Statistique pour données fonctionnelles [1]

1. Enseignant : G. Chagny, gaelle.chagny@univ-rouen.fr.

Table des matières

Ce cours propose une introduction aux données fonctionnelles, et aux méthodes statistiques permettant de les appréhender, sans bien sûr chercher à être exhaustif. L'essentiel des résultats, exemples, et méthodes pourront être retrouvées, de manière beaucoup plus détaillée dans les monographies de Ramsay et Silverman (2005, 2002); Ferraty et Vieu (2006); Green et Silverman (1994). Les graphiques de ce document, et les codes associés sont issus du logiciel (libre) R. Les fonctions spécifiques à l'analyse de données fonctionnelles, ainsi que les principaux jeux de données que nous utilisons peuvent être trouvés en particulier dans le package `fda`, disponible sur le site du CRAN [2]. On pourra utiliser également le livre associé de Ramsay *et al.* (2009). Des références bibliographiques supplémentaires sont indiquées à la Section 1.3. Ce document doit aussi beaucoup aux discussions avec A. Roche, ainsi qu'à ses travaux : il s'inspire en particulier de l'introduction de sa thèse (Roche, 2014).

2. `https://cran.r-project.org/web/packages/fda/index.html`

Chapitre 1

Introduction aux données fonctionnelles

1.1 Des données continues

1.1.1 Introduction et exemples

On s'intéresse dans ce cours à l'étude statistique des observations qui ne sont pas, comme généralement en statistique, des réalisations de variables aléatoires réelles ou vectorielles (vecteurs aléatoires), mais des fonctions aléatoires : courbes, images, etc... Il s'agit de données de dimension infinie, c'est-à-dire rentrant dans le champ de la "très grande dimension". Elles apparaissent de plus en plus fréquemment dans de nombreux domaines scientifiques, grâce aux progrès récents en matière de stockage et traitement des données. Sans chercher l'exhaustivité, la biologie, la climatologie, l'économétrie ou encore la chimie sont susceptibles de produire des données considérées comme des courbes aléatoires.

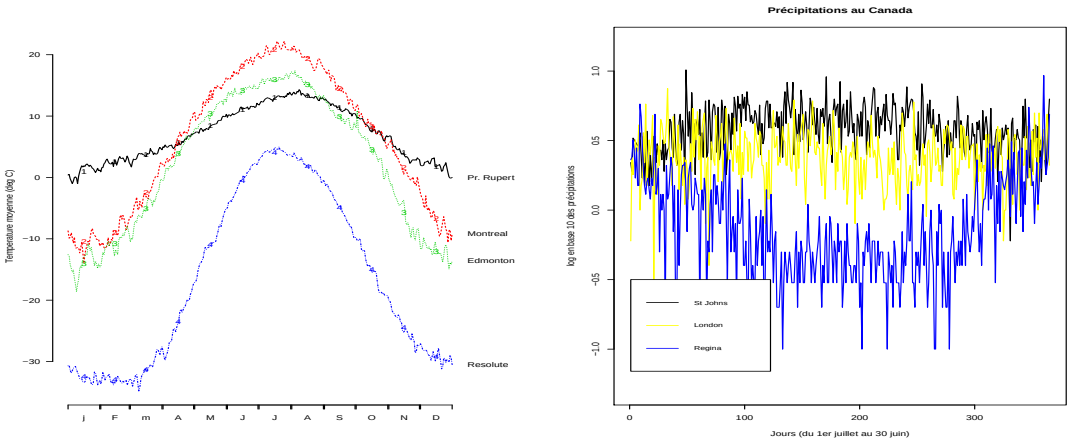
Quelques exemples.

Études longitudinales. Il est courant de disposer de données concernant un même phénomène mesuré quantitativement à différents temps de mesure. On peut alors souvent considérer que l'on dispose de courbes aléatoires dépendant du temps (réalisation de processus à temps continu indexés par le temps).

— Courbe des températures relevées en un point donné, à différents instants, courbe des cumuls mensuels de précipitations en un point donné. De nombreux exemples sont disponibles dans les données `CanadianWeather` du package `fda`. On représente à titre d'exemple au graphique (a) de la Figure 1.1 l'évolution des températures au cours d'une année (mesures moyennes journalières) dans 4 stations météorologiques canadiennes, et au graphique (b) les précipitations journalières dans 3 autres stations. Un autre exemple est constitué par des données issues du CEA et simulant l'évolution de la température dans un réacteur nucléaire lors d'un accident de type perte de réfrigérant primaire (Figure 1.2).

— Courbe de croissance d'un individu (ou d'une plante) au cours du temps. Les données `growth` du package `fda` en sont un exemple : elles contiennent les tailles de filles et de garçons, mesurées à 31 âges, entre 1 et 18 ans. Les courbes associées pour 10 filles de l'échantillon sont représentées à la Figure 1.3.

— Données de biomécanique. Les données `pinch` du package `fda` représentent 20 enregistrements de l'évolution au cours d'une manipulation de la force exercée entre le pouce et l'index d'un sujet : celui-ci devait d'abord maintenir une force de



(a) (b)

FIGURE 1.1 – (a) Courbes des températures moyennes journalières dans 4 stations météorologiques canadiennes. (b) Courbes des précipitations moyennes journalières dans 3 stations météorologiques canadiennes.

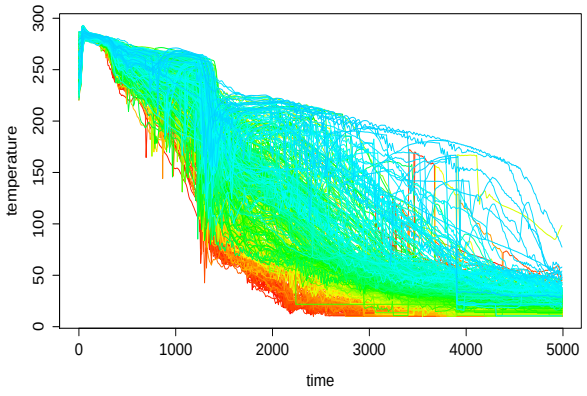


FIGURE 1.2 – Simulations de l'évolution de la température dans un réacteur nucléaire lors d'un accident de type perte de réfrigérant primaire (CEA).

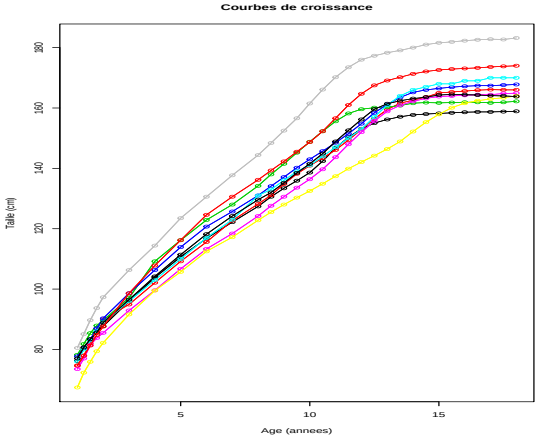


FIGURE 1.3 – Courbes de croissance de 10 filles, de 1 à 18 ans.

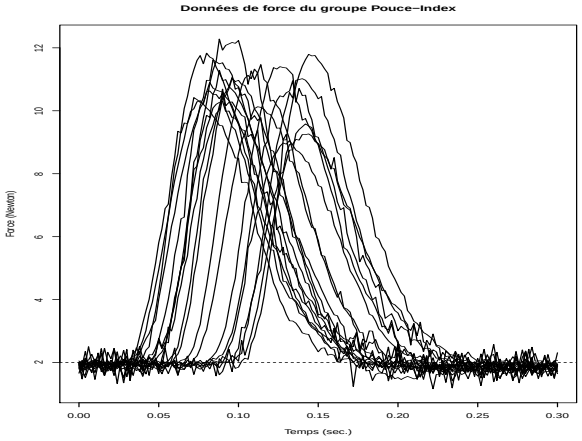


Figure 1.4 – 20 enregistrements de l'évolution au cours d'une manipulation de la force exercée entre le pouce et l'index d'un sujet .

repos de l'ordre de 2 Newtons, avant de développer une force maximale, puis de retourner au niveau de repos (le but était l'étude du groupe musculaire contrôlant le pouce et l'index). Les courbes obtenues sont représentées à la Figure 1.4.

— Électrocardiogramme d'un individu, courbe d'évolution d'un taux d'intérêt au cours du temps, ...

Autres exemples de courbes aléatoires dépendant d'une variable. L'analyse des données fonctionnelles ne se réduit toutefois pas à étudier des quantités évoluant au cours du temps. L'*analyse spectrométrique* en chimie en constitue un exemple typique. L'objectif est de déduire des propriétés physico-chimiques de matériaux en étudiant un spectre de lumière issu du matériau en question, c'est-à-dire la courbe d'absorbance de la lumière en fonction de la longueur d'onde. Un exemple classique concerne l'industrie agro-alimentaire, et plus précisément un problème de contrôle de qualité d'échantillons de viande hachée. La Figure 1.5 fait apparaître 100 courbes représentant chacune le spectre d'absorption de la lumière dans un morceau de viande, enregistré par spectrométrie dans le proche infrarouge. Les données complètes sont disponibles en ligne [1], précisément décrites au Chapitre 2 de Ferraty et Vieu (2006), et ont été largement étudiées dans la littérature (voir aussi Ferraty et Vieu 2002; Ferraty *et al.* 2007).

Données fonctionnelles multivariées ou spatiales. Certaines images peuvent être modélisées comme des fonctions aléatoires, dépendant cette fois de deux variables (on peut alors parler de données fonctionnelles multivariées). C'est le cas des données d'écriture manuscrite `handwrit` du package `fda`. Une personne a écrit 20 fois les trois lettres 'fda' et le jeu de données contient les abscisses et ordonnées de la marque du stylo à différents instants d'écriture, pour chacun des tracés. Les données complètes sont tracées à la Figure 1.6 ainsi que l'évolution d'une des deux variables (l'abscisse) au cours du temps, pour chacun des mots écrit.

Dans tous les exemples ci-dessus, les données consistent souvent en des réplications d'un même phénomène (constituant un échantillon). Mais on pourra parfois travailler sur un seul enregistrement. C'est le cas des données `refinery` du package `fda`, issues d'une raffinerie de pétrole aux États-Unis. On observe une seule réalisation de deux variables, voir Figure 1.7 :

1. `https://www.math.univ-toulouse.fr/ ferraty/SOFTWARES/NPFDA/npfda-datasets.html`

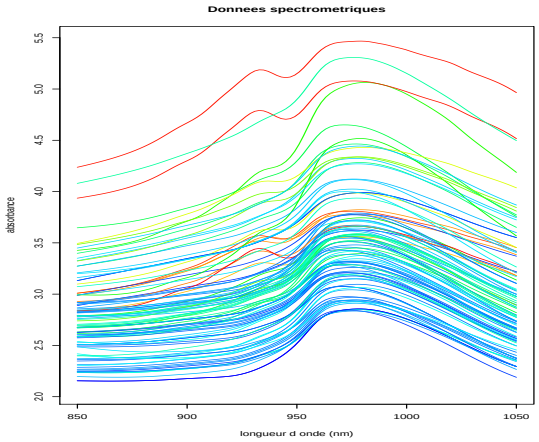
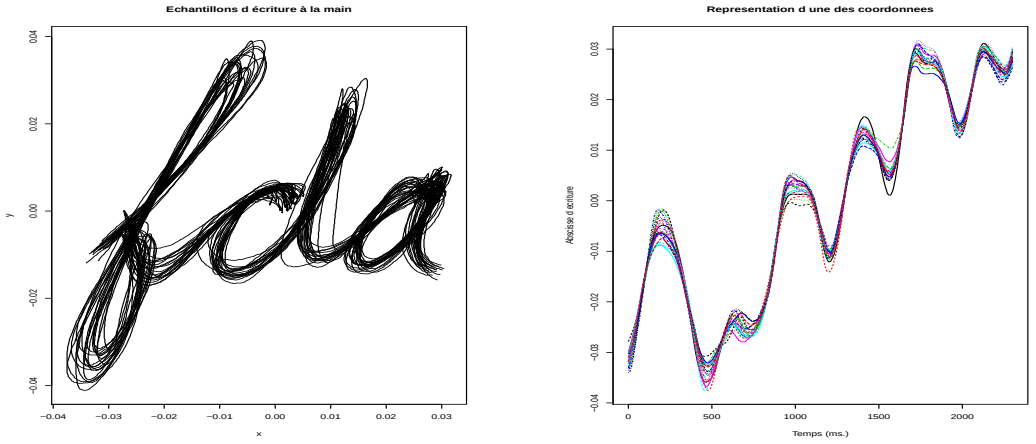
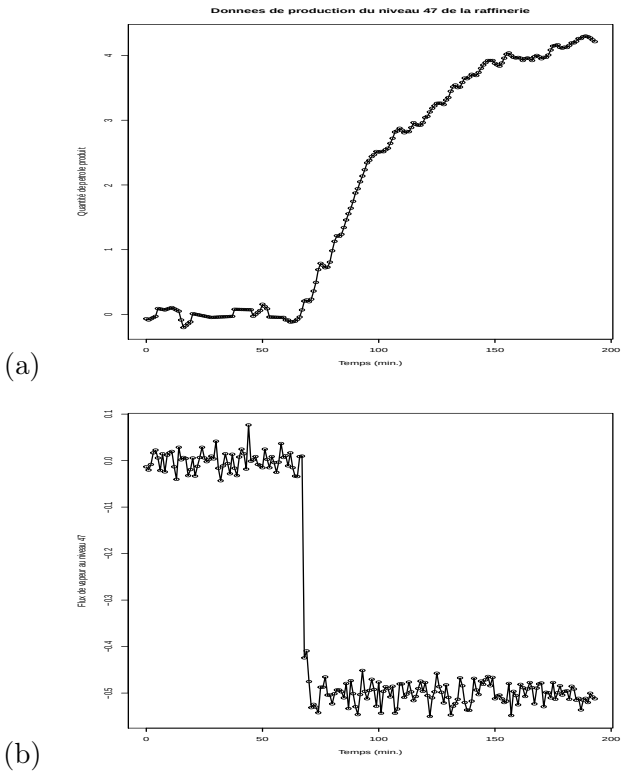


FIGURE 1.5 – Trajectoires observées d'un spectre d'absorbance de 100 canaux mesurés par spectrophotomètre (échantillons de viande, appareil de mesure : Tecator Infratec Food and Feed Analyzer).



(a) (b)

FIGURE 1.6 – Échantillon d'écriture à la main. (a) 20 tracés (l'unité de l'axe des abscisses est le cm). (b) courbes d'évolution des abscisses au cours du temps (20 courbes).



(a)

(b)

FIGURE 1.7 – Données de production de pétrole. (a) quantité de pétrole produit à un certain niveau de la colonne de distillation, en fonction du temps. (b) flux de vapeur dans cette même colonne, en fonction du temps.

la quantité de pétrole produit à un certain niveau de la colonne de distillation (graphique (a)) et le flux de vapeur dans la colonne (graphique (b)). C'est aussi le cas de la mesure de l'incidence du mélanome dans un échantillon de population de 100 000 individus entre 1936 et 1972, aux États-Unis toujours (données `melanoma` du package `fda`, Figure 1.8).

1.1.2 Modélisation mathématique

Le modèle communément utilisé en analyse de données fonctionnelles, et convenant en particulier pour tous les exemples précédemment cités est le suivant : on considère que les courbes sont des réalisations de processus stochastiques comportant des trajectoires relativement régulières (lisses). Dans la suite du cours, on note $(\Omega, \mathcal{A}, \mathbb{P})$ un espace probabilisé (Ω est un ensemble, $\mathcal{A}$ une tribu sur cet ensemble, et $\mathbb{P}$ une mesure de probabilité sur $\mathcal{A}$). On note également $\mathcal{F}$ un espace de fonctions (dont on supposera que c'est au moins un espace de Banach séparable, c'est-à-dire un espace vectoriel normé complet, contenant un sous ensemble dense dénombrable, voir Section 2.1.1).

Définition 1.1 *Une variable aléatoire est dite* ***variable aléatoire fonctionnelle*** *si elle prend ses valeurs dans un espace vectoriel de dimension infinie. Typiquement, il s'agit donc d'une application mesurable $X : \Omega \to \mathcal{F}$. Une* ***donnée fonctionnelle*** *est alors une réalisation de la variable X.*

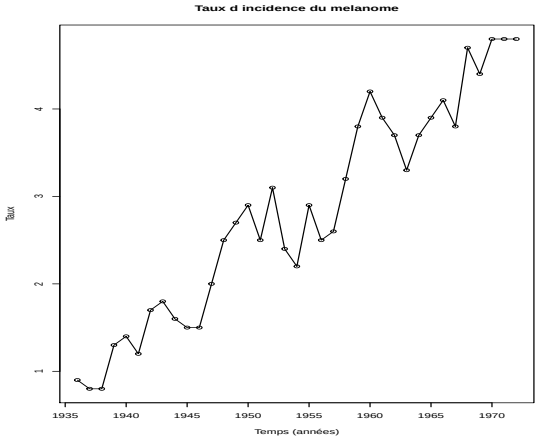


FIGURE 1.8 – Évolution du taux d'incidence du mélanome dans un échantillon de la population des États-Unis.

Si l'espace $\mathcal{F}$ est constitué de fonctions définies sur un ensemble T (par exemple l'intervalle $[0, 1]$) et à valeurs réelles, on peut voir une variable fonctionnelle X comme une application $X : \Omega \times T \to \mathbb{R}$, telle que

— pour tout $\omega \in \Omega$ fixé, la fonction $X(\omega, \cdot) : T \to \mathbb{R}$ est une *trajectoire* de X,

— pour tout $t \in T$ fixé, la variable $X(\cdot, t) : \Omega \to \mathbb{R}$ est une variable aléatoire réelle.

Si l'ensemble T est inclus dans $\mathbb{R}$, la variable X est une courbe aléatoire ; lorsque T est de dimension supérieure, on a un processus aléatoire plus complexe, par exemple une surface aléatoire lorsque T est inclus dans $\mathbb{R}^2$ (une image typiquement). Comme habituellement en probabilités et en statistique, on omettra les arguments ω et t dans la suite, s'il n'y a pas de confusion possible, et on confondra les variables et leurs réalisations.

Dans la suite de ce cours (Chapitre 4), on considérera souvent un échantillon $X_1, \ldots, X_n$ ($n \in \mathbb{N}\backslash\{0\}$ est la taille de l'échantillon) de données fonctionnelles indépendantes et identiquement distribuées selon la loi de X. Notons qu'il est également possible dans ce cadre fonctionnel de considérer des données dépendantes (séries temporelles fonctionnelles par exemple), mais nous ne traiterons pas ce cas.

