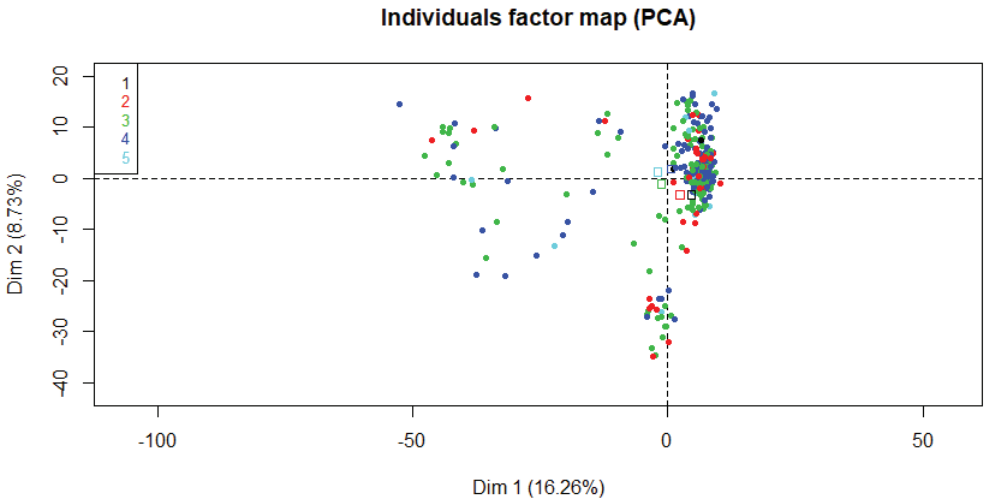


Figure C.19: Graphique des individus de l'ACP coloré par note

Le graphe des modalités de la variable note (cf. Figure C.21) met en évidence que la taille des ellipses de confiance est inversement proportionnelle aux nombre de sauts. Les notes extrêmes, peu nombreuses, rendent difficiles la distinction des modalités. L'apport de plus de notes extrêmes dans la base devrait permettre de distinguer les notes 1 et 2 des notes 3 et 4. Tandis que les notes 4 et 5 ont un barycentre très proches et ne seront peut être pas distingables à l'aide des signaux collectés par l'IMU.