1.2 Choix des données fonctionnelles et nécessité d'outils statistiques spécifiques

On n'observe en réalité jamais une courbe aléatoire ou un processus X dans son ensemble : cela supposerait en effet d'une part que l'on dispose d'instruments de mesure avec une vitesse d'enregistrement infinie, et d'autre part que l'on est capable de stocker un nombre infini (non dénombrable) de valeurs. Une donnée fonctionnelle se présente donc en fait tout d'abord sous forme vectorielle : elle est constituée d'un certain nombre de valeurs discrètes qui ont été mesurées sur une grille suffisamment fine, et enregistrées. Considérons des fonctions d'un ensemble T dans $\mathbb{R}$. Ainsi, on n'observe pas intégralement l'ensemble $\{X(t),\, t \in T\}$, mais un ensemble

$$\{X(t_1), \ldots, X(t_p)\},$$

où $\{t_1, \ldots, t_p\} \subset T$ est une grille de discrétisation (une suite ordonnée de valeurs). Dans le cas d'un échantillon de données, on a généralement accès à n vecteurs $(X_i(t_{i,1}), \ldots, X_i(t_{i,p}))$, où $\{t_{i,1}, \ldots, t_{i,p}\}$ représente toujours la grille, pour $i \in \{1, \ldots, n\}$: cette grille peut, ou non, être identique pour toutes les courbes X_i de l'échantillon. Ainsi, sur les données de croissance représentées à la Figure 1.3, les observations sont en fait représentées par des cercles qui décrivent la taille en centimètres des 10 individus en fonction de leur âge. Les mesures sont faites tous les trimestres jusqu'à 2 ans, puis de manière annuelle ensuite, de 2 à 8 ans, et enfin semestriellement jusqu'à 18 ans. On a donc une discrétisation, aux points d'observation, du phénomène continu qu'est la croissance. De même, les mesures de température du graphique (a) de la Figure 1.1 sont journalières. Prises tout au long d'une année, elles représentent les valeurs de la courbe aléatoire de la température aux instants d'observations. Le dernier exemple est celui des mesures d'indices sur les marchés financiers : aussi rapprochées et aussi rapides soient-elles, les valeurs des indices enregistrées sont toujours séparées de quelques millisecondes.

Ce type de données n'est pas nouveau, et a longtemps été étudié avec les techniques classiques de la statistique multivariée : les données sont alors vues comme des vecteurs aléatoires de $\mathbb{R}^p$. L'apport de la statistique pour données fonctionnelles est de traiter ces données comme des réalisations de fonctions aléatoires (en théorie, on pourrait obtenir des points aussi rapprochés que l'on veut, c'est-à-dire des mesures sur une grille extrêmement fine). Les avantages sont les suivants :

Prise en compte de grilles d'enregistrement possiblement différentes. Il arrive que les données récoltées ne le soient pas sur la même grille, en pratique. C'est par exemple le cas lorsqu'on a des données incomplètes, avec des observations manquantes. L'analyse de données fonctionnelles propose un cadre théorique rigoureux permettant de prendre en compte ce problème, et permet d'atténuer les effets de la non-correspondance des instants d'observation : si on dispose de données observées par exemple à des instants différents, une approximation continue permet d'évaluer la variable aux mêmes points pour tous les individus.

Prise en compte des problématiques liées à la grande dimension. Il arrive de plus en plus souvent, du fait de l'amélioration des appareils de mesures en matière de recueil et stockage des données, que la fréquence de discrétisation des courbes, c'est-à-dire la taille p de la grille, soit très grande, et surtout très grande devant la taille n de l'échantillon elle-même : on note généralement $p >> n$. Les méthodes classiques de statistique multivariée se sont alors montrées inadéquates dans ce contexte, où paradoxalement, l'afflux de données aboutit à une détérioration des résultats classiques. C'est la fameuse "malédiction de la dimension" (*curse of dimensionality*). Pour y faire face, des outils propres aux problèmes de la grande dimension se sont développés ces dernières années, parmi lesquelles les méthodes et modèles de la statistique pour données fonctionnelles.

Prise en compte de la structure intrinsèque des données, (et en particulier de la régularité du phénomène observé). Considérer qu'on peut traiter les données explicitées sous forme discrètes comme des fonctions permet de prendre en compte la nature continue intrinsèque du phénomène observé, celle de fonctions aléatoires, et permet de faire intervenir également leurs régularités. Une corrélation importante existe généralement entre deux observations rapprochées d'un même phénomène continu : si X mesure un phénomène continu, $X(t_j)$ et $X(t_{j+1})$ sont nécessairement liés l'un à l'autre et peu susceptibles d'être très différents. Ceci incite à considérer par exemple des opérations de dérivation, pour étudier la régularité du phénomène, opérations qui n'auraient pas de sens dans un cadre vectoriel classique. Grâce à l'utilisation de données fonctionnelles, on pourra par exemple, pour un phénomène donné, étudier non seulement

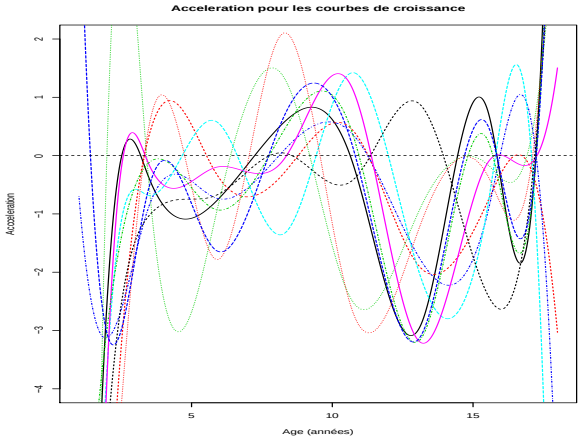


FIGURE 1.9 – Accélération estimée pour la croissance des 10 filles, mesurée en cm par an^2 (voir les courbes de croissance correspondantes à la Figure 1.3).

le phénomène lui-même mais tirer également parti de sa vitesse ou son accélération, en considérant les dérivées premières et secondes de X (voir Figure 1.9). L'étude des dérivées permet également d'analyser les aspects géométriques des variables. On peut aussi introduire d'autres informations *a priori* sur les courbes étudiées. Lors de la modélisation, on pourra par exemple prendre en compte des phénomènes de périodicité (c'est le cas des courbes de températures du graphique (a) de la Figure 1.1).

Tout ceci justifie l'intérêt porté ces dernières décennies au développement de méthodes statistiques spécifiques pour l'étude des données fonctionnelles afin de prendre en compte à la fois leur richesse et leur complexité (Ferraty et Vieu, 2011). En respectant la nature fonctionnelle de l'échantillon, on espère améliorer les résultats de l'inférence statistique par rapport à ceux qui seraient obtenus avec des méthodes vectorielles classiques. La question qui se pose naturellement est alors de savoir dans quels cas l'approche fonctionnelle est plus efficace. Deux situations sont généralement citées dans la littérature :

1. le premier cas est celui de données observées représentant un phénomène régulier (la croissance, l'évolution des températures, etc... voir par exemple Figure 1.3),
2. le second concerne le cas de données observées à un bruit près, donc pouvant paraitre irrégulières, mais représentant un phénomène supposé régulier (voir par exemple le graphique (b) de la Figure 1.1).

1.3 Historique et références

On ne peut prétendre résumer en quelques lignes l'engouement suscité par la statistique pour données fonctionnelles, tant la littérature scientifique est foisonnante sur le sujet. Ce paragraphe ne fait donc que citer quelques références, sans prétendre à l'exhaustivité.

Historiquement, c'est probablement d'abord en chimie ou en météorologie que les chercheurs ont été confrontés à ce type de données. Les premiers travaux de la communauté mathématique généralement évoqués sur le sujet sont ceux de Deville (1974) qui introduit l'Analyse en Composantes Principales (ACP dans la suite) pour les courbes, puis Dauxois et Pousse (1976); Besse et Ramsay (1986) qui proposent des études asymptotiques des propriétés de l'ACP fonctionnelle, encore largement citées aujourd'hui. Un autre travail fondateur est

celui de Dauxois *et al.* (1982), proposant un cadre théorique pour les variables fonctionnelles. La plupart des méthodes statistiques d'analyse exploratoire ont ensuite été déclinées pour les données fonctionnelles (analyse canonique et discriminante...), ainsi que l'analyse prédictive : modèle linéaire (Ramsay et Dalzell, 1991; Cardot *et al.*, 1999), modèle linéaire généralisé. Outre les contributions de la communauté française (voir références précédentes), la communauté américaine a largement contribué à populariser l'analyse de données fonctionnelles, en témoignent les ouvrages généraux de Ramsay et Silverman (2002, 2005); Ramsay *et al.* (2009) (la première édition du livre de 2005 datant de 1997) et le développement d'un package R spécifique `fda`. La statistique non-paramétrique pour des modèles de données fonctionnelles, en particulier la régression, s'est également largement développée dans la dernière décennie : l'une des références majeures sur le sujet est l'ouvrage de Ferraty et Vieu (2006). Citons également les contributions de la communauté espagnole, en particulier via un autre package R, `fda.usc`. Enfin, on pourra se faire une idée de l'état des recherches récentes sur le sujet, en consultant la monographie de Ferraty et Romain (2011), ou encore les divers numéros spéciaux consacrés à cette branche de la statistique par des revues scientifiques classiques (González Manteiga et Vieu, 2007; Valderrama, 2007; Ferraty, 2010; Goia et Vieu, 2016).

1.4 Suite du cours

La statistique pour données fonctionnelles requiert des bases mathématiques solides, relatives à la fois aux espaces fonctionnels sous-jacents (l'"analyse fonctionnelle" des mathématiciens), et aux probabilités nécessaires à l'étude des processus en temps continu. Quelques bases, simplifiées, avec des détails pouvant être omis en première lecture, seront présentées au Chapitre 2. Ceci permettra également d'introduire la représentation des fonctions comme somme de séries de fonctions de bases, les approximations associées, et d'évoquer la simulation de données fonctionnelles. Ensuite, partant des observations discrétisées évoquées plus haut, la première étape de toute étude de données fonctionnelles consiste en la reconstitution des données sous leur forme continue (recherche de la meilleure -en un certain sens- approximation des données par un processus continu). Le principe, et quelques techniques de lissage associées, feront l'objet du Chapitre 3. Le dernier chapitre sera enfin consacré à un début d'étude statistique à proprement parler de données fonctionnelles : analyse descriptive et exploratoire (ACP).

Chapitre 2

Bases mathématiques

Il n'est pas question, pour une introduction aux données fonctionnelles, de faire un cours complet de topologie, puis d'analyse fonctionnelle. On se contentera donc de quelques éléments très restreints. Pour plus de détails, on pourra se référer à Briane et Pages (2000) pour tout ce qui concerne l'intégration, la théorie de la mesure, les espaces L^p, et Hirsch et Lacombe (1997) ou Brezis (1983) pour l'analyse fonctionnelle à proprement parler et en particulier la théorie des opérateurs.

2.1 Quelques éléments d'analyse fonctionnelle

2.1.1 Espaces vectoriels normés

Nous avons défini une variable aléatoire fonctionnelle X comme une variable à valeurs dans un espace de fonctions $\mathcal{F}$ (Définition 1.1). Il va être fondamental de considérer uniquement des espaces vectoriels normés, pour pouvoir déterminer des développements de la fonction aléatoire X dans une base donnée (voir Section 2.3). Rappelons qu'un *espace vectoriel* (réel) E est dit *normé* s'il est muni d'une application $\|\cdot\| :\to \mathbb{R}_+$ qui est une *norme*, c'est-à-dire vérifie les 3 hypothèses suivantes :

séparation $\forall x \in E,\ \|x\| = 0 \ \Rightarrow\ x = 0$,

homogénéité $\forall x \in E,\ \lambda \in \mathbb{R},\ \|\lambda x\| = |\lambda|\|x\|$,

inégalité triangulaire $\forall x, y \in E,\ \|x + y\| \leq \|x\| + \|y\|$.

On peut définir plusieurs normes sur un même espace vectoriel. Si l'espace E est de dimension finie, toutes les normes sont *équivalentes* : si $\|\cdot\|_a$ et $\|\cdot\|_b$ sont 2 normes sur E, il existe 2 constantes $c, C > 0$ telles que pour tout $x \in E$, $c\|x\|_a \leq \|x\|_b \leq C\|x\|_a$. Ce n'est plus le cas en dimension infinie, en particulier pour les espaces de fonctions.

On rappelle également la notion de *convergence d'une suite* dans un espace vectoriel normé $(E, \|\cdot\|)$: une suite $(x_n)_{n\in\mathbb{N}}$ de points de E a pour limite $x \in E$ dans $(E, \|\cdot\|)$, si $\lim_{n\to\infty} \|x_n - x\| = 0$. En dimension finie, si une suite de E converge pour une norme donnée, elle converge au sens de toutes les autres. Ce n'est pas le cas en dimension infinie, où il est donc essentiel de spécifier la norme utilisée.

Toute suite convergente dans $(E, \|\cdot\|)$ est *de Cauchy*[1], mais la réciproque n'est pas

1. Une suite $(x_n)_n \subset E$ est de Cauchy dans E si

$$\forall \varepsilon > 0,\ \exists n_0 \in \mathbb{N},\ \forall p, q \geq n_0,\ ||x_p - x_q|| \leq \varepsilon.$$

toujours vraie. Un espace vectoriel normé $(E, \|\cdot\|)$ est dit *complet* si toute suite de Cauchy dans $(E, \|\cdot\|)$ converge. On parle alors d'*espace de Banach*. Tout espace de dimension finie est complet. En dimension infinie, il existe des espaces non complet.

Enfin, un espace vectoriel normé est dit *séparable* s'il contient un sous-ensemble dénombrable dense, c'est-à-dire s'il n'est pas "trop grand" (ce qui ne veut pas dire qu'il n'est pas de dimension infinie !).

Exemples utiles pour le cours.

Espace normé de dimension finie. L'exemple le plus classique d'espace vectoriel normé de dimension finie est $\mathbb{R}^d$, muni de la norme dite euclidienne usuelle :

$$\forall x = {}^t(x_1, \dots, x_d) \in \mathbb{R}^d, \quad \|x\| = \left(\sum_{j=1}^{d} x_j^2 \right)^{1/2}. \tag{2.1}$$

Espaces fonctionnels (dimension infinie). Pour simplifier, on se contente de considérer des fonctions définies sur $[0,1]$ et à valeurs réelles, même s'il est plus fréquent de s'intéresser aux fonctions définies sur $\mathbb{R}^d$.

Espace des fonctions continues sur un compact. On note $\mathcal{C}^0([0,1])$ l'espace des fonctions continues sur $[0,1]$. C'est un espace de Banach séparable pour la norme infinie, définie par : $\forall x \in \mathcal{C}^0([0,1])$, $\|x\|_\infty = \sup_{t\in[0,1]} |x(t)|$.

Espace des fonctions continument dérivables. On note plus généralement, pour tout entier $m \geq 1$

$$\mathcal{C}^m([0,1]) = \left\{ x \,:\, [0,1] \,\to\, \mathbb{R},\ x^{(m)} \in \mathcal{C}^0([0,1]) \right\}.$$

Il est également possible de munir cet espace d'une norme qui en fait un Banach. Dans le cas où $m = 1$ par exemple, on pose $\|x\| = \|x\|_\infty + \|x'\|_\infty$.

Espaces L^p. Pour simplifier, on définira l'espace $L^p([0,1])$ comme l'espace des fonctions de puissance $p-$ième intégrable sur $[0,1]$, pour $1 \leq p < \infty$:

$$L^p([0,1]) = \left\{ x \,:\, [0,1] \,\to\, \mathbb{R},\ \int_{[0,1]} |x(t)|^p dt < \infty \right\}. \tag{2.2}$$

Le théorème de Riesz-Fischer prouve que les espaces L^p, munis de la norme $\|x\|_p = (\int_{[0,1]} |x(t)|^p dt)^{1/p}$, $x \in L^p([0,1])$, sont des espaces de Banach séparables.

Espaces de Sobolev. On note, pour $m \geq 1$ un entier et $1 \leq p < \infty$,

$$\mathcal{W}_p^m([0,1]) = \left\{ x \in L^p([0,1]),\ \forall j \in \{1, \dots, m\},\ x^{(j)} \in L^p([0,1]) \right\}.$$

Autres espaces. Espaces des fonctions $\mathcal{C}^\infty$ à support compact, des fonctions à décroissance rapide, de Hölder, de Besov, etc $\cdots$

2.1.2 Espaces de Hilbert

Nous allons très vite nous restreindre à des espaces de Banach particulier, dont nous rappelons la définition et quelques propriétés ici. On appelle d'abord *espace préhilbertien réel* un espace vectoriel E sur $\mathbb{R}$ muni d'un *produit scalaire*, c'est à dire d'une forme $\langle \cdot, \cdot \rangle$, application de $E \times E$ dans $\mathbb{R}$ qui est

1. bilinéaire : linéaire par rapport à chacune de ses variables,
2. symétrique : $\forall x, y \in E,\ \langle x, y\rangle = \langle y, x\rangle$,
3. positive : $\forall x \in E,\ \langle x, x\rangle \geq 0$,
4. définie : $\forall x \in E,\ \langle x, x\rangle = 0 \ \Rightarrow\ x = 0$.

Deux éléments x, y de $(E, \langle\cdot,\cdot\rangle)$ sont dits *orthogonaux* si $\langle x, y\rangle = 0$. Tout espace préhilbertien $(E, \langle\cdot,\cdot\rangle)$ est un espace vectoriel normé : en posant, quelque soit $x \in E$, $\|x\| = \sqrt{\langle x, x\rangle}$, on définit bien une norme. Deux éléments de E sont dits orthonormés s'ils sont orthogonaux et de norme 1.

Un *espace de Hilbert* est un espace préhilbertien qui est complet. C'est donc un espace de Banach dans lequel la norme découle d'un produit scalaire par la formule ci-dessus.

Exemples utiles pour le cours.

Espace euclidien $\mathbb{R}^d$. L'espace vectoriel $\mathbb{R}^d$ est un espace de Hilbert de dimension finie, c'est-à-dire un espace euclidien, pour le produit scalaire $\langle x, y\rangle = \sum_{j=1}^d x_j y_j$ ($x = {}^t(x_1, \ldots, x_d) \in \mathbb{R}^d$, $y = {}^t(y_1, \ldots, y_d) \in \mathbb{R}^d$). La norme euclidienne définie par (2.1) est la norme découlant du produit scalaire.

Espace L^2. L'espace $L^2([0,1])$ des fonctions de carré intégrable sur $[0,1]$ (voir (2.2)), muni de $\langle x, y\rangle = \int_{[0,1]} |x(t)y(t)|dt$, pour $x, y \in L^2([0,1])$ est un espace de Hilbert.

Le résultat le plus important qui nous sera utile est le suivant :

Théorème 2.1 *Soit $(H, \langle\cdot,\cdot\rangle)$ un espace de Hilbert séparable.*

(i) H admet une (des) base(s) hilbertienne(s) dénombrable(s), c'est-à-dire une famille $\{\varphi_j,\ j \in \mathbb{N}\}$ d'éléments orthonormés de H totale, au sens où l'adhérence de l'espace vectoriel engendré par cette famille est égale à l'espace H tout entier. Cela signifie que, tout élément de H se décompose de façon unique sous la forme

$$\forall x \in H,\ x = \sum_{j\in\mathbb{N}} \langle x, \varphi_j\rangle \varphi_j.$$

(ii) (Riesz) Toute forme linéaire sur H, $f : H \to \mathbb{R}$, peut se mettre sous la forme $f = \langle\cdot, x\rangle$ pour un élément x de H. On dit que H est isomorphe à son dual H^*.

Les espaces de Hilbert généralisent donc de manière naturelle les espaces euclidiens. On pourra également utiliser la notion de *projection orthogonale* dans ce type d'espace.

2.2 Processus stochastiques : espérance et covariance

La question posée dans cette section est celle de l'extension des notions d'espérance et de matrice de covariance d'un vecteur aléatoire aux variables à valeurs dans un espace de dimension infinie. On suppose dans la suite que $\mathcal{F}$ est un espace de Banach, plus précisément un espace de fonctions définies sur un ensemble T et à valeurs réelles, dont on note $\|\cdot\|$ la norme, et muni de la tribu borélienne, c'est-à-dire la tribu associée à la norme, $(\Omega, \mathcal{A}, \mathbb{P})$ un espace probabilisé, et X une variable aléatoire fonctionnelle sur Ω, à valeurs dans $\mathcal{F}$.

2.2.1 Espérance

On peut définir au moins deux notions d'espérance pour un processus stochastique. Une première manière rigoureuse et générale consiste à introduire l'intégrale de Bochner qui généralise l'intégrale de Lebesgue pour des fonctions à valeurs dans un espace de Banach (voir Dunford et Schwartz 1958).

Définition 2.1 *Si* $\mathbb{E}[\|X\|] < \infty$*, on peut définir l'****espérance*** *de* X *comme*

$$\mathbb{E}[X] = \int_\Omega X(\omega) d\mathbb{P}(\omega),$$

*où l'intégrale est l'****intégrale de Bochner*** *de* X*. On définit ainsi un élément de* $\mathcal{F}$*. On dira que* X *est centrée lorsque* $\mathbb{E}[X] = 0$*.*

Cette espérance a des propriétés "classiques" : par exemple, $\|\mathbb{E}[X]\| \leq \mathbb{E}[\|X\|]$, ou $\|\mathbb{E}[X]\|^2 \leq \mathbb{E}[\|X\|^2]$. Lorsque $\mathcal{F}$ est un espace de Hilbert, de produit scalaire $\langle \cdot, \cdot \rangle$, on a en fait plus généralement $\mathbb{E}[\langle X, f \rangle] = \langle \mathbb{E}[X], f \rangle$.

Une autre possibilité pour définir l'espérance d'une variable fonctionnelle consiste à prendre point par point l'espérance de X : on se contentera de la définition (légèrement imprécise) suivante.

Définition 2.2 *Lorsque cela a un sens, on peut définir l'****espérance*** *de* X *comme la fonction non aléatoire* $(\mathbb{E}[X])(\cdot)$ *(élément de* $\mathcal{F}$*) définie point par point par*

$$(\mathbb{E}[X])(t) = \mathbb{E}[X(t)] = \int_\Omega X(t, \omega) d\mathbb{P}(\omega),$$

Ces deux notions d'espérance coïncident souvent. Par exemple, si $\mathcal{F} = L^2(T)$, l'espérance point à point de X existe et définit bien un élément de $L^2(T)$ dès que $\mathbb{E}[\|X\|] < \infty$, et on a égalité presque partout des deux notions d'espérance dans ce cas.

2.2.2 Opérateur de covariance et fonction de covariance

Dans le cadre multivarié, la covariance a une forme matricielle : l'extension naturelle à la dimension infinie est donc un opérateur. On se restreint à présent au cas où $\mathcal{F}$ est un espace de Hilbert.

Définition 2.3 *Supposons que* $\mathbb{E}[\|X\|^2] < \infty$*. L'****opérateur de covariance*** *de* X *est défini par*

$$\Gamma : f \in \mathcal{F} \longmapsto \Gamma f = \mathbb{E}[\langle X - \mathbb{E}[X], f \rangle (X - \mathbb{E}[X])].$$

Il est à valeurs dans $\mathcal{F}$*.*

Dans le cas où X est centrée, on a la réécriture plus simple, $\Gamma : f \in \mathcal{F} \longmapsto \mathbb{E}[\langle X, f \rangle X]$. Notons qu'une telle définition est en fait possible également dans le cas où $\mathcal{F}$ est seulement un espace de Banach. On remplace alors le produit scalaire par le *crochet de dualité*, Γ appliquant

alors $\mathcal{F}^*$ dual de $\mathcal{F}$ dans $\mathcal{F}$. Mais les propriétés qui suivent vont justifier qu'on se borne au cas hilbertien. Dans tous les cas, Γ est un *opérateur linéaire* (ceci découle de la linéarité du produit scalaire), et *continu*, puisque $\|\Gamma f\| \leq \mathbb{E}[\|X\|^2]\|f\|$ (en utilisant l'inégalité de Cauchy-Schwarz). On peut donc voir l'opérateur de covariance comme une matrice de dimension infinie, extension directe de la notion de matrice de variance-covariance d'un vecteur aléatoire.

On a, dans le cas hilbertien, le résultat suivant, que l'on pourra omettre si l'on n'est pas familier de la théorie des opérateurs, pour n'en retenir que les conséquences (voir section suivante).

Proposition 2.1 *Supposons que X est une variable aléatoire à valeurs dans un espace de Hilbert séparable $\mathcal{F}$ telle que $\mathbb{E}[\|X\|^2] < \infty$. Alors, son opérateur de covariance Γ est*
(i) ***autoadjoint*** *: $\forall f, g \in \mathcal{F}$, $\langle \Gamma f, g\rangle = \langle f, \Gamma g\rangle$,*
(ii) ***de Hilbert-Schmidt*** *: il existe une base hilbertienne $(\varphi_j)_{j\in\mathbb{N}}$ telle que $\sum_{j\in\mathbb{N}} |\Gamma\varphi_j|^2 < \infty$.*

On en déduit,

Corollaire 2.1 *Avec les hypothèses et notations de la proposition précédente, l'opérateur de covariance Γ de X est ainsi*
(i) ***compact*** *(l'image de la boule unité de $\mathcal{F}$ par Γ est relativement compacte),*
(ii) ***diagonalisable en base orthonormée****.*

Le premier point est une conséquence du caractère Hilbert-Schmidt de Γ. Le second point provient du résultat fondamental de décomposition des opérateurs autoadjoints compacts. Nous tirerons les conséquences de cette propriété à la Section 2.2.3 ci-dessous. Enfin, on notera que si Γ est de rang fini, alors X appartient en fait un espace de dimension finie.

Une autre quantité que l'opérateur de covariance est fréquemment utilisée en théorie des processus stochastiques. Pour simplifier (encore), on suppose que $\mathcal{F}$ est l'espace $L^2([0,1])$.

Définition 2.4 *Soit X une variable aléatoire à valeurs dans $L^2([0,1])$. La* ***fonction de covariance*** *de X est l'application*

$$C : (s,t) \in [0,1]^2 \longmapsto C(s,t) = \mathbb{E}[X(t)X(s)].$$

Le lien entre fonction de covariance et opérateur de covariance est donné par le résultat suivant.

Proposition 2.2 *Soit X une variable aléatoire à valeurs dans $L^2([0,1])$, d'opérateur de covariance Γ et de fonction de covariance C. Alors,*

$$\forall f \in L^2([0,1]),\ \forall t \in [0,1],\ (\Gamma f)(t) = \int_{[0,1]} C(s,t)f(s)ds.$$

Ainsi, Γ est un ***opérateur à noyau****, de noyau la fonction de covariance.*

2.2.3 Décomposition de Karhunen-Loève

On considère ici X une variable aléatoire à valeurs dans un espace de Hilbert $(\mathcal{F}, \langle \cdot, \cdot \rangle)$, d'opérateur de covariance Γ. Nous avons vu que Γ est diagonalisable (Corollaire 2.1). Ainsi, il existe une base hilbertienne de $\mathcal{F}$, formée de vecteurs propres $(\psi_j)_{j\in\mathbb{N}}$ de Γ associés aux valeurs propres $(\lambda_j)_{j\in\mathbb{N}}$ rangées par ordre décroissant (ie. $\Gamma\psi_j = \lambda_j\psi_j$). Ces valeurs propres sont positives. On peut alors développer X dans cette base hilbertienne (voir Théorème 2.1).

Définition 2.5 *On appelle* ***développement de Karhunen-Loève*** *d'une variable X à valeurs dans un espace de Hilbert telle que $\mathbb{E}[\|X\|^2] < \infty$, son développement dans la base de fonctions propres de l'opérateur de covariance associé. Avec les notations précédentes, on obtient l'écriture en série aléatoire suivante :*

$$X = \mathbb{E}[X] + \sum_{j=1}^{\infty} \langle X, \psi_j \rangle \psi_j = \mathbb{E}[X] + \sum_{j=1}^{\infty} \sqrt{\lambda_j}\xi_j\psi_j,$$

où $\xi_j = \langle X, \psi_j \rangle / \sqrt{\lambda_j}$.

Dans le cas où X est centrée, on voit facilement que les variables aléatoires réelles ξ_j sont centrées, réduites et décorrélées. Lorsqu'en pratique on dispose de données fonctionnelles, nous n'avons bien évidemment pas accès, ni à l'espérance, ni à l'opérateur de covariance, ni à la décomposition de Karhunen-Loève d'un processus. On définira donc des estimateurs pour ces quantités (équivalents des moyennes empiriques et matrice de variance empirique du cas multivarié) au Chapitre 4. Il existe toutefois certains cas où l'on connait ces quantités, au moins de manière théorique, c'est le cas de certains processus gaussiens.

2.2.4 L'exemple des processus gaussiens

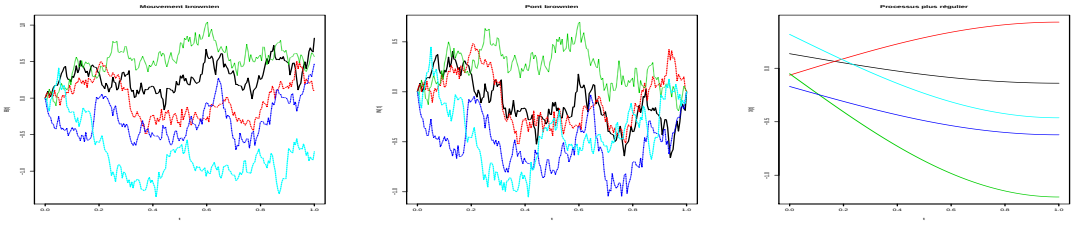
Les processus gaussiens constituent une classe importante de processus, très étudiés en probabilité, et largement utilisés.

Définition 2.6 *Une variable aléatoire fonctionnelle X sur un espace de Hilbert $\mathcal{F}$ est dite gaussienne, si pour tout élément $f \in \mathcal{F}$, $\langle X, f \rangle$ est une variable aléatoire réelle distribuée selon la loi normale. Si X est une fonction aléatoire d'un intervalle T de $\mathbb{R}$, on dit que X est gaussienne si pour tout $r \in \mathbb{N}\backslash\{0\}$ et pour tout $(t_1, \ldots, t_r) \in T^r$, le vecteur $(X(t_1), \ldots, X(t_r))$ est gaussien, ie. pour tout choix de réels $(\alpha_j)_{1\leq j\leq r}$, $\sum_{j=1}^{r} \alpha_j X(t_j)$ est une variable gaussienne.*

Les deux définitions coïncident dans certains cas, voir Ibragimov et Rozanov (1978). On notera que si X est gaussien, les variables ξ_j intervenant dans la décomposition de Karhunen-Loève de X ne sont plus seulement décorrélées, elles sont indépendantes, et suivent de plus la loi normale $\mathcal{N}(0, 1)$.

Exemples classiques.

Mouvement brownien standard W restreint à $[0, 1]$. Ce processus gaussien vérifie de plus les propriétés suivantes :



(a) (b) (c)

FIGURE 2.1 – Trajectoires simulées de processus gaussiens : (a) Mouvement brownien (b) Pont brownien (c) Processus $X(t) = \xi_1 + \sqrt{2}\xi_2 \sin(\pi t/2)$, avec $\xi_1, \xi_2 \sim \mathcal{N}(0,1)$ indépendantes.

— trajectoires presque sûrement continues : la fonction $t \mapsto W(t)$ est presque sûrement continue,
— processus centré, et de fonction de covariance $C(s,t) = \min(s,t)$, pour tous $s,t \in [0,1]$.
— $W(0) = 0$ presque sûrement.

Sa décomposition de Karhunen-Loève est connue (Ash et Gardner, 1975) :

$$W(t) = \sum_{j=1}^{\infty} \frac{1}{\pi(j-1/2)} \xi_j \psi_j(t), \quad \psi_j(t) = \sqrt{2}\sin(\pi(j-1/2)t).$$

Pont brownien B Il est défini par $B(t) = W(t) - tW(1)$. Sa décomposition de Karhunen-Loève est également connue (MacNeill, 1978; Deheuvels, 2007) :

$$B(t) = \sum_{j=1}^{\infty} \frac{1}{\pi j} \xi_j \psi_j(t), \quad \psi_j(t) = \sqrt{2}\sin(\pi j t).$$

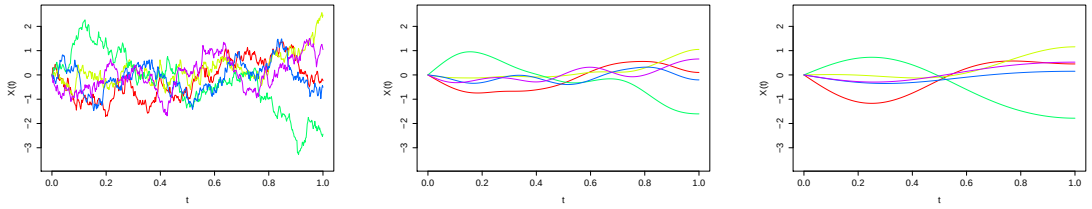
Ces formules, et plus généralement l'existence de la décomposition de Karhunen-Loève, nous permettent de simuler de larges classes de courbes aléatoires, dont des trajectoires du mouvement brownien : on tronque la série à un indice J fini pour approcher le processus, on simule les réalisations des variables $(\xi_j)_{1 \le j \le J}$ selon une loi donnée, on choisit des fonctions de base, des valeurs propres, on effectue la combinaison linéaire, et on représente le résultat. Quelques trajectoires sont représentées aux Figures 2.1 et 2.2. On s'aperçoit que la vitesse de décroissance des valeurs propres de l'opérateur de covariance mesure la régularité des courbes simulées : plus celles-ci décroissent vite, plus les trajectoires sont régulières.

***Exercice 1.* Autre simulation du mouvement brownien.** La méthode ci-dessus de simulation de trajectoires du mouvement brownien à partir de la décomposition de Karhunen-Loève n'est pas la plus classique. On utilise plutôt généralement le fait que si W est un brownien indexé par $\mathbb{R}_+$, ses accroissements sont indépendants et stationnaires, de loi gaussienne : si on se donne $\{t_0 = 0 < t_1 < t_2 < \cdots < t_p\}$, les variables aléatoires réelles $W(t_1), W(t_2) - W(t_1), \ldots, W(t_p) - W(t_{p-1})$ sont indépendantes, et $W(t_j) - W(t_{j-1}) \sim \mathcal{N}(0, t_j - t_{j-1})$. Utiliser cette caractérisation pour écrire une fonction R simulant n trajectoires du mouvement brownien restreint à $[0,1]$, et tracer les réalisations obtenues.

2.2.5 Probabilités de petites boules

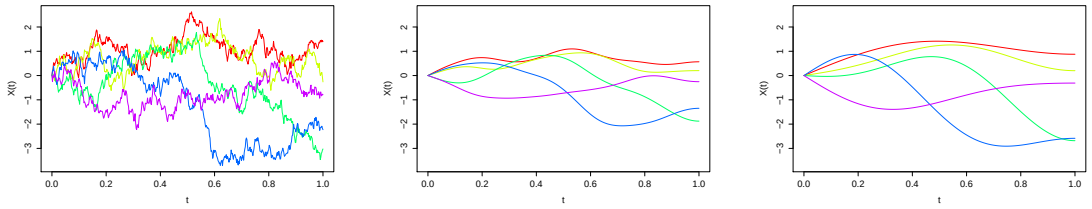
Des quantités importantes, dont le comportement influence grandement celui de certains estimateurs dans des modèles de données fonctionnelles, sont les probabilités de petites boules

Cas 1 : $\xi_j \sim \mathcal{N}(0,1)$



$\lambda_j = j^{-2}$ $\lambda_j = e^{-j}$ $\lambda_j = j^{-2}\mathbf{1}_{j\leq 3}$

Cas 2 : $\xi_j \sim \mathcal{U}([-\sqrt{3},\sqrt{3}])$



$\lambda_j = j^{-2}$ $\lambda_j = e^{-j}$ $\lambda_j = j^{-2}\mathbf{1}_{j\leq 3}$

FIGURE 2.2 – Trajectoires simulées de courbes aléatoires dans la base de Karhunen-Loève du mouvement brownien : $X(t) = \sum_{j\geq 1} \sqrt{\lambda_j}\xi_j\psi_j(t), \quad \psi_j(t) = \sqrt{2}\sin(\pi(j-0.5)t)$.

d'une variable fonctionnelle X.