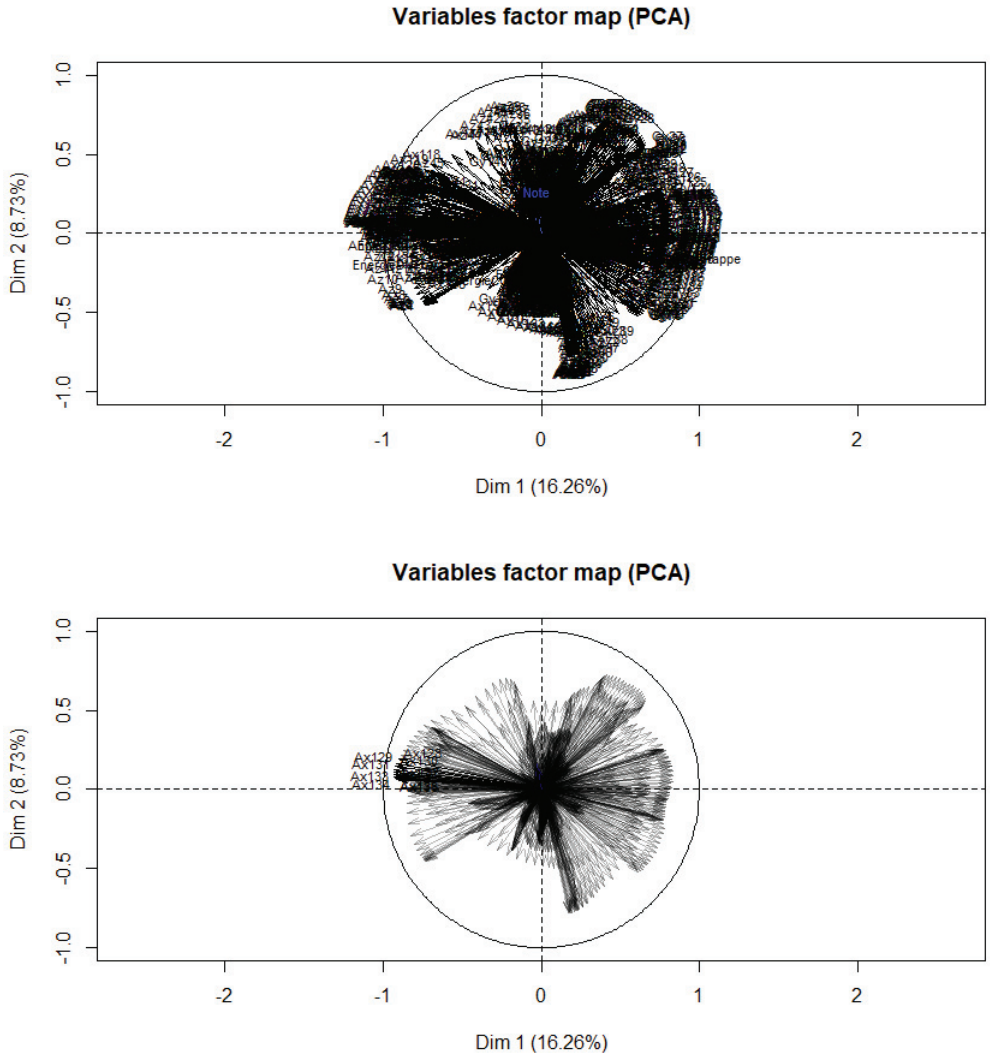


Figure C.20: Graphique des variables actives de l'ACP coloré avec la variable Note en supplémentaire (en haut) et graphique des 10 premières variables qui ont le plus contribué la la création du premier plan factoriel (en bas)

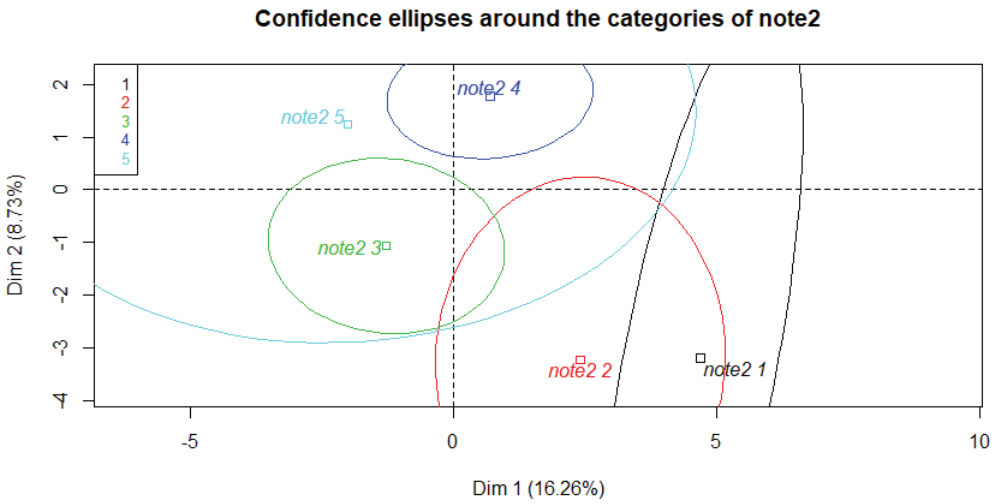


Figure C.21: Représentation des modalités de la variable Note (notée note2) avec les ellipses de confiance associées

5 ACP sur la base de données planer

La base de données est composée de 700 notes dont la répartition est résumée à la Table C.5.

Table C.5: Répartition des notes

Note	1	2	3	4	5
Occurence	10	94	270	290	36

Cette répartition est à nouveau très déséquilibrée avec 80% des notes situées entre 3 et 4. On ne distingue pas de groupes nets de sauts par rapports aux notes sur le graphe des individus (Figure C.22), toutes les couleurs sont en effet mélangées et le nuage de points est réparti de façon uniforme autour du centre de 0.

Lorsque l'on couple ces résultats avec le graphes des 10 variables ayant le plus contribué à la construction du premier plan factoriel (Figure C.23), on observe que ce premier plan factoriel oppose, sur l'axe des abscisses, les sauts qui prennent de fortes valeurs pour la vitesse angulaire Gy dans la première moitié du signal (points 69 à 73) aux sauts qui prennent de fortes valeurs au début du signal de vitesse angulaire Gy (points 28 à 32).