Définition 2.7 *Soit X une variable aléatoire fonctionnelle à valeurs dans un espace de Banach $\mathcal{F}$. On appelle* ***probabilités de petites boules*** *de X ("small ball probabilities") les quantités suivantes*

$$\varphi^{x_0}(h) := \mathbb{P}(\|X - x_0\| \leq h), \quad x_0 \in \mathcal{F}, \quad h > 0.$$

Le comportement de ces probabilités est complexe et très étudié : de nombreux auteurs proposent des encadrements pour ces quantités, aussi bien dans le cas particulièrement étudié des processus gaussiens (une revue des résultats existant figure dans Li et Shao 2001), mais aussi dans le cas de processus plus généraux s'écrivant comme somme de variables indépendantes (Lifshits, 1997; Dunker *et al.*, 1998; Mas, 2012). On retiendra uniquement que ces probabilités décroissent très rapidement vers 0 quand h, le rayon de la petite boule, tend vers 0. Sous des hypothèses générales, Mas (2012) a prouvé que pour tout $p \in \mathbb{N}$, $\lim_{h\to 0} h^{-p}\varphi^{x_0}(h) = 0$. La décroissance est donc plus rapide qu'une vitesse polynomiale. Par ailleurs, on peut également déduire de ses résultats que s'il existe $p \in \mathbb{N}$ tel que $\varphi^{x_0}(h)$ est de l'ordre de h^p, alors en fait la variable X vit dans un espace de dimension finie : dans le cas hilbertien, cela signifie que l'opérateur de covariance admet un nombre fini de valeurs propres non nulles.

Dans le cas hilbertien, on peut relier le comportement de ces probabilités de petites boules avec la décomposition de Karhunen-Loève. Les notations sont celles de la définition 2.5. Hoffmann-Jørgensen *et al.* (1979) ont par exemple montré les résultats suivants :

(i) Si $\lambda_j = j^{-2a}$, $a > 1/2$, alors il existe des constantes $C_1, C_2, c_1, c_2 > 0$ telles que

$$C_1 h^{\rho(1-a)} e^{-c_1 h^{-2\rho}} \leq \varphi^0(h) \leq C_2 h^{\rho(3-a)} e^{-c_2 h^{-2\rho}}, \quad \rho = 1/(2a-1).$$

(ii) Si $\lambda_j = e^{-2j}/j$, alors il existe des constantes $C_1, C_2 > 0$ telles que

$$C_1 h^{-3/2} \log^{-1/4}(1/h) e^{-\log(1/h)^2/2} \leq \varphi^0(h) \leq C_2 h^{1+\log 2} \log^{-1/4}(1/h) e^{-\log(1/h)^2/2}.$$

2.3 Représentation des fonctions dans une base

Écrire une fonction comme combinaison linéaire de fonctions élémentaires avec des coefficients aléatoires permet, on l'a déjà vu ci-dessus, de simuler un grand nombre de processus. On se propose tout d'abord, dans l'exercice suivant d'implémenter ainsi la simulation de courbes aléatoires d'expression simple.

Exercice 2. Écrire, pour chacune des courbes suivantes, une fonction R permettant de simuler un échantillon de n courbes aléatoires discrétisées en p points :

1. $X(t) = \xi_0 \cos(2\pi t) + \xi_1 \sin(4\pi t) + \xi_3 (t - 0.5)(t - 0.25)$ avec $\xi_1, \xi_2, \xi_3 \sim \mathcal{U}_{[0,1]}$, indépendantes (Ait-Saï di *et al.*, 2008).
2. $X(t) = \xi_0 + \xi_1 t + \xi_2 \exp(t) + \sin(\xi_3 t)$ avec $\xi_1 \sim \mathcal{U}_{[0,100]}$, $\xi_2 \sim \mathcal{U}_{[-30,30]}$, $\xi_3 \sim \mathcal{U}_{[0,10]}$, et $\xi_4 \sim \mathcal{U}_{[1,3]}$ indépendantes (Ferraty et Vieu, 2002).

Pour chacune des fonctions, illustrer le résultat obtenu.

La suite de cette section concerne plus précisément les développements de fonctions (aléatoires) comme combinaison linéaire de fonctions dont on sait qu'elles ont de bonnes propriétés d'approximation.

2.3.1 Principe

On se place dans le cas de variables aléatoires à valeurs dans $\mathcal{F} = L^2(T)$, où T est un intervalle de $\mathbb{R}$. Les sections précédentes montrent l'importance, pour une variable fonctionnelle, de la décomposition de Karhunen-Loève. Celle-ci étant en général inconnue, il va être important de pouvoir approcher les variables fonctionnelles comme combinaisons linéaires de fonctions d'une base donnée. Rappelons qu'un système de fonctions $(\varphi_j)_{j\in\mathbb{N}}$ forme une base hilbertienne s'il est composé de fonctions orthonormales (donc linéairement indépendantes), qui ont la propriété suivante : en prenant une combinaison linéaire d'un nombre suffisamment grand de ces fonctions, on peut approcher n'importe quelle autre fonction de l'espace $\mathcal{F}$.

Plus précisément, si X est une variable à valeurs dans $\mathcal{F} = L^2(T)$, on a l'égalité suivante, dans $L^2(T)$,

$$X = \sum_{j=1}^{\infty} \theta_j \varphi_j,$$

la convergence de la série ayant lieu dans $L^2(T)$. L'idée naturelle pour approcher X est alors de tronquer la série à un certain niveau D, et d'estimer les coefficients : il s'agit donc d'approcher X par $\widetilde{X}$, définie par

$$\widetilde{X}(t) = \sum_{j=1}^{D} \widetilde{\theta}_j \varphi_j(t), \quad t \in T$$

On obtient ainsi une approximation appartenant à un sous-espace de dimension finie D de $L^2(T)$ (l'espace vectoriel engendré par les D premières fonctions de la base choisie). En notant $\widetilde{\theta} = {}^t(\widetilde{\theta}_1, \ldots, \widetilde{\theta}_D)$ le vecteur (colonne) des coefficients et $\Phi(t) = {}^t(\varphi_1(t), \ldots, \varphi_D(t))$ le vecteur dont les éléments sont les D premiers éléments de la base évalués en t, on a la représentation matricielle suivante :

$$\widetilde{X}(t) = {}^t\widetilde{\theta}\Phi(t),$$

utile pour l'implémentation. Lorsque l'on souhaite approcher une fonction, les coefficients θ sont bien évidemment à estimer, mais ce ne sont pas les seuls paramètres inconnus : il faut voir aussi D, la dimension de l'espace d'approximation, comme un paramètre à choisir (voir Chapitre 3 et en particulier les Sections 3.2 et 3.3). Le choix de la base $(\varphi_j)_j$ dépend des hypothèses *a priori* sur la variable à étudier. On en présente brièvement quelques une, avec lesquelles le package `fda` permet de travailler. En particulier, le choix de la base peut dépendre du fait qu'on veut ou non estimer les dérivées de X. Une représentation précise et fidèle de X à l'aide de nombreuses oscillations peut par exemple mener à une très mauvaise représentation des dérivées.

2.3.2 Bases classiques

Codes R. Les fonctions du package `fda` dont le nom est de type `create.Nom_Base.basis` permettent de générer des systèmes de fonctions classiques, dont le nom est précisé par `Nom_Base`, et qui vont nous servir par la suite. L'objet créé a pour type R `basisfd`, mais nous n'aurons pas besoin de nous soucier des propriétés détaillées. Le premier argument de toutes ces fonctions est `rangeval=`, qui permet de spécifier les extrémités de l'intervalle sur lequel on travaille, sous forme d'un vecteur à 2 composantes. Les arguments suivants sont spécifiques à la base choisie.

2.3.2.1 Base de Fourier

La *base de Fourier*, ou *base trigonométrique*, est probablement la plus connue des bases hilbertiennes, et la plus appropriée pour approcher des courbes (aléatoires) au comportement périodique (comme les phénomènes d'évolution de température par exemple). Elle est définie de la manière suivante : quel que soit t,

$$\psi_0(t) = 1, \;\; \psi_{2k-1}(t) = \sin(k\omega t), \;\; \psi_{2k}(t) = \cos(k\omega t) \quad k \in \mathbb{N}\backslash\{0\},$$

le paramètre ω déterminant la période, $2\pi/\omega$. Les coefficients du développement d'une fonction dans cette base sont appelés *coefficients de Fourier*. Les 10 premières fonctions de la base de Fourier, avec $\omega = 2\pi$, sont représentées à la Figure 2.3.

L'usage de cette base est particulièrement simple et attractif :

— la théorie mathématique sur le sujet est largement développée : la reconstruction d'une fonction à l'aide de sa série de Fourier est légitimée par un grand nombre de résultats de la théorie associée (résultats de convergence L^2 - en moyenne quadratique, résultats de convergence simple ou uniforme de Dirichlet pour les fonctions continues, dérivables, de dérivées continues par morceaux, ...) ;

— les séries de Fourier ont toujours été un outil essentiel en théorie du signal, permettant à la fois de reconstruire et d'analyser un signal périodique (interprétation des coefficients associés en terme de fréquence) ;

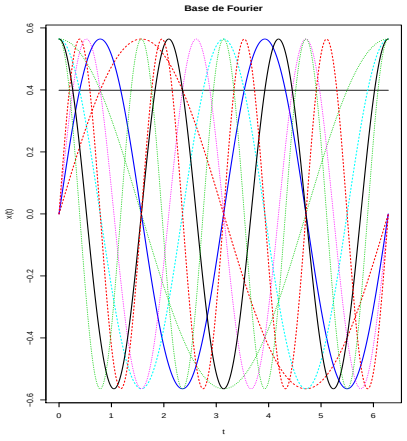


Figure 2.3 – Courbes représentatives des 10 premières fonctions de la base de Fourier, de période 2π.

— connaissant le développement d'une fonction régulière en série de Fourier, les coefficients des développements de ses dérivées successives dans la même base sont particulièrement simples à obtenir, puisque

$$(\sin(k\omega\cdot))' = k\omega\cos(k\omega\cdot), \quad (\cos(k\omega\cdot))' = -k\omega\sin(k\omega\cdot),$$

ce qui implique que si les coefficients du développement de X sont $(\theta_0, \theta_1, \ldots)$, alors ceux de X' sont

$$(0, \omega\theta_1, -\omega\theta_2, 2\omega\theta_3, -2\omega\theta_4, \ldots),$$

ceux de X'' sont

$$(0, -\omega^2\theta_1, -\omega^2\theta_2, -4\omega^2\theta_3, -4\omega^2\theta_4, \ldots),$$

et de manière similaire on obtient les coefficients des dérivées d'ordre supérieur.

— les coefficients de Fourier d'une fonction se calculent de manière extrêmement rapide et efficace, grâce au lien entre coefficients de Fourier et *transformée de Fourier discrète* (qui se calcule via un algorithme spécial appelé Transformée de Fourier Rapide).

La base de Fourier est bien adaptée pour reconstruire des fonctions extrêmement régulières, de courbure à peu près semblable en tout point. Elle ne convient pas pour des fonctions reflétant des discontinuités, en raison, entre autres, d'effets de bord important (oscillations d'amplitude importante). On parle de phénomène de Gibbs pour les effets de bords au voisinage des discontinuités de la fonction. On préférera alors utiliser des bases dites "localisées".

Code R. La fonction `create.fourier.basis` du package `fda` permet de générer les fonctions de la base de Fourier. Citons quelques paramètres importants :

`rangeval=` extrémités de l'intervalle sur lequel on travaille,

`nbasis=` nombre de fonctions de base (nombre impair, sinon un correction automatique est faite),

`period=` période (par défaut, longueur de l'intervalle sur lequel on travaille),

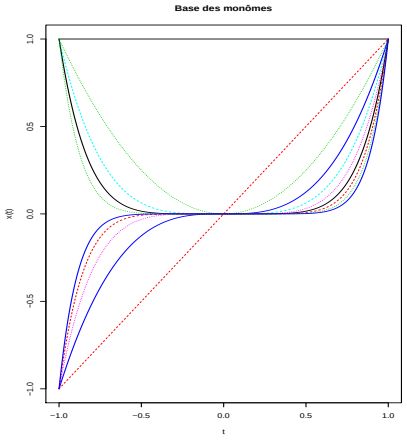


FIGURE 2.4 – Courbes représentatives des 10 premiers monômes.

2.3.2.2 Base polynomiale et base de B-splines

La *base des monômes* $\varphi_j(t) = (t-\omega)^j$, $j \geq 0$, où ω est le paramètre de translation, souvent choisi comme le milieu de l'intervalle T, a longtemps été la référence pour la reconstruction de fonctions non-périodiques, en raison de la simplicité d'estimation des coefficients. On représente, pour $\omega = 0$, les courbes des 10 premiers monômes sur $[-1, 1]$ à la Figure 2.4.

Code R. La fonction `create.monomial.basis` du package `fda` permet de générer les fonctions de la base des monômes. Citons deux paramètres importants :

`rangeval=` extrémités de l'intervalle sur lequel on travaille,

`nbasis=` nombre de fonctions de base (nombre impair, sinon un correction automatique est faite).

Cependant, la base des monômes a progressivement été remplacée, d'abord par des *bases de polynômes par morceaux* (définies par exemple à partir des polynômes de Legendre), que nous ne détaillerons pas (voir par exemple Comte 2015), ou des bases de splines. Il serait hors de propos de faire ici un cours entier sur les splines, nous donnons ici juste quelques éléments sur la notion de fonction spline, et une manière classique d'en construire. Pour une étude détaillée, on pourra se référer à de Boor (2001).

Définition 2.8 *Une* ***spline*** *sur un intervalle $T = [a, b]$ est une fonction possédant les propriétés suivantes :*

- *elle se caractérise par la donnée de L sous-intervalles de T délimités par des points appelés* ***noeuds*** *("knots") ou* ***points de rupture*** *("breakpoints") $\tau_0 = a \leq \tau_1 \leq \ldots \leq \tau_L = b$, non nécessairement répartis régulièrement dans T ;*
- *sur chaque sous-intervalle, la spline est un polynôme, d'****ordre*** *m [2] ;*
- *les liaisons entre les différents polynômes aux points de rupture sont régulières [3], et, plus précisément, pour une spline définie avec des polynômes d'ordre m, les dérivées doivent être continues sur T jusqu'à l'ordre $m-2$.*

On appelle ***spline cubique*** *une spline d'ordre 4.*

Notion de splines Ainsi, plus les splines ont un ordre élevé, plus elles sont régulières. Et plus on augmente le nombre de noeuds, plus on gagne en flexibilité et précision : pour approcher une fonction avec des splines, on aura tendance à placer plus de noeuds aux endroits de T où la fonction semble avoir le comportement le plus complexe. On notera également que toute combinaison linéaire de fonctions spline est encore une fonction spline. Retenons, pour la suite, qu'une fonction spline est polynomiale par morceaux, avec conditions de continuité sur la fonction et ses dérivées aux jointures.

Remarque. Les notions de "noeuds" et de "points de rupture" ne coïncident en fait pas tout à fait. Il est possible d'avoir plusieurs "points de rupture" qui coincident. Le terme de "point de rupture" se réfère précisément aux valeurs des noeuds répétées une seule fois, tandis que le terme "noeuds" se réfère aux points de rupture, répétés avec multiplicité : un point de rupture peut être égal à plusieurs noeuds.

Définition 2.9 *On appellera* ***base de spline d'ordre*** m ***et de séquence de noeuds*** τ *une famille de fonctions vérifiant les propriétés suivantes :*

(i) chaque fonction de base est elle-même une fonction spline (toute combinaison linéaire de ces fonctions est donc encore une fonction spline) ;

(ii) toute spline d'ordre m et de séquence de noeuds τ peut s'exprimer comme combinaison linéaire de ces fonctions de base ;

(iii) les fonctions de bases sont linéairement indépendantes (pas nécessairement orthonormées).

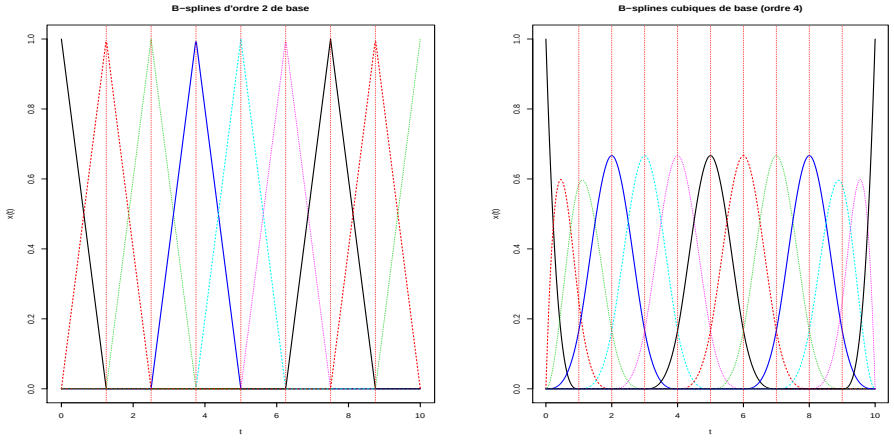
B-splines Il existe plusieurs systèmes classiques de telles bases. La plus célèbre et fréquemment utilisée est sans doute celle introduite par de Boor (2001), appelé système de B-splines. Un tel système est entièrement caractérisé par

- un ordre m ou, de manière équivalente, le degré maximal $m-1$ des morceaux de polynômes.
- une séquence de noeuds τ, qui entraine la connaissance des points de rupture (jointure) des sous-intervalles.

Nous ne définirons pas précisément les B-splines (sauf dans un cas particulier, voir ci-dessous), mais nous nous contenterons de citer les quelques propriétés spécifiques suivantes :

- Nombre de fonctions de base = ordre + nombre de noeuds intérieurs (τ_0 et τ_L ne sont pas comptés, avec les notations du paragraphe précédent).
- Support compact : une fonction B-spline de base d'ordre m est non-nulle et positive sur au plus m sous-intervalles, adjacents (ceci a des conséquences sur le calcul des coefficients d'une fonction spline dans la base des B-splines : la matrice des produits scalaires des fonctions de base entre elles est une matrice bande [4], avec au plus m diagonales non nulles).
- Forme : la forme des B-splines est définie par les noeuds. Pour des noeuds équidistants par exemple, toutes les fonctions de base ont la même forme.
- Baisse de continuité aux extrémités de l'intervalle T : quand on veut approcher une fonction sur un intervalle T, on ne connait généralement pas son comportement en dehors de l'intervalle et il peut être naturel de vouloir des fonctions moins régulières sur les bords, ceci est réalisable dans le cas des B-splines en choisissant une série de noeuds avec plusieurs noeuds égaux sur les bords.

4. Une matrice a une structure bande si ses coefficients non nuls sont concentrés sur un "petit" nombre de diagonales.



(a) (b)

FIGURE 2.5 – Courbes représentatives des fonctions de bases B-splines avec noeuds équidistants tous distincts sur $[0, 10]$. (a) Cas des splines d'ordre 2 (9 fonctions de base). (b) Cas des splines cubiques (13 fonctions de base).

— Somme des valeurs des fonctions de bases B-spline en tout point t égale à 1.
— Base de Riesz : la famille des B-splines n'est pas orthonormée, mais elle a des propriétés qui font qu'elles se rapprochent d'une telle base.

Code R. La fonction `create.bspline.basis` du package `fda` permet de générer les fonctions des bases de B-splines. Citons quelques paramètres importants :

`rangeval=` extrémités de l'intervalle sur lequel on travaille,

`nbasis=` nombre de fonctions de base (nombre impair, sinon un correction automatique est faite),

`norder=` ordre des splines considérées (4 par défaut, pour des splines cubiques),

`breaks=` liste des noeuds D'après les propriétés ci-dessus, le nombre de fonctions de base doit vérifier : `nbasis = norder + length(breaks) - 2`.

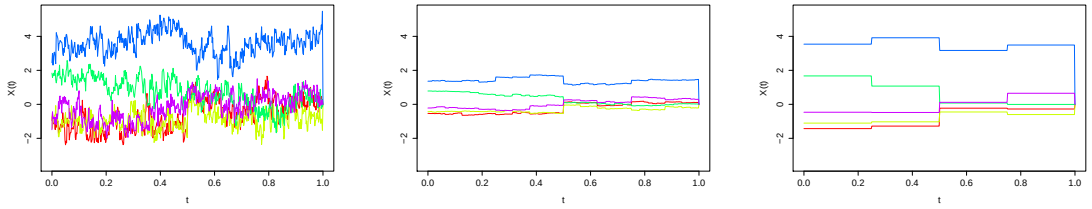
On représente à la Figure 2.5 les fonctions de 2 bases B-splines avec noeuds équidistants tous disctincts sur $[0, 10]$: ordre 2 (splines linéaires, graphique (a)), et ordre 4 (splines cubiques, graphique (b)).

2.3.2.3 Bases d'ondelettes

Nous rentrerons encore moins dans les détails que pour les bases de splines dans cette section. La théorie des ondelettes s'est développée plus récemment que les théories associées aux deux types de bases précédentes, et l'on pourra se référer à Härdle *et al.* (1998) pour une présentation détaillée.

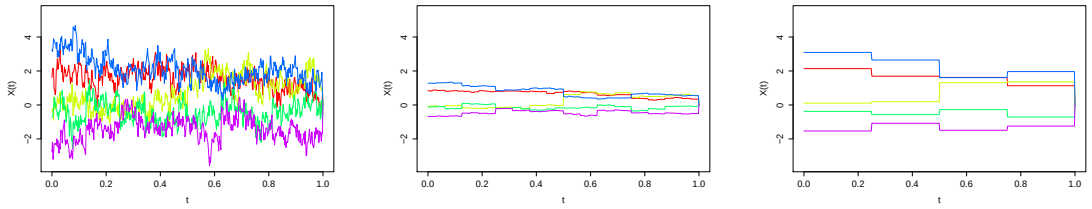
Les ondelettes combinent les avantages de la base de Fourier (approximation "en fréquence" des fonctions) et des splines (bases localisées). Une base d'ondelette est une famille totale dans $L^2(\mathbb{R})$, de fonctions non nécessairement orthonormées, mais formant une base de Riesz. Les fonctions de base sont définies par dilatation-translation d'une fonction ψ dite *mère des*

Cas 1 : $\xi_j \sim \mathcal{N}(0,1)$



$\lambda_j = j^{-2}$ $\lambda_j = e^{-j}$ $\lambda_j = j^{-2}\mathbf{1}_{j\leq 3}$

Cas 2 : $\xi_j \sim \mathcal{U}([-\sqrt{3},\sqrt{3}])$



$\lambda_j = j^{-2}$ $\lambda_j = e^{-j}$ $\lambda_j = j^{-2}\mathbf{1}_{j\leq 3}$

FIGURE 2.6 – Trajectoires simulées de courbes aléatoires dans la base de Haar : $X(t) = \sum_{j\geq 0}\sqrt{\lambda_j}\sum_{k=1}^{2^j}\xi_{j,k}\psi_{j,k}(t)$, $(\psi_{j,k})$ base de Haar.

ondelettes :

$$\psi_{j,k}(t) = 2^{j/2}\psi(2^j t - k), \quad t \in \mathbb{R}.$$

Le développement d'une fonction dans la base associée donne une *analyse multirésolution* au sens où le coefficient du développement d'indices j,k apporte des informations sur la fonction au voisinage de la position $2^{-j}k$ (localisation) à l'échelle 2^{-j} (fréquence proche de 2^j).

Grâce à la localisation et à des résultats mathématiques poussés, on peut espérer approcher finement une fonction par un développement dans une base d'ondelettes avec seulement un petit nombre de coefficients non nuls à calculer.

À titre d'exemple, la Figure 2.6, montre des courbes aléatoires obtenues comme combinaisons linéaires de fonctions de la base de Haar, l'une des bases d'ondelettes les plus simples.

2.3.3 Création d'objets fonctionnels avec le package `fda` de R

Le package `fda` de R permet de créer des objets de la classe "fd" : ce sont des objets de type fonctionnels. On ne rentrera pas dans les détails des objets en eux mêmes, mais on verra comment en construire, et comment les utiliser. En particulier, et comme on l'a déjà vu un petit peu sur les graphiques, on peut construire de tels objets en choisissant un système de fonctions de bases φ_j, $j = 1,\ldots,D$ et des coefficients θ_j puis en faisant appel à la fonction `fd`. C'est ce que l'on détaille brièvement ici.

Code R. Création et représentation d'objets fonctionnels comme combinaisons linéaires de fonctions de base $X(t) = \sum_{j=1}^{D}\theta_j\varphi_j$.

Étape 1. Construction de la base. On utilise typiquement les fonctions de la forme `create.Nom_Base.basis` dont on a vu quelques éléments de syntaxe à la Section 2.3.2 précédente. La fonction `plot` peut s'appliquer directement aux objets créés. Il est possible aussi d'évaluer les fonctions de base et leurs dérivées en des points de discrétisation pour construire une matrice de type $(\varphi_j(t_k))_{\substack{1\leq j\leq D\\1\leq k\leq p}}$, à l'aide de la fonction `eval.basis` qui prend comme paramètres principaux

`evalarg` le vecteur de points en lesquels on évalue les fonctions ;

`basisobj` l'objet créé avec `create.Nom_Base.basis` ;

`Lfdobj` (dans le cas où l'on veut évaluer les dérivées des fonctions de base) un entier qui indique l'ordre éventuel de la dérivée concernée.

La fonction `predict` appelée avec les arguments `basisobj` et `evalarg` (dans cet ordre) donne le même résultat lorsqu'il s'agit d'évaluer les fonctions (et pas leurs dérivées). On peut tracer les différentes colonnes d'une matrice avec la fonction `matplot`.

Étape 2. Définition des coefficients. Il s'agit de définir les coefficients θ_j sous forme d'un vecteur de même longueur que le nombre de fonctions de base créées. On peut, dans le cas où l'on souhaite construire des fonctions aléatoires, utiliser par exemple des fonctions de simulation de variables aléatoires de lois données comme `rnorm, rpois, rexp`. La liste complète peut-être trouvée en tapant `help(distributions)`.

Étape 3. Création de l'objet fonctionnel. On appelle ensuite la fonction `fd` dont les deux principaux arguments sont les suivants :

`coeff` le vecteur des coefficients ;

`basisobj` l'objet créer avec `create.Nom_Base.basis`.

Étape 4. Premiers exemples d'utilisation de l'objet : évaluation et tracé. La fonction `plot` peut directement s'appliquer à l'objet fonctionnel, et permet de le représenter, ou de représenter ses dérivées (utilisation du paramètre `LfdObj` pour spécifier l'ordre). Par défaut, la fonction est évaluée en 201 points équirépartis. L'argument `nx` permet de modifier ceci (201 peut ne pas être suffisant). La fonction `eval.fd` fonctionne de manière similaire à `eval.basis` et permet d'évaluer les objets de la classe `fd` et leurs dérivées.

Cas de la création d'un échantillon ou/et de données multidimensionnelles. La méthode est la même, il faut juste simuler plus de coefficients, et les ordonner sous forme matricielle voulue (en utilisant `matrix` ou `array`). Par exemple, dans le cas d'un échantillon unidimensionnel de données $X_i(t) = \sum_{j=1}^{D} \theta_{j,i}\varphi_j(t)$, $i = 1, \ldots, n$, on peut créer le tableau des $(\theta_{i,j})_{\substack{1\leq j\leq D\\1\leq i\leq n}}$ comme une matrice à p lignes et n colonnes. Dans le cas d'un échantillon bidimensionnel $(X_i, Y_i) = (\sum_{j=1}^{D} \theta_{j,i}\varphi_j(t), \sum_{j=1}^{D} \theta'_{j,i}\varphi_j(t))$, on crée un tableau tridimensionnel des $(\theta_{i,j}, \theta'_{i,j})$.

Exercice 3.

1. Construire une base de splines cubiques avec 21 noeuds équirépartis dans l'intervalle $[0, 2]$. Faire différents essais pour les tracer toutes, ou seulement certaines, et tracer également pour certaines fonctions, leurs dérivées jusqu'à l'ordre 3.
2. Construire une courbe aléatoire X exprimée comme combinaison linéaire de la base précédente, avec pour coefficients des variables indépendantes de même loi gaussienne centrée réduite. Tracer l'objet obtenu, ainsi que sa dérivée.

3. Construire maintenant un échantillon de 10 courbes comme la précédente, simulées à partir de variables indépendantes. Les représenter d'abord toutes sur le même graphique, puis représenter sur un autre graphique les 2 premières courbes ainsi que leur somme.

Chapitre 3

Des données fonctionnelles aux fonctions (lisses)

3.1 Introduction : interpolation et lissage

Nous avons vu que toute analyse de données considérées comme fonctionnelles devait se donner comme première étape la reconstitution des données sous leur forme continue, fonctionnelle justement, à partir des données brutes qui sont toujours discrétisées. Supposons qu'un échantillon n de données fonctionnelles est collecté sous forme discrétisée, avec pour tout $i \in \{1, \ldots, n\}$, $y_i = {}^t(y_{i,1}, \ldots, y_{i,p_i})$. La première tâche du statisticien est donc de convertir ces mesures en une fonction x_i, dont les valeurs $x_i(t)$ peuvent être calculées pour toute valeur de t dans un certain ensemble.

On distingue deux cas :

1. les observations sont obtenues sans erreur (ou les erreurs de mesures sont négligeables) : dans ce cas, les égalités $y_{i,j} = x_i(t_{i,j})$, $j = 1, \ldots, p_i$ résument l'observation de la courbe x_i. Pour trouver x_i, on fait de l'*interpolation* des points de coordonnées $(t_{i,j_i}, y_{i,j})$.
2. les observations sont bruitées, ce que l'on modélise de la façon suivante,

$$y_{i,j} = x_i(t_{i,j}) + \varepsilon_{i,j}, \quad j = 1, \ldots, p_i, \quad i = 1, \ldots, n,$$

où les $\varepsilon_{i,j}$ sont des variables non observées centrées, admettant une variance (inconnue), représentant les erreurs de mesure : c'est un bruit, une perturbation qui contribue au caractère brut des données. Il faut alors faire du *lissage*, pour ôter l'erreur de mesure, c'est-à-dire prendre en compte le bruit qui se superpose au signal, de le filtrer.

On se concentrera sur le second cas, le plus fréquent, dans ce chapitre. On simplifie ici les notations, en supposant que l'on dispose d'une seule trajectoire : on part de $y = {}^t(y_1, \ldots, y_p) \in \mathbb{R}^p$ pour reconstruire une fonction x. La prise en compte du bruit se superposant au signal se traduit par le modèle de régression à design fixe suivant :

$$y_j = x(t_j) + \varepsilon_j, \quad j = 1, \ldots, p, \tag{3.1}$$

où les ε_j sont des variables non observées *i.i.d.* centrées, admettant une variance (inconnue). La fonction x, supposée exister (c'est le point de départ de l'analyse de données fonctionnelles !), est inconnue, on en cherche donc une approximation ou estimation que l'on notera $\widehat{x}$, qui ne dépend que de ce qui est connu, les $(y_j, t_j)_{1 \leq j \leq p}$.

Remarque. Dans le cas où l'on dispose non pas d'une seule trajectoire, mais d'un échantillon complet, plusieurs stratégies peuvent être adoptées : soit on estime chaque courbe pour chaque individu (mise en oeuvre possible uniquement si les observations sont suffisamment denses pour chaque sujet - ce qui signifie p_i grand pour tout i), soit on rassemble les observations de tous les sujets, et on choisit plutôt d'inférer des structures des populations (fonctions de moyenne, de covariance...). On considère uniquement la première possibilité ici.

On distingue plusieurs exigences pour la construction de $\widehat{x}$.

1. On veut tout d'abord naturellement que la courbe reconstruite se rapproche, en un certain sens, des données initiales. Le "sens" le plus communément donné à une certaine proximité entre observation et reconstruction est celui des moindres carrés : on veut rendre petite la somme des carrés des erreurs, c'est-à-dire la sommes des carrés des écarts entre observations et reconstruction :

$$\sum_{j=1}^{p}(y_j - x(t_j))^2.$$

   C'est le *lissage par moindres carrés*, voir Section 3.2.

2. On peut vouloir aussi imposer une certaine régularité à la courbe que l'on reconstruit, ce qui revient à supposer que $\widehat{x}$ appartient à un espace de fonctions régulières ("lisses"). C'est le cas lorsque l'on cherche à reconstruire non pas seulement x mais aussi ses dérivées successives. On verra que ceci est relativement contradictoire avec le fait de rendre le critère des moindres carrés petit, d'où la nécessité d'un compromis. Les techniques utilisées pour la recherche de fonctions régulières consistent souvent en la minimisation d'un critère faisant intervenir les normes L^2 des dérivées successives. On cherchera alors à minimiser

$$\sum_{j=1}^{p}(y_j - x(t_j))^2 + \lambda \int_T (x^{(k)}(t))^2 dt,$$

   pour un certain entier k et une certaine constante λ. On parlera de *lissage par moindres carrés pénalisés*, voir Section 3.3.

3. Les méthodes ci-dessus sont globales, au sens où le critère va être le même pour estimer x en tout point de T. On peut aussi vouloir qu'en un certain point $t \in T$, l'estimation tienne compte principalement des observations proche de t. On parle alors de méthodes *locales*.

3.2 Lissage par moindres carrés

3.2.1 Principe : moindres carrés ordinaires et pondérés

Définition 3.1 *On appelle* ***critère des moindres carrés*** *pour le problème de régression* (3.1)*, consistant à ajuster une courbe $\widehat{x}$ aux données $(t_j, y_j)_{j=1,\dots,p}$ la fonction de contraste suivante :*

$$Crit_{LS}(x) = \sum_{j=1}^{p}(y_j - x(t_j))^2.$$

Le principe de la méthode des *moindres carrés ordinaires* est le suivant : on cherche

$$\widehat{x} \in \arg\min_x \mathrm{Crit}_{LS}(x).$$

La question est de savoir sur quel espace de fonctions l'arg min est déterminé, c'est-à-dire sous quelle forme on cherche x (à quel espace de fonction appartient-elle ?). Tenant compte de ce qui précède, voir Section 2.3, la méthode la plus classique consiste à chercher $\hat{x}$ sous la forme d'une combinaison linéaire de fonctions de base données. On cherche donc le minimum du critère ci-dessus sur le sous-espace

$$S_D = \mathrm{Vect}\{\varphi_1, \ldots, \varphi_D\} \tag{3.2}$$

pour $(\varphi_k)_{k=1,\ldots,D}$ des fonctions linéairement indépendantes de $L^2(T)$. Cet espace est appelé *espace d'approximation*, ou encore *modèle* ("sieve"). Or,

$$\min_{x\in S_D} \mathrm{Crit}_{LS}(x) = \min_{x\in S_D} \sum_{j=1}^{p} (y_j - x(t_j))^2 = \min_{\theta\in\mathbb{R}^D} \sum_{j=1}^{p} \left(y_j - \sum_{k=1}^{D} \theta_k \varphi_k(t_j) \right)^2 := \min_{\theta\in\mathbb{R}^D} \mathrm{Crit}_{LS,\theta}(\theta).$$

Ainsi, la solution à ce problème $\widehat{x}$ aura la forme suivante :

$$\widehat{x}(t) = \sum_{k=1}^{D} \widehat{\theta}_k \varphi_k(t), \quad t \in T, \quad \text{avec } \widehat{\theta} = (\theta_k)_{k\in\{1,\ldots,D\}} \in \arg\min_{\theta\in\mathbb{R}^D} \mathrm{Crit}_{LS,\theta}(\theta).$$

En notant, comme au chapitre précédent, $\Phi(t) = {}^t(\varphi_1(t), \ldots, \varphi_D(t))$ le vecteur dont les éléments sont les D premiers éléments de la base évalués en t, on a la représentation matricielle suivante :

$$\widehat{x}(t) = {}^t\widehat{\theta}\Phi(t).$$

La solution à ce problème d'optimisation est la suivante.

Proposition 3.1 *Avec les notations ci-dessus, la solution $\widehat{\theta}$ au problème d'optimisation satisfait*

$$\widehat{\theta} = ({}^t\mathbf{\Phi}\mathbf{\Phi})^{-1}{}^t\mathbf{\Phi} y,$$

où $y = {}^t(y_1, \ldots, y_p)$ est le vecteur des observations et $\mathbf{\Phi}$ la matrice à p lignes et D colonnes définie par : $\mathbf{\Phi} = (\varphi_k(t_j))_{\substack{1\le j\le p,\\ 1\le k\le D}}$.

Le vecteur $\widehat{y} = {}^t(\widehat{y}_1, \ldots, \widehat{y}_p) = {}^t(\widehat{x}(t_1), \ldots, \widehat{x}(t_p))$ des valeurs ajustées est donc

$$\widehat{y} = \mathbf{\Phi}\widehat{\theta} = \mathbf{\Phi}({}^t\mathbf{\Phi}\mathbf{\Phi})^{-1}{}^t\mathbf{\Phi} y.$$

Preuve de la Proposition 3.1 On peut réécrire le critère Crit_{LS} de la façon matricielle suivante :

$$\mathrm{Crit}_{LS,\theta}(\theta) = {}^t(y - \mathbf{\Phi}\theta)(y - \mathbf{\Phi}\theta) = {}^t yy - 2{}^t(\mathbf{\Phi}\theta)y + {}^t(\mathbf{\Phi}\theta)(\mathbf{\Phi}\theta).$$

Ainsi, $\arg\min_{\theta\in\mathbb{R}^D} \mathrm{Crit}_{LS,\theta}(\theta) = \arg\min_{\theta\in\mathbb{R}^D} -2{}^t(\mathbf{\Phi}\theta)y + {}^t(\mathbf{\Phi}\theta)(\mathbf{\Phi}\theta)$, et le résultat est obtenu en différentiant cette fonction de D variables, en cherchant le point critique, et en vérifiant ensuite par calcul de la hessienne qu'il s'agit bien d'un minimum.