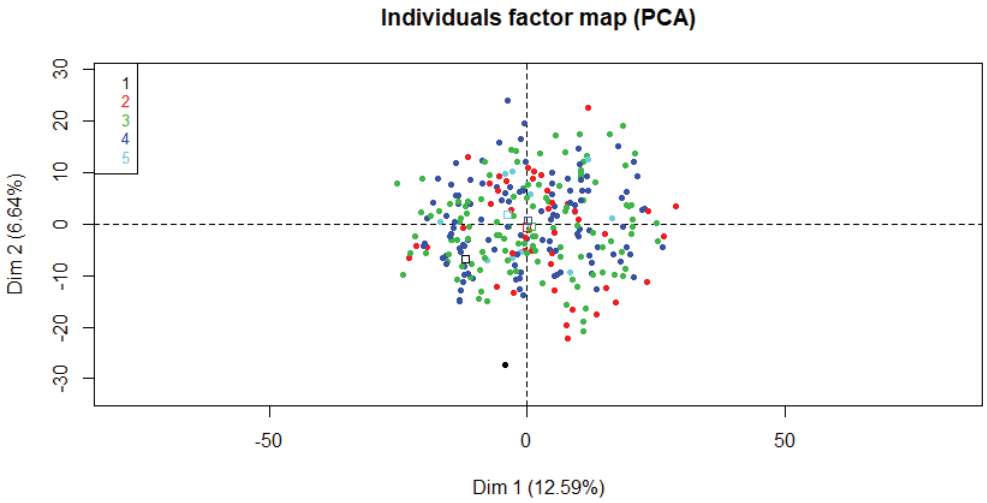
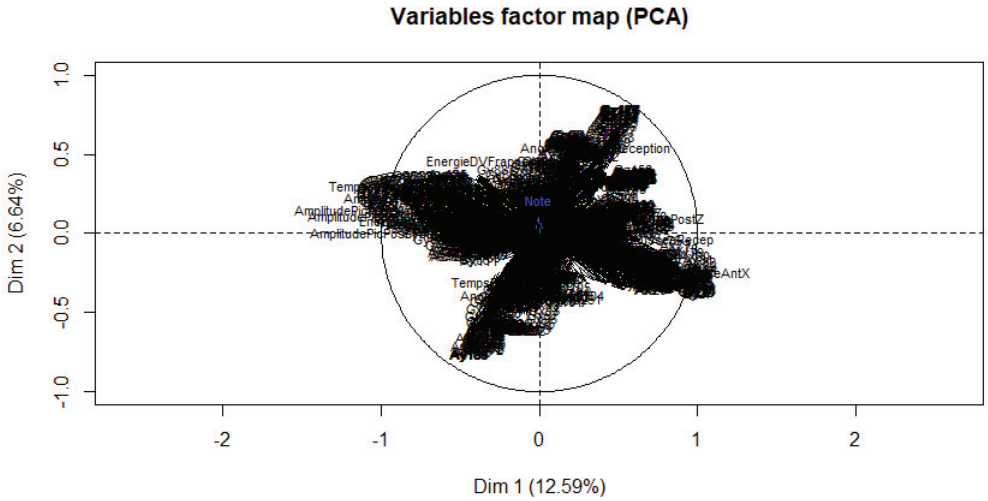


Figure C.22: Graphique des individus de l'ACP coloré par note

Le graphe des modalités de la variable note met en évidence des ellipses de confiance plus petites que dans le cas des données d'abord, mais on observe que les notes 2, 3 et 4 sont très proches et situées au niveau du profil moyen. Même en apportant plus de données il est possible que l'on ne puisse pas les distinguer. Comme dans le cas de la base générale on n'arrive pas à distinguer plus finement les sauts que différencier les bons des mauvais sauts.