On peut prouver que la méthode des moindres carrés présente des propriétés d'optimalité, dans le cas où les erreurs ε_j, $j = 1, \dots, p$, du modèle de régression sous-jacent (3.1), sont centrées, indépendantes et distribuées selon une loi normale. Dans ce cas, la matrice de variance-covariance des observations $y = {}^t(y_1, \dots, y_p)$ est $\mathrm{Var}(y) = \mathrm{Var}(\varepsilon) = I$. Il se peut cependant que les variances varient en fonction des temps d'observation. On utilise alors généralement une alternative aux moindres carrés ordinaires, les *moindres carrés pondérés*. Si on introduit $w_1, \dots, w_p$ des poids positifs, le critère est

$$\widehat{x} \in \arg\min_x \sum_{j=1}^{p} w_j (y_j - x(t_j))^2.$$

Et, en définissant W comme la matrice diagonale de coefficients diagonaux les w_j, l'analogue de la Proposition 3.1 est

$$\widehat{\theta} = ({}^t\mathbf{\Phi} W \mathbf{\Phi})^{-1} {}^t\mathbf{\Phi} W y.$$

Si l'on suppose bien les erreurs décorrélées, et que l'on connait leurs variances, on peut choisir $W = \mathrm{Var}(y)$. Sinon, on tente généralement d'estimer cette matrice de variance covariance. La méthode fonctionne également avec des matrices W non diagonales (mais symétriques définies positives), ce qui peut permettre de prendre en compte des corrélations entre les erreurs.

3.2.2 Lissage par moindres carrés avec R

On décrit maintenant une manière de construire l'objet fonctionnel $\widehat{x}$ correspondant au lissage des données $(t_j, y_j)_{j=1,\dots,p}$ obtenu par moindres carrés non pondérés. On suppose que l'on a construit la matrice $\mathbf{\Phi}$ des fonctions de bases $(\varphi_j)_{j=1,\dots,D}$ évaluées en les $(t_k)_{k=1,\dots,p}$, par les méthodes décrites à la Section 2.3.3. Il s'agit donc juste de calculer le vecteur des coefficients $\widehat{\theta}$ dont l'expression est donnée à la Proposition 3.1.

Code R. Lissage par moindres carrés.

Ayant calculé la matrice $\mathbf{\Phi} = (\varphi_k(t_j))_{\substack{j=1,\dots,p \\ k=1,\dots,D}}$, et disposant des données $(y_j)_{j=1,\dots,p}$, on peut calculer $\widehat{\theta} = ({}^t\mathbf{\Phi}\mathbf{\Phi})^{-1} {}^t\mathbf{\Phi} y$, à l'aide des fonctions R suivantes.

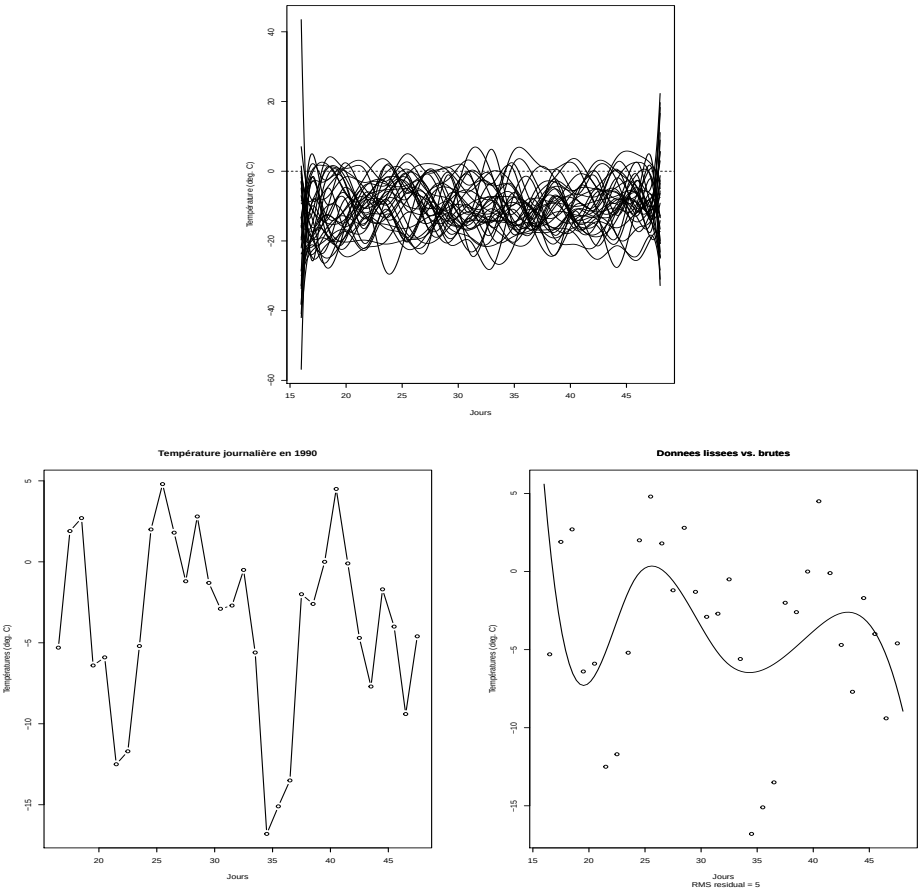
- `crossprod`. Avec pour seul argument à $\mathbf{\Phi}$, cette fonction renvoie $A = {}^t\mathbf{\Phi}\mathbf{\Phi}$. Avec pour arguments $\mathbf{\Phi}$ et y, elle renvoie $b = {}^t\mathbf{\Phi} y$.
- `solve`. Connaissant les deux objets A et b obtenus avec `crossprod`, le vecteur $\widehat{\theta}$ est obtenu par résolution du système $A\widehat{\theta} = b$, ce qui peut se faire en appelant la fonction `solve` avec les arguments A et b.

On peut aussi éventuellement utiliser la fonction `lsfit`, avec en arguments la matrice $\mathbf{\Phi}$ et le vecteur y.

Une fois que l'on dispose des coefficients $\widehat{\theta}$ et de la base $(\varphi_j)_j$, l'objet fonctionnel peut être construit avec la fonction `fd` (voir Section 2.3.3). On peut comparer les données brutes au lissage obtenu grâce à la fonction `plotfit.fd`.

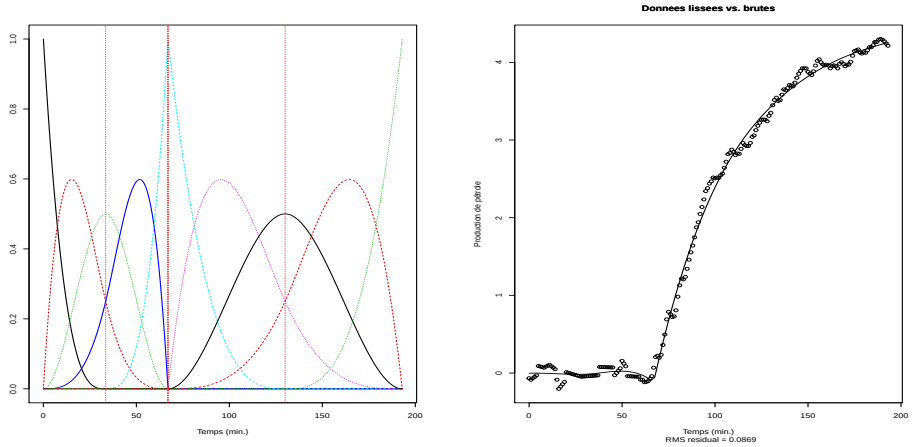
On représente par exemple à la Figure 3.1 le lissage par moindres carrés obtenu dans une base de 7 B-splines cubiques pour des courbes de températures journalières à Montréal (données `MontrealTemp` du package `fda`) du 15 janvier au 16 février (entre 1960 et 1994, une courbe par année). Les noeuds sont équirépartis et tous distincts.

La situation est un peu différente pour le lissage des données `refinery` que l'on avait représentées à la Figure 1.7 : en effet, une discontinuité des dérivées apparaît à la 67ème minute. On propose également un lissage par splines cubiques, mais on place plusieurs noeuds



(a) (b)

FIGURE 3.1 – Lissage par moindres carrés des données de températures moyennes journalières à Montréal (entre le 15 janvier et le 16 février), entre 1960 et 1994 (une courbe par année). 1ère ligne : les 34 courbes obtenues par lissage. 2nde ligne : le cas de l'année 1990 (a) Données brutes (b) Données brutes (représentées par des cercles) et données ajustées (ligne continue).



(a) (b)

FIGURE 3.2 – Lissage par moindres carrés des données de production de pétrole (voir Figure 1.7). (a) Base de B-splines utilisées (b) Données brutes (représentées par des cercles) et données ajustées (ligne continue).

à la 67ème minute, pour avoir une dérivée discontinue. La base de spline utilisée ainsi que les données ajustées sont représentées à la Figure 3.2. On représente également la courbe dérivée à la Figure 3.3.

Exercice 4. On considère les données `growth` du package `fda`, décrites à la Section 1.1.1, et tracées à la Figure 1.3. Proposer un code R permettant d'effectuer le lissage par moindres carrés des données de croissance des filles, dans une base de 12 B-splines d'ordre 6 sur l'intervalle $[1, 18]$. Tracer sur le même graphique les données brutes et la courbe lissée pour un individu de l'échantillon.

3.2.3 Choix de la dimension de l'espace d'approximation

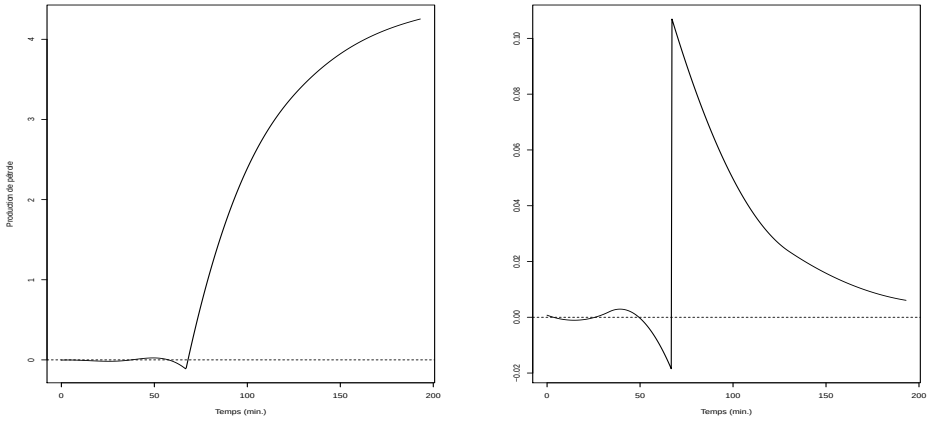
Les méthodes précédentes posent la question du choix, crucial, de la dimension D de l'espace S_D d'approximation, c'est-à-dire celui du nombre de coefficients $\widehat{\theta}_k$ estimés.

On peut constater en pratique la nécessité de faire un compromis entre de petites valeurs pour D et de grandes valeurs. Ceci peut aussi être démontré en théorie, en calculant un *risque* d'estimation de x par $\widehat{x}$ - risque quadratique intégré par exemple, et en cherchant à le rendre le plus petit possible. L'étude de ce risque montre qu'un *compromis biais-variance* est requis : le terme de biais du risque va décroitre quand D augmente, alors que le terme de variance va lui augmenter avec D. On retiendra le phénomène suivant.

(i) Si D est choisi trop petit, le biais de l'estimation va être trop grand. On a trop peu de flexibilité, les données sont sous-ajustées ("underfitting").

(ii) Inversement, si D est choisi grand, on a beaucoup de flexibilité mais ceci peut mener à un sur-ajustement des données ("overfitting").

Le problème du choix de D a été largement étudié dans la littérature, dans le cadre de problèmes non-paramétriques généraux (et pas nécessairement dans l'objectif de faire de l'étude de données fonctionnelles). Des méthodes variées ont été proposées : on citera, sans détails, celles de validation croisée, ou les méthodes dites de sélection de modèle par pénalisation de contraste (développées dans les années 2000 par Barron, Birgé et Massart,



(a) (b)

FIGURE 3.3 – Lissage par moindres carrés des données de production de pétrole (voir Figure 1.7). (a) Courbe obtenue par lissage dans une base de B-spline (b) Courbe dérivée de la fonction lissée.

voir par exemple Massart 2007).

Par ailleurs, il est utile de garder en mémoire qu'augmenter D ne se traduit pas toujours de la même façon, en fonction du type de fonctions de base choisie. Dans le cas de la base de Fourier, augmenter D revient simplement à ajouter des fonctions de base à l'espace choisi, $S_D \subset S_{D'}$, pour $D \leq D'$. Mais dans le cas d'une base de splines, c'est plus compliqué : augmenter l'ordre ou le nombre de noeuds ne revient pas à ajouter des fonctions à l'espace existant. Les espaces ne sont pas emboîtés.

Dans le cas du lissage de données fonctionnelles, pour parer ces inconvénients des moindres carrés ordinaires, la méthode la plus fréquemment utilisée pour effectuer de manière automatique le compromis biais-variance, en tenant compte de la régularité des courbes est l'utilisation de splines de lissage, que nous détaillons dans la section suivante.

3.3 Lissage par moindres carrés pénalisés : splines de lissage

3.3.1 Principe

L'approche par moindres carrés pénalisés vise à produire un estimateur $\widehat{x}$ pour une courbe x en minimisant un critère qui rend explicite deux objectifs contradictoires en estimation de courbe :

— on veut bien sûr que la courbe estimée donne un bon ajustement aux données,
— mais que l'ajustement ne soit pas trop précis, sous peine d'obtenir une courbe excessivement oscillante (sur-ajustement).

Il va donc s'agir de minimiser un critère des moindres carrés, mais sous la contrainte que la fonction cible appartient à un espace de fonctions "lisses". Il faut alors choisir un tel espace, parmi une large gamme d'espace de fonctions régulières (voir Section 2.1.1), et déterminer un critère permettant de mesurer la "rugosité" (roughness) d'une fonction. Le critère classiquement utilisé en analyse de données fonctionnelles est celui de la convergence

des intégrales suivantes :

$$\text{pen}_m(x) = \int_T \left(x^{(m)}(t)\right)^2 dt. \tag{3.3}$$

Ainsi, $x^{(1)} = x'$ mesure par exemple la pente de x à chaque instant, $x^{(2)} = x''$ mesure la courbure de x en chaque point. Prendre l'intégrale du carré des fonctions correspondantes revient à obtenir un mesure globale, souvent appelée *énergie*.

En tenant compte des objectifs contradictoires ci-dessus, on va donc cette fois chercher à minimiser le critère suivant.

Définition 3.2 *On appelle* ***critère des moindres carrés pénalisés, avec pénalité de lissage*** *pour le problème de régression* (3.1)*, consistant à ajuster une courbe x aux données $(t_j, y_j)_{j=1,\dots,p}$, le critère suivant*

$$Crit_{PenLS}(x) = \sum_{j=1}^{p} (y_j - x(t_j))^2 + \lambda \text{pen}_m(x),$$

où $\text{pen}_m(x) = \int_T \left(x^{(m)}(t)\right)^2 dt$, et où $\lambda > 0$ est un paramètre à calibrer, appelé ***paramètre de lissage*** *ou* ***de pénalité****.*

Le principe est donc de chercher $\widehat{x} \in \arg\min_x \text{Crit}_{PenLS}(x)$. Reste à savoir sur quel espace on minimise le critère. Celui-ci doit bien sûr contenir uniquement des fonctions m fois dérivables, de dérivées m-ièmes intégrables sur T. Il s'agit d'un problème dit *variationnel*. Le puissant résultat suivant est dû à Green et Silverman (1994) (on peut aussi le trouver dans de Boor 2001). Soit $\mathcal{S}^m(T)$ l'espace des fonctions m fois dérivables sur T, avec des dérivées $m-$ièmes absolument continues : si $x \in \mathcal{S}^m(T)$ il existe une fonction $g = x^{(m)}$, telle que $\int_a^t g(t)dt = x^{(m-1)}(t) - x^{(m-1)}(a)$, pour tout $t \in T = [a, b]$.

Théorème 3.1 *On pose*

$$\widehat{x} \in \arg \min_{x \in \mathcal{S}^m(T)} Crit_{PenLS}(x).$$

Alors $\widehat{x}$ est unique, et est appelée ***spline naturelle de lissage d'ordre*** *m associée aux données $(t_j, y_j)_{j=1,\dots,p}$. Elle est définie sur $T = [a, b]$ de la façon suivante :*

(i) la restriction de $\widehat{x}$ aux intervalles $[a, t_1]$ et $[t_p, b]$ est un polynôme de degré au plus $m - 1$;

(ii) la restriction de $\widehat{x}$ aux intervalles $[t_j, t_{j+1}]$, $j = 1, \dots, p - 1$, est un polynôme de degré au plus $2m - 1$;

(iii) la fonction $\widehat{x}$ est de classe $\mathcal{C}^{2m-2}(T)$.

En particulier, $\widehat{x}$ est une spline dont les noeuds sont les points t_j où il y a des données.

Notons que nous n'avons fait aucune hypothèse de structure sur $\widehat{x}$ pour énoncer le résultat : la structure de spline est une conséquence du théorème. Cela résout le problème du choix de la place des noeuds. En particulier, les splines de lissage s'adaptent automatiquement à des points de design non équirépartis.

Remarque. Il est également possible de considérer d'autres pénalités, permettant d'autres mesures de rugosité des fonctions à reconstruire. Par exemple, dans le cas de données périodiques, on pourra considérer

$$\text{pen}(x) = \int_T (Lx)^2(t)dt, \quad Lx = x^{(3)} + \omega^2 x', \tag{3.4}$$

où ω est la période et où l'opérateur L est appelé *opérateur d'accélération harmonique*. On a en effet $L\cos(\omega\cdot) = L\sin(\omega\cdot) = 0$.

Cas des splines naturelles cubiques. On se restreint dans la suite au cas $m = 2$, le plus utilisé en pratique. La spline naturelle cubique est donc définie par

$$\begin{array}{l}\widehat{x}(t) = a_j(t-t_j)^3 + b_j(t-t_j)^2 + c_j(t-t_j) + d_j, \quad t \in [t_j, t_{j+1}],\ j \in \{2,\dots,p-2\},\\ \widehat{x}(t) = e_j(t-t_j) + f_j \quad t \in [t_j, t_{j+1}],\ j \in \{1, p-1\},\end{array}$$

avec les contraintes $\widehat{x}_{|[t_{j-1},t_j]}(t_j) = \widehat{x}_{|[t_j,t_{j+1}]}(t_j)$, $\widehat{x}'_{|[t_{j-1},t_j]}(t_j) = \widehat{x}'_{|[t_j,t_{j+1}]}(t_j)$, et $\widehat{x}''_{|[t_{j-1},t_j]}(t_j) = \widehat{x}''_{|[t_j,t_{j+1}]}(t_j)$. Les dérivées d'ordre 2 et 3 en a et b sont nulles (spline naturelle, et cubique, les bords sont donc des segments de droites).

3.3.2 Existence et unicité de la spline minimisante : preuve dans un cas simple, et calcul par l'algorithme de Reinsch

On considère ici le cas des splines naturelles cubiques, et l'on va démontrer qu'il existe un unique minimum sur l'espace des splines cubiques naturelles, pour le critère Crit_{PenLS} introduit à la Définition 3.2 : c'est donc une version très simplifiée du Théorème 3.1 puisqu'on se restreint au cas $m = 2$ et que l'on calcule le minimum non pas sur tout $\mathcal{S}^m(T)$ mais sur l'espace des splines directement.

3.3.2.1 Première étape : représentation $g - \gamma$ d'une spline cubique

Nous allons voir qu'une spline cubique naturelle peut être entièrement spécifiée par ses valeurs et les valeurs de sa dérivées aux noeuds t_j. Le point de départ est le suivant :

Proposition 3.2 *Soit P une fonction polynôme de degré 3, considérée définie sur un intervalle $[s_1, s_2]$ de $\mathbb{R}$. Alors, P est entièrement déterminé sur $[s_1, s_2]$ par les deux vecteurs $g = (P(s_1), P(s_2))$ et $\gamma = (P''(s_1), P''(s_2))$.*

Soit maintenant x une spline cubique naturelle de noeuds les "abscisses" du design, $t_1 < \dots < t_p$. On introduit les deux vecteurs suivants

$$g = {}^t(x(t_1), \dots, x(t_p)) \in \mathbb{R}^p \text{ et } \gamma = {}^t(x''(t_2), \dots, x''(t_{p-1})) \in \mathbb{R}^{p-2}.$$

Rappelons que $x''(t_1) = x''(t_p) = 0$ (spline naturelle). La proposition précédente entraîne que la spline cubique x sur $[t_1, t_p]$ est entièrement déterminée par la données des vecteurs g et γ : leur donnée permet de la reconstruire intégralement, au sens où, connaissant g et γ, on peut calculer $x(t)$ quel que soit $t \in [t_1, t_p]$.

Mais la donnée de deux vecteurs arbitraires g et γ ne permettent pas toujours de définir une spline cubique naturelle : il faut qu'ils soient liés par une certaine relation, que l'on

détaille maintenant. La condition va dépendre de la donnée de deux matrices bandes, Q (à p lignes et $p-2$ colonnes) et R (à $p-2$ lignes et $p-2$ colonnes). Soit $h_j = t_{j+1} - t_j$, pour $j = 1, \ldots, p-1$.

On définit $Q = (q_{i,j})_{\substack{1 \leq i \leq p \\ 2 \leq j \leq p-1}}$ (attention à la numérotation des colonnes) par

$$\begin{cases} q_{j-1,j} = h_{j-1}^{-1}, \ q_{j,j} = -h_{j-1}^{-1} - h_j^{-1} \ \ q_{j+1,j} = h_j^{-1}, \ \ j \in \{2, \ldots, p-1\}, \\ q_{i,j} = 0 \ \ |i-j| \geq 2. \end{cases}$$

Ainsi

$$Q = \begin{pmatrix} h_1^{-1} & 0 & \cdots & \cdots & 0 \\ -h_1^{-1} - h_2^{-1} & h_2^{-1} & \ddots & \ddots & \vdots \\ h_2^{-1} & \ddots & \vdots & \ddots & \vdots \\ 0 & \ddots & -h_i^{-1} - h_{i+1}^{-1} & \ddots & 0 \\ \vdots & \ddots & h_{i+1}^{-1} & \ddots & h_{p-2}^{-1} \\ \vdots & \cdots & \ddots & \ddots & -h_{p-2}^{-1} - h_{p-1}^{-1} \\ 0 & \cdots & \cdots & 0 & h_{p-1}^{-1} \end{pmatrix}.$$

Puis on définit $R = (r_{i,j})_{\substack{1 \leq i \leq p-2 \\ 1 \leq j \leq p-2}}$ par

$$\begin{cases} r_{i,i} = (h_{i-1} + h_i)/3, \ \ i \in \{2, \ldots, p-1\}, \\ r_{i,i+1} = r_{i+1,i} = h_i/6 \ \ i \in \{2, \ldots, p-2\}, \\ r_{i,j} = 0 \ \ |i-j| \geq 2. \end{cases}$$

Ainsi

$$R = \begin{pmatrix} (h_1 + h_2)/3 & h_2/6 & 0 & \cdots & & 0 \\ h_2/6 & \ddots & \ddots & \ddots & & \vdots \\ 0 & \ddots & (h_{i-1} + h_i)/3 & h_i/6 & & \vdots \\ & \ddots & h_i/6 & \ddots & \ddots & 0 \\ \vdots & & & \ddots & \ddots & h_{p-2}/6 \\ 0 & \cdots & \cdots & 0 & h_{p-2}/6 & (h_{p-2} + h_{p-1})/3 \end{pmatrix}.$$

La matrice R est symétrique, à diagonale dominante (au sens où $r_{i,i} > \sum_{i \neq j} |r_{i,j}|$), donc définie positive, donc inversible. Le théorème suivant est dû à Green et Silverman (1994).

Théorème 3.2 *La donnée de deux vecteurs $g \in \mathbb{R}^p$ et $\gamma \in \mathbb{R}^{p-2}$ définit une spline cubique naturelle x si la relation suivante est vérifiée*

$$^tQg = R\gamma.$$

Si cette condition est satisfaite,

$$\int_{t_1}^{t_p} x''(t)dt = {}^t\gamma R\gamma = {}^tgKg, \text{ où } K = QR^{-1}{}^tQ.$$

3.3.2.2 Deuxième étape : existence et unicité de la spline minimisante

On étudie maintenant l'existence de l'estimateur de lissage par spline obtenu par minimisation du critère Crit_{PenLS} introduit à la Définition 3.2. Par le Théorème 3.2, la connaissance du vecteur g entraîne celle de γ et donc, par la Proposition 3.2 celle de la spline minimisante x. On réécrit donc le problème en fonction de g,

$$\begin{aligned}\text{Crit}_{PenLS}(x) &= {}^t(y-g)(y-g) + \lambda {}^t g K g = {}^t yy + {}^t gg - 2{}^t gy\lambda {}^t gKg, \\ &= {}^t yy - 2{}^t gy + {}^t g(I+\lambda K)g := \text{Crit}_{PenLS,g}(g).\end{aligned}$$

Comme λK est symétrique et positive, la matrice $I+\lambda K$ est définie positive. En différentiant par rapport à g le critère $\text{Crit}_{PenLS,g}$ (même méthode que pour prouver la Proposition 3.1), on montre donc que cette fonction a un unique minimum atteint en

$$g = (I+\lambda K)^{-1} y. \tag{3.5}$$

La matrice $A(\lambda) = (I+\lambda K)^{-1}$ est appelée *matrice chapeau*.

3.3.2.3 Calcul pratique de g

En pratique, un algorithme dû à Reinsch (1967) permet de calculer le vecteur g avec une complexité extrêmement raisonnable en tirant partie des structures bande des matrices considérées. Le fondement de l'algorithme consiste à calculer d'abord γ, avant d'en déduire g (voir Proposition 3.2). Le point de départ est le suivant : partant de (3.5), on a $(I+\lambda K)g = y$, ce qui se réécrit $g = y - \lambda Kg$. Tenant compte de la relation entre g et γ (Proposition 3.2), on a $g = y - \lambda Q\gamma$, et aussi ${}^t Qg = {}^t Qy - \lambda {}^t QQ\gamma$. Ceci se réécrit $R\gamma = {}^t Qy - \lambda {}^t QQ\gamma$, toujours par la Proposition 3.2, et donc

$$(R + \lambda {}^t QQ)\gamma = {}^t Qy \tag{3.6}$$

Cette relation est plus facilement exploitable que (3.5) car $R + \lambda {}^t QQ$ est symétrique définie positive et de structure bande de largeur de bande 5 (Q et R ont elle-même une structure bande, de largeur de bande 3). Elle admet donc une décomposition de Cholesky "simple" : $R + \lambda {}^t QQ = LD{}^t L$, avec D diagonale à éléments diagonaux strictement positifs, L triangulaire inférieure avec $L_{ii} = 1$ pour tout $i = 1, \ldots, p-2$, $L_{i,j} = 0$ pour $j < i-2$ et $j > i$.

L'algorithme est donc le suivant :

Étape 1 Calcul de ${}^t Qy$ d'une part et calcul de $R + \lambda {}^t QQ$ et de sa décomposition de Cholesky d'autre part.

Étape 2 Résolution en γ le système linéaire $LD{}^t L\gamma = {}^t Qy$ donné par (3.6).

Étape 3 Calcul de $g = y - \lambda Q\gamma$.

3.3.3 Choix du paramètre de lissage

On se place toujours dans le cas $m = 2$ ici. Le paramètre de lissage λ apparaissant devant la pénalité dans le critère Crit_{PenLS} de la Définition 3.2 peut être vu comme le multiplicateur de Lagrange apparaissant dans tout problème d'optimisation sous contrainte (théorème des extrema liés)- ici minimisation du contraste des moindres carrés sous contrainte de régularité. Par conséquent, il est inévitable.

En pratique, la calibration doit tenir compte des remarques suivantes.

— Si λ est petit, le poids de la pénalité dans la minimisation du critère est peu important, on privilégie l'ajustement aux données. Dans le cas extrême où $\lambda = 0$, minimiser le critère sur $\mathcal{S}^m(T)$ revient à interpoler les données, sans tenir compte de la régularité de la courbe.
— Si λ est grand, le caractère lisse de la fonction choisie sera privilégié par rapport à la fidélité aux données. Dans le cas extrême où $\lambda = \infty$, la solution au problème de minimisation sera la droite de régression des y_j sur les t_j.

La méthode la plus souvent utilisée est la *validation croisée leave one-out*. Pour tout $j = 1, \ldots, p$, on retire l'observation (t_j, y_j) de l'échantillon de données, et on calcule la solution au problème de minimisation défini à partir des autres données :

$$\widehat{x}_\lambda^{(-j)} \in \arg \min_{x \in \mathcal{S}^2(T)} \sum_{j' \neq j} (y_{j'} - x(t_{j'}))^2 + \lambda \int_T (x''(t))^2 dt.$$

La spline obtenue $\widehat{x}_\lambda^{(-j)}$ dépend du paramètre λ. On voit alors (t_j, y_j) comme une nouvelle observation, et on considère l'erreur de prévision : qualité de $\widehat{x}_\lambda^{(-j)}(t_j)$ pour estimer y_j, et ceci pour tout indice j. Ceci nous conduit au critère suivant.

Définition 3.3 *On appelle* ***score de validation croisée*** *pour le problème de régression à partir des données $(t_j, y_j)_{j=1,\ldots,p}$ le critère suivant, avec les notations introduites ci-dessus, et pour tout $\lambda \geq 0$.*

$$CV(\lambda) = \frac{1}{p} \sum_{j=1}^{p} \left(y_j - \widehat{x}_\lambda^{(-j)}(t_j) \right)^2.$$

On souhaite bien sûr minimiser ce critère. En pratique, on le fait sur une grille de valeurs pour λ : on a donc à calculer, p splines minimisantes à partir d'échantillons de taille $p-1$ et ce pour chaque λ dans la grille. Le calcul est en fait moins couteux qu'annoncé, puisque le score de validation croisée s'exprime en fait de manière plus simple, en utilisant les coefficients diagonaux $a_{j,j}(\lambda)$ de la matrice chapeau $A(\lambda) = (I + \lambda K)^{-1}$ intervenant dans (3.5).

Théorème 3.3 *Avec les notations ci-dessus, le score de validation croisée peut s'écrire*

$$CV(\lambda) = \frac{1}{p} \sum_{j=1}^{p} \left(\frac{y_j - \widehat{x}_\lambda(t_j)}{1 - a_{j,j}(\lambda)} \right)^2, \quad \lambda \geq 0$$

où $\widehat{x}_\lambda$ est la spline calculée avec l'échantillon complet et paramètre de pénalité λ.

Le calcul des $a_{j,j}(\lambda)$ nécessite *a priori* l'inversion d'une matrice de grande taille, au vu de la définition de la matrice chapeau $A(\lambda)$, mais il existe également des algorithmes permettant le calcul à un coût raisonnable. Par ailleurs, il existe des versions améliorées du critère précédent. En particulier, on verra que R utilise un critère plus général, la validation croisée généralisée.

3.3.4 Lissage par moindres carrés pénalisés avec R

Comme dans la sous-section 3.2.2, on décrit maintenant une manière de construire l'objet fonctionnel $\widehat{x}$ correspondant au lissage des données $(t_j, y_j)_{j=1,\ldots,p}$ obtenu par moindres carrés pénalisés.

Ce qui est programmé dans le package `fda` ne permet pas (bien sûr) de trouver la spline naturelle qui minimise le critère Crit_{PenLS} sur l'espace de dimension infinie $\mathcal{S}^m(T)$ (voir Théorème 3.1). Par conséquent, une fois le choix d'une dimension D et de fonctions linéairement indépendantes $\varphi_1, \ldots, \varphi_D$ (qui peuvent être une base de splines, mais aussi toutes autres fonctions de bases classiques : Fourier, ...). Les fonctions R vont permettre de minimiser

$$\theta \in \mathbb{R}^D \longmapsto \text{Crit}_{PenLS}\left(\sum_{j=1}^{D} \theta_j \varphi_j\right).$$

On suppose que l'on a construit la matrice $\mathbf{\Phi}$ des fonctions de bases $(\varphi_j)_{j=1,\ldots,D}$ évaluées en les $(t_k)_{k=1,\ldots,p}$, par les méthodes décrites à la Section 2.3.3.

Code R. Lissage par moindres carrés pénalisés.

On utilise la fonction `smooth.basis` du package `fda`, qui minimise, sur un espace de fonctions donné un critère des moindres carrés pénalisés. Elle retourne un objet de type `fdsmooth`, comportant plusieurs éléments. En particulier, si on affecte le résultat de `smooth.basis` à une variable `Donnees_Lissees`, l'objet `Donnees_Lissees$fd` est l'objet fonctionnel cherché. Les arguments à fournir à `smooth.basis`, quelle que soit l'utilisation que l'on souhaite en faire, sont les suivants.

`argvals` les valeurs des $(t_j)_{j=1,\ldots,p}$;

`y` les données $(y_j)_{j=1,\ldots,p}$.

Ci-dessous, en fonction de ce que l'on veut ou non spécifier, on précise d'autres arguments et utilisation.

1. **Calcul du lissage (création de l'objet fonctionnel) sans spécifier la pénalité**. Par défaut, `smooth.basis` n'utilise aucune pénalité, si on lui fournit un objet de la classe `basisfd`. Il suffit alors de l'appeler avec les deux paramètres précédents (`argvals` et `y`), ainsi qu'une base de fonctions (typiquement, le résultat d'un appel à `create.Nom_base.basis`). Le résultat est analogue à un lissage des moindres carrés ordinaire.
2. **Calcul du lissage avec spécification de la pénalité**. Il est possible de préciser d'une part le paramètre λ intervenant dans la pénalité, d'autre part l'opérateur utilisé pour la définir. On fait alors précéder l'appel à `smooth.basis` d'un appel à la fonction `fdPar`, qui renvoie des paramètres fonctionnels, utilisés par d'autres fonctions ensuite et dont les paramètres (qu'on utilisera) sont les suivants :

   `fdobj` typiquement, la base de fonctions créée à l'aide de `create.Nom_base.basis` ;

   `Lfdoj` l'ordre m de la dérivée à utiliser si on choisit comme pénalité pen_m définie en (3.3) ou alors un opérateur différentiel (défini avec `vec2Lfd` par exemple) ;

   `lambda` le paramètre λ devant la pénalité.

   Puis, on appelle `smooth.basis` avec comme arguments remplaçant la base de fonction, le résultat de l'appel à `fdPar`. On utilisera cette méthode dans les deux cas suivants.
3. **Calcul du lissage avec choix optimal du paramètre λ de la pénalité** au sens de la validation croisée généralisée (gcv). La fonction `smooth.basis` renvoie l'objet fonctionnel (obtenu avec `$fd`) mais aussi la valeur du critère gcv à minimiser pour choisir le paramètre λ optimal. On peut donc se donner une grille de λ possible, appeler pour chacun les fonctions `fdPar` et `smooth.basis`, stocker la valeur du critère

gcv moyen pour chaque valeur de la grille, et enfin, choisir celle qui le minimise, avant d'effectuer le lissage final.

Fonction `vec2Lfd` **:** permet de définir un opérateur différentiel de la forme

$$Lx(t) = \beta_0 x(t) + \beta_1 x'(t) + \cdots + \beta_{m-1} x^{(m-1)}(t) + x^{(m)}(t)$$

Elle prend en argument le vecteur des coefficients $(\beta_0, \ldots, \beta_{m-1})$ et `rangeval` les extrémités de l'intervalle de définition de l'opérateur. Par exemple, pour l'opérateur harmonique intervenant dans (3.4), le premier argument est `c(0,(2*pi/`ω`)^2,0)`.

Quelques exemples de lissages de données décrites à la Section 1.1.1.

1. **Données** `handwrit`, tracées à la Figure 1.6. On propose un premier lissage de ces données dans une base de B-splines cubiques, avec 21 noeuds. La précision n'étant pas tout à fait suffisante, on change ensuite la base, en augmentant le nombre de noeuds ainsi que l'ordre. On obtient ainsi un lissage plus adéquate, voir Figure 3.4. Une fois les objets fonctionnels liés aux données lissées obtenus, on peut calculer et représenter les courbes dérivées, ce qui est fait à la Figure 3.5.
2. **Données** `Refinery,` tracées à la Figure 1.7. Un lissage de ces données avait été proposé à la section précédente 3.2.2, en base de spline, et nous avions discuté l'intérêt de placer plusieurs noeuds au point de discontinuité (67-ième minute). On utilise maintenant un lissage par moindres carrés pénalisés en base de B-splines avec plusieurs noeuds au temps 67, et on observe l'influence du paramètre λ, comme décrit ci-dessus. Si λ est trop grand, c'est le caractère "lisse" de la courbe qui est privilégié au détriment de l'ajustement aux données, voir Figure 3.6.
3. **Données** `Canadian Weather, températures et précipitations,` voir Figure 1.1. On propose un lissage en base de Fourier de période 365 avec pénalité définie par l'opérateur d'accélération harmonique (3.4). Le paramètre de lissage est obtenu en optimisant le critère gcv, voir Figures 3.7 et 3.8.
4. **Données** `Growth`, tracées à la Figure 1.3. Le lissage représenté au graphique (b) de la Figure 3.9 est effectué en base de $B-$splines, mais la pénalité est cette fois pen_4 : on veut contraindre la régularité de la courbe de la dérivée seconde, représentant l'accélération de la croissance. Le paramètre λ est choisi à l'aide du critère gcv, tracé au graphique (a).