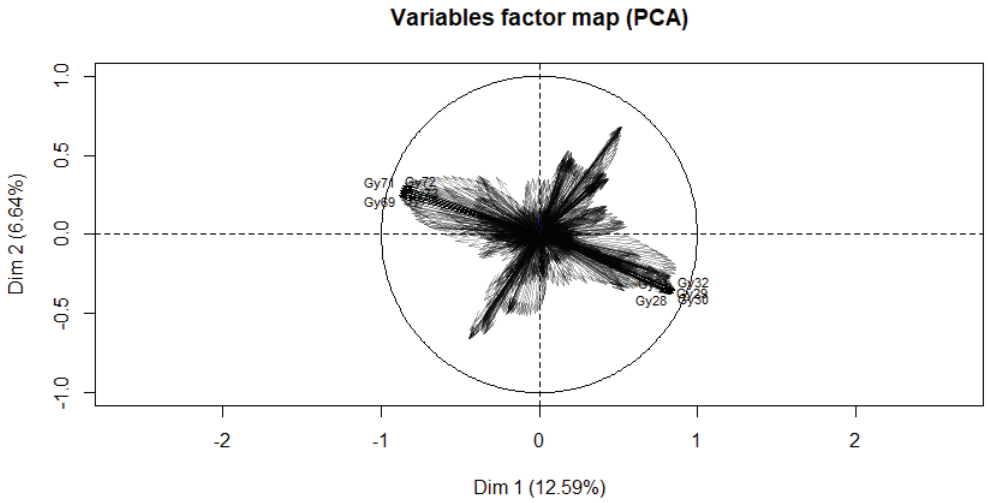


Figure C.23: Graphique des variables actives de l'ACP coloré avec la variable Note en supplémentaire (en haut) et graphique des 10 premières variables qui ont le plus contribué la la création du premier plan factoriel (en bas)

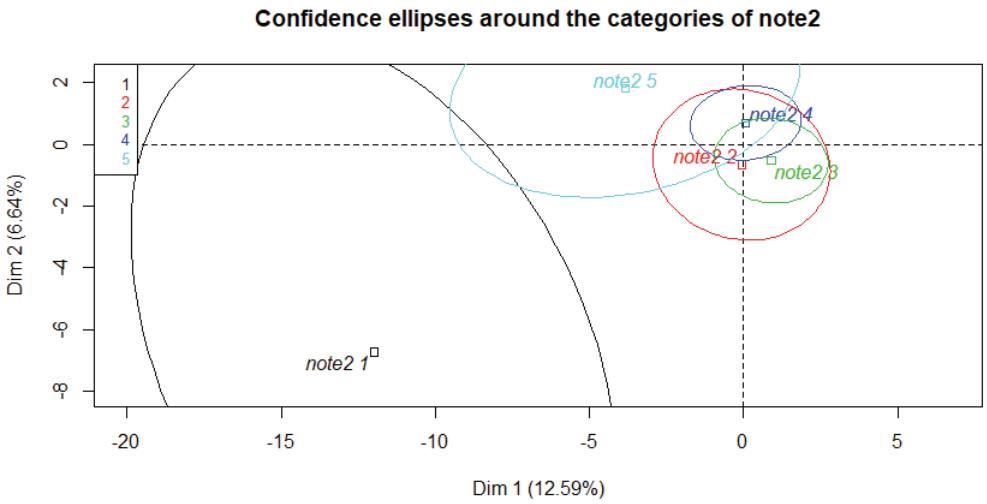


Figure C.24: Représentation des modalités de la variable Note (notée note2) avec leurs ellipses de confiance

6 ACP sur la base de données réception

La base de données est composée de 697 notes dont la répartition est résumée à la Table C.6. Comme précédemment la base est déséquilibrée avec 79% de notes moyennes (3 et 4), 14.5% de notes mauvaises (2) et très peu de notes extrêmes (1 et 5).

Table C.6: Répartition des notes

Note	1	2	3	4	5
Occurence	12	101	313	238	33

Le graphe des individus (cf. Figure C.25) met en évidence deux groupes principaux de part et d'autre de l'axe des ordonnées. D'après le graphe des variables (cf. Figure C.26), le groupe de gauche, comprenant la majorité des sauts, prend des fortes valeurs pour le début du signal de vitesse angulaire Gy (points 74 à 86) et de faibles valeurs pour les points 154 et 155. Inversement pour le groupe de droite.