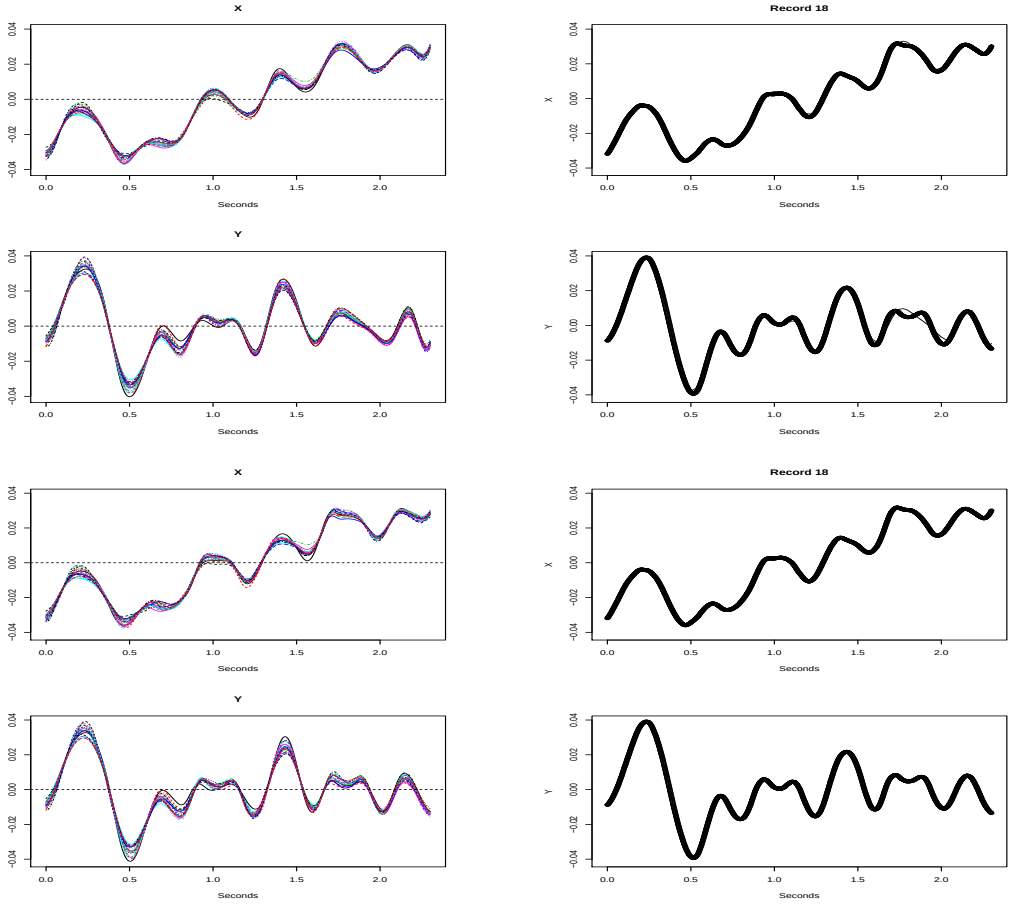
Exercice 5.

1. Générer un vecteur t de 101 points équirépartis entre 0 et 2. Définir, pour tout point de t, les valeurs de la fonction x suivante

$$x(t) = \begin{cases} (t-0.5)^2 \text{ pour } t \leq 1, \\ 0.25(t-1) \text{ pour } t > 1 \end{cases}$$

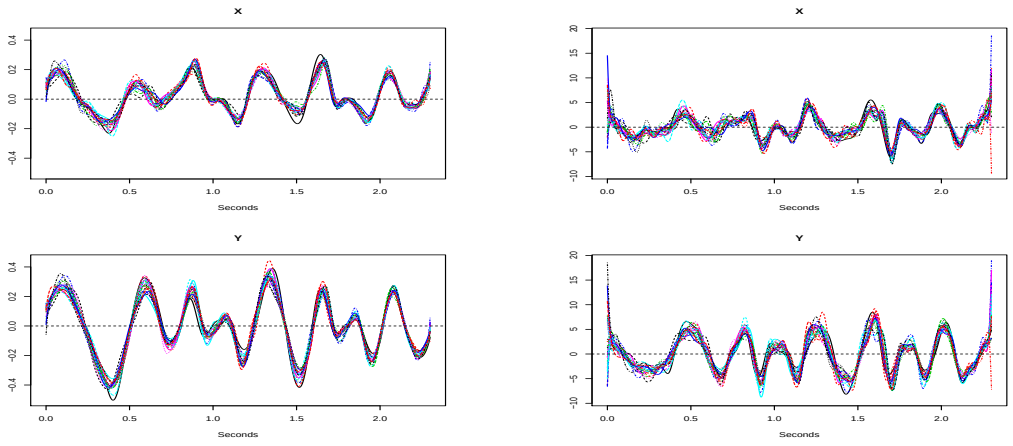
   Tracer le résultat obtenu.

2. Créer une base de B-splines cubiques avec points de rupture 0, 0.5,1, 1.5 et 2, pour effectuer le lissage par moindres carrés de ces observations. Comment prendre en compte la discontinuité en $t = 1$?



(a) (b)

FIGURE 3.4 – Lissage par moindres carrés non pénalisé des données `handwrit`. Première ligne : base de B-splines cubiques, avec 21 noeuds (23 fonctions de base) ; seconde ligne : base de B-splines d'ordre 6 avec 51 noeuds (55 fonctions de base). (a) Les 20 courbes lissées (abscisses et ordonnées enregistrées à différents temps). (b) Enregistrement 18 : comparaison des données brutes et des données lissées.



(a) (b)

FIGURE 3.5 – Courbes dérivées des données `handwrit` lissées en base de B-splines d'ordre 6 avec 51 noeuds et pénalité pen_2 (a) dérivées premières (b) dérivées secondes.

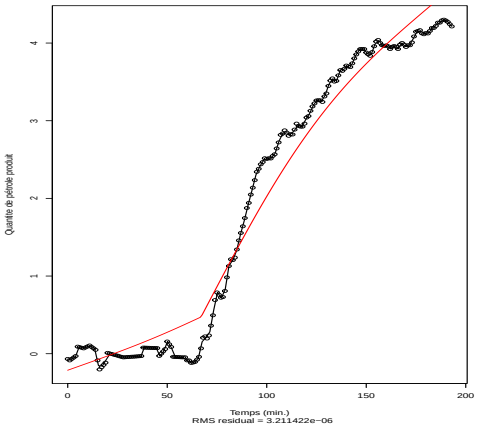
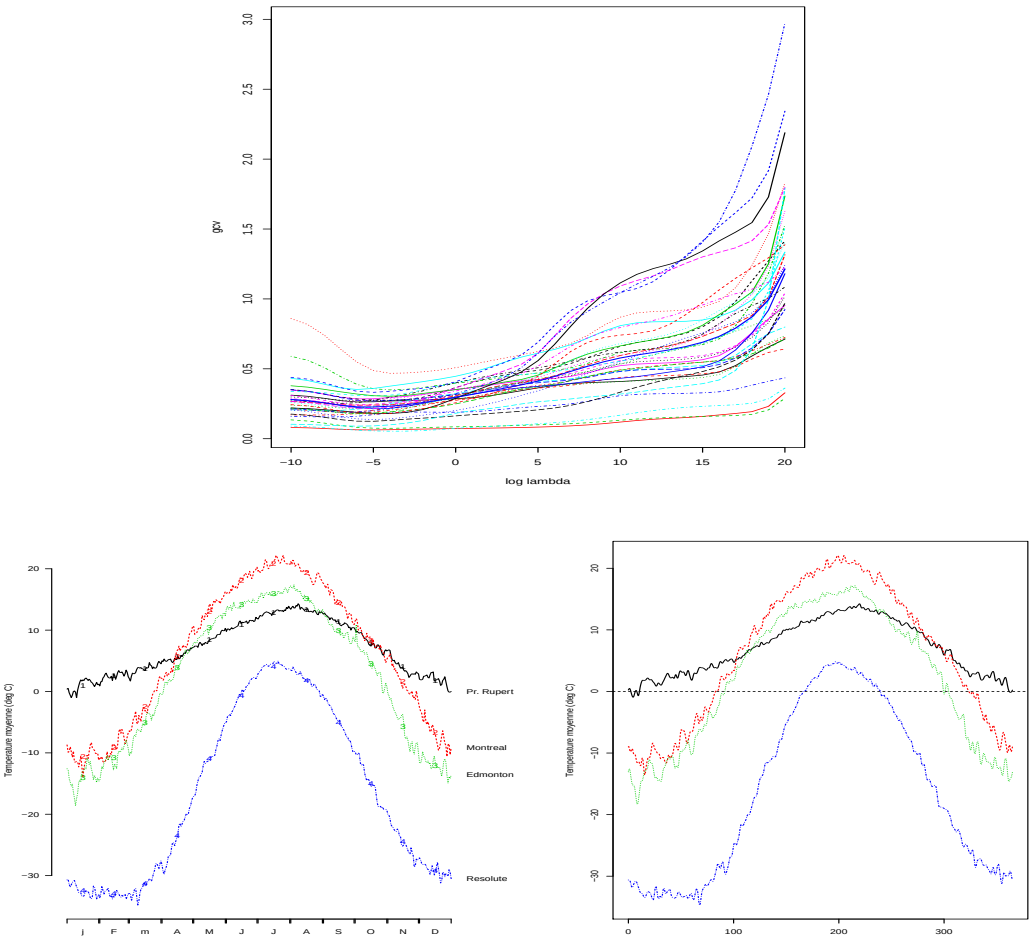
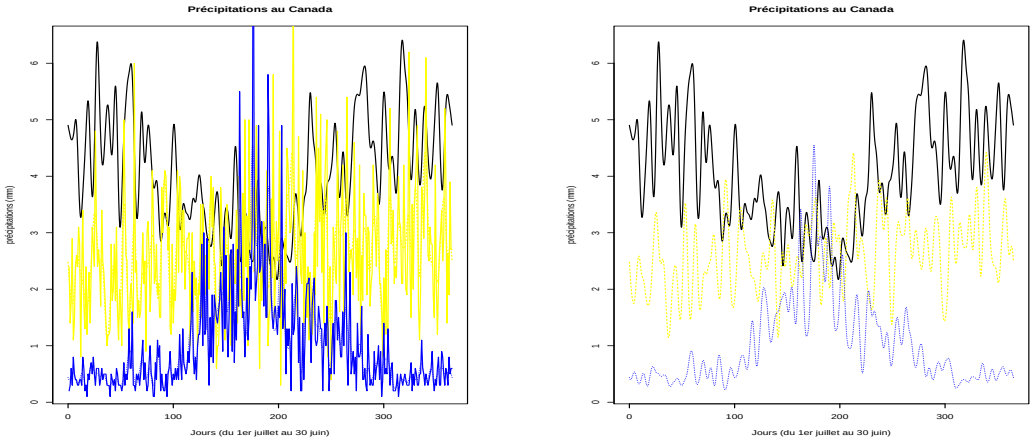


FIGURE 3.6 – Deux lissages des données `refinery` (production de pétrole) en base de B-splines cubiques avec noeuds aux points d'observation (et un noeud supplémentaire en la discontinuité) et pénalité pen_2, en fonction de la valeur du paramètre λ de pénalité. Courbe noire : obtenue avec $\lambda = 10^{-5}$, Courbe rouge : obtenue avec $\lambda = 10^{6}$.



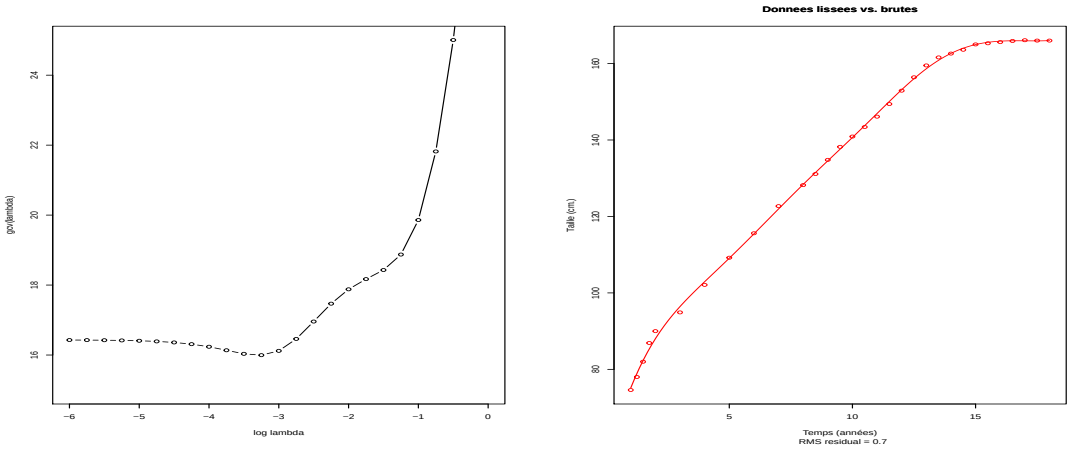
(a) (b)

FIGURE 3.7 – Données de températures journalières dans 5 stations météorologiques canadiennes. Première ligne : Critère de validation croisée généralisée (gcv) en fonction du paramètre de pénalité λ. Seconde ligne : (a) Données brutes. (b) Données lissées en base de Fourier (période : 365 jours), avec 365 fonctions de bases, et pénalité définie avec l'opérateur harmonique (3.4), et paramètre λ sélectionné par gcv.



(a) (b)

FIGURE 3.8 – Données de précipitation journalière dans 3 stations météorologiques canadiennes : Regina (bleu), St Johns (noir), London (jaune). (a) Données brutes. (b) Données lissées en base de Fourier (période : 365 jours), avec 365 fonctions de bases, et pénalité définie avec l'opérateur harmonique (3.4), et paramètre λ sélectionné par validation croisée généralisée (gcv).



(a) (b)

FIGURE 3.9 – Données de croissance de filles. (a) Somme du critère de validation croisée généralisée (gcv) en fonction de λ pour toutes les filles de l'échantillon, pour une base de $B-$splines, d'ordre 6 avec noeuds à chaque année (entière), et pénalité pen_4. (b) Données lissées dans la base précédente avec paramètre λ sélectionné par validation croisée généralisée, pour l'un des individus de l'échantillon.

3. À l'aide de la fonction `smooth.basis`, créer un objet fonctionnel `ylisse` avec les données de la question 1. et la base créée à la question 2. Améliorer les résultats obtenus en enlevant l'observation correspondant à $t = 1$.
4. Évaluer, à l'aide de la fonction `eval.fd` l'objet fonctionnel `ylisse` en 200 points pour créer un vecteur `yeval`. Comparer les résultats donnés par les calculs suivants :
   (i) `2*ylisse+4` et `2*yeval+4` ;
   (ii) `ylisse2` et `yeval2` ;
   (iii) `sqrt(ylisse)` et `sqrt(yeval)` ;

Exercice 6. On considère les données `melanoma` du package `fda` décrites à la Section 1.1.1 et tracées à la Figure 1.8.

1. Pour i allant de 1 à 35, créer une base de Fourier à i fonctions de base sur l'intervalle de temps des données, effectuer le lissage des données à l'aide de la fonction `smooth.basis`, et récupérer la valeur du critère gcv correspondante.
2. Choisir le nombre de fonctions de base minimisant le critère gcv, et superposer sur un même graphique le lissage correspondant et les données brutes.

Chapitre 4

Statistique descriptive et exploratoire pour données fonctionnelles

Introduction

Il est question dans ce chapitre d'explorer des données fonctionnelles, sans inférence sur une population plus large : il s'agit de décrire ce qui est disponible, de tirer les caractéristiques principales d'un échantillon de données. On s'intéresse dans une première section aux mesures résumées de statistique descriptive spécifiques à l'analyse de données fonctionnelles, puis à l'extension des techniques d'Analyse en Composantes Principales (ACP) au cas fonctionnel.

4.1 Éléments de statistique descriptive pour données fonctionnelles

Toute analyse de données débute par des éléments basiques : l'estimation de la moyenne et des déviations. Nous donnons dans ce paragraphe des versions fonctionnelles pour ces mesures résumées. Le cadre est le suivant : nous supposons observer n réalisations de variables aléatoires fonctionnelles $X_1, \ldots, X_n$, *i.i.d.*, telles que $X_i \in L^2(T)$, $i \in \{1, \ldots, n\}$.

4.1.1 Moyenne et variance

Définition 4.1 *La* ***fonction moyenne empirique*** *de l'ensemble des réalisations* $\{X_1, \ldots, X_n\}$ *de variables de* $L^2(T)$ *est l'application suivante*

$$\begin{array}{rccl} \bar{X}_n : T & \longrightarrow & \mathbb{R}, \\ t & \longmapsto & \bar{X}_n(t) = \dfrac{1}{n}\displaystyle\sum_{i=1}^{n} X_i(t). \end{array}$$

Définition 4.2 *La **fonction variance empirique** de l'ensemble des réalisations $\{X_1, \ldots, X_n\}$ de variables de $L^2(T)$ est l'application suivante*

$$\begin{aligned} Var_n : T &\longrightarrow \mathbb{R}, \\ t &\longmapsto Var_n(t) = \frac{1}{n}\sum_{i=1}^{n}\left(X_i(t) - \bar{X}_n(t)\right)^2. \end{aligned}$$

On définit également l'écart-type comme étant la racine carrée de cette fonction variance.

Code R. Calcul des fonctions moyenne et variance. Le package `fda` contient des fonctions permettant de calculer $\bar{X}_n$ et $\sqrt{\mathrm{Var}_n}$.

1. Fonction `mean.fd` : appliquée à un objet fonctionnel comportant plusieurs courbes (éventuellement multidimensionnel) de la classe `fd`, elle renvoie un objet de la classe `fd` contenant la (les) fonction(s) moyenne de l'objet (ou des objets) en paramètres.
2. Fonction `sd.fd` ou