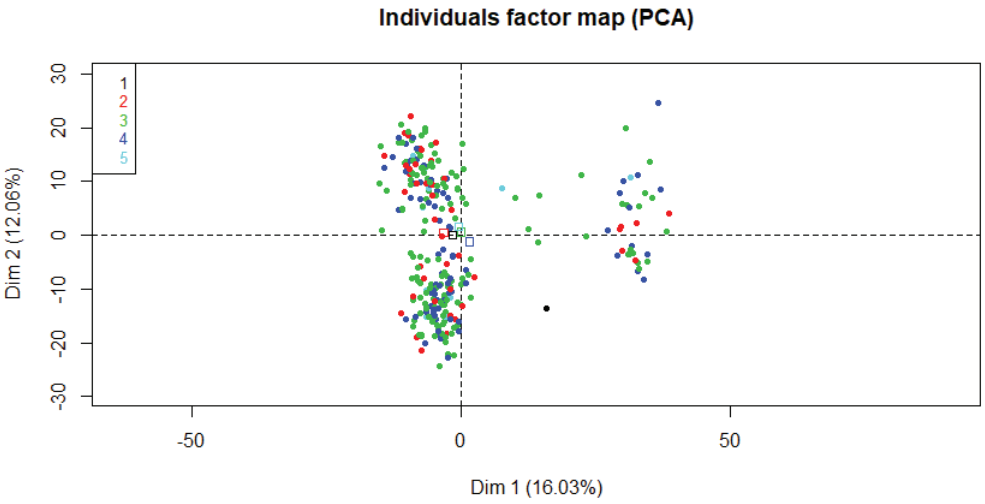
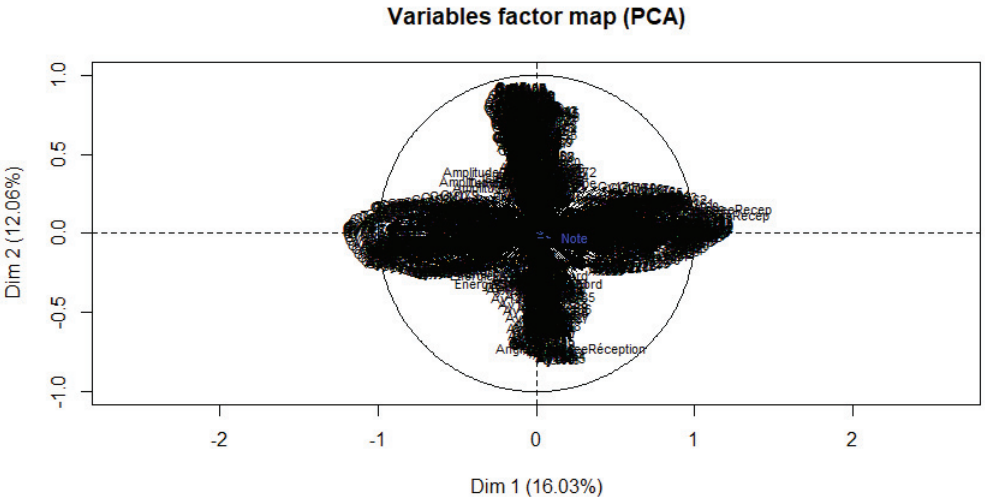


Figure C.25: Graphique des individus de l'ACP coloré par note

Pour conclure, l'étude des trois phases de saut prises séparément ne nous a pas permis d'être plus précis dans notre recherche de critères permettant de discriminer les notes attribuées aux sauts. Le problème majeur observé est un déséquilibre dans la base de données à notre disposition. La seconde limite est le faible pourcentage de variance expliqué par nos variables mesurées et calculées. Il faudrait donc approfondir les recherches sur la locomotion du cheval de saut pour voir quels autres paramètres biomécaniques peuvent impacter la qualité et pourraient être intégrés à notre analyse.



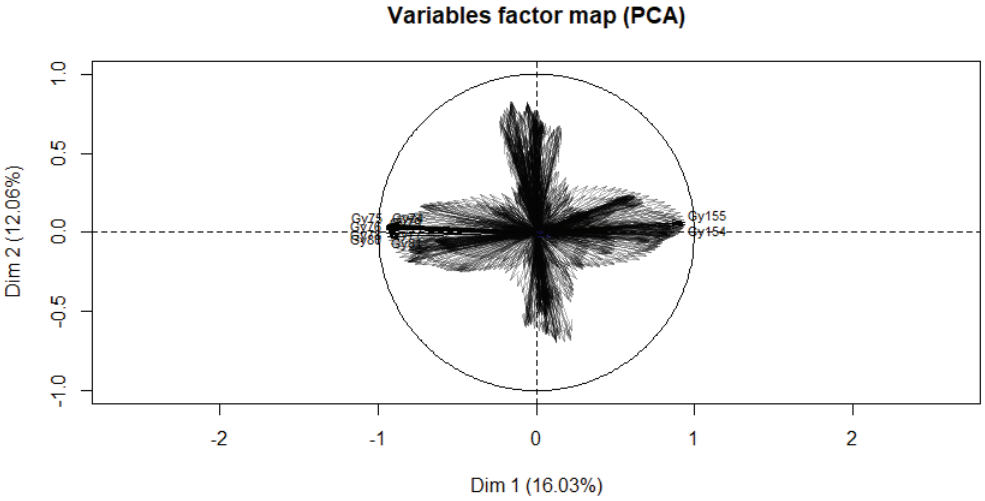


Figure C.26: Graphique des variables actives de l'ACP coloré avec la variable Note en supplémentaire (en haut) et graphique des 10 premières variables qui ont le plus contribué la la création du premier plan factoriel (en bas)

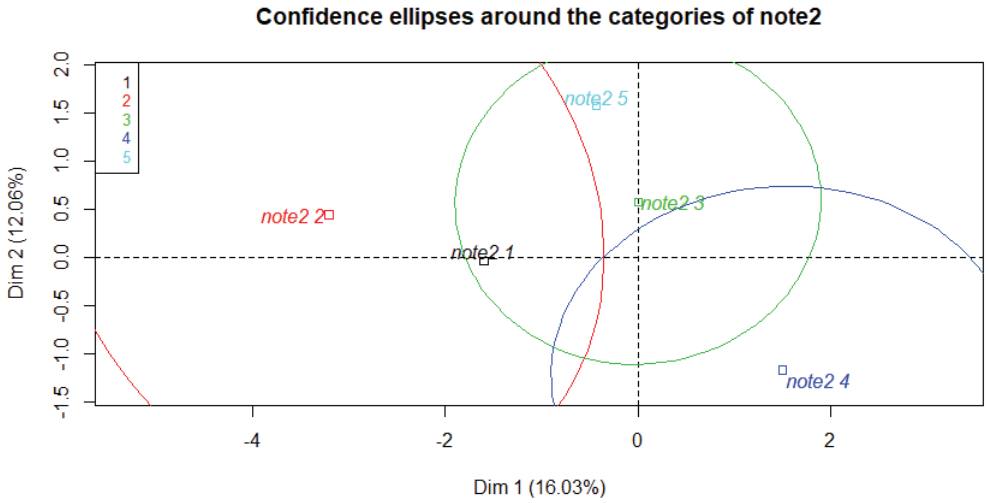


Figure C.27: Représentation des modalités de la variable Note sur le graphe des individus

Abstract

With the growth of smart devices market to provide athletes and trainers a systematic, objective and reliable follow-up, more and more parameters are monitored for a same individual. An alternative to laboratory evaluation methods is the use of inertial sensors which allow following the performance without hindering it, without space limits and without tedious initialization procedures. Data collected by those sensors can be classified as multivariate functional data: some quantitative entities evolving along time and collected simultaneously for a same individual.

The aim of this thesis is to find parameters for analysing the athlete horse locomotion thanks to a sensor put in the saddle. This connected device (inertial sensor, IMU) for equestrian sports allows the collection of acceleration and angular velocity along time in the three space directions and with a sampling frequency of 100 Hz. The database used for model development is made of 3221 canter strides from 58 ridden jumping horses of different age and level of competition. Two different protocols are used to collect data: one for straight path and one for curved path.

We restricted our work to the prediction of three parameters: the speed per stride, the stride length and the jump quality. To meet the first to objectives, we developed a multivariate functional clustering method that allow the division of the database into smaller more homogeneous sub-groups from the collected signals point of view. This method allows the characterization of each group by it average profile, which ease the data understanding and interpretation. But surprisingly, this clustering model did not improve the results of speed prediction, Support Vector Machine (SVM) is the model with the lowest percentage of error above 0.6 m/s. The same applied for the stride length where an accuracy of 20 cm is reached thanks to SVM model. Those results can be explained by the fact that our database is build from 58 horses only, which is a quite low number of individuals for a clustering method.

Then we extend this method to the co-clustering of multivariate functional data in order to ease the datamining of horses' follow-up databases. This method might allow the detection and prevention of locomotor disturbances, main source of interruption of jumping horses.

Lastly, we looked for correlation between jumping quality and signals collected by the IMU. First results show that signals collected by the saddle alone are not sufficient to differentiate finely the jumping quality. Additional information will be needed, for example using complementary sensors or by expanding the database to have a more diverse range of horses and jump profiles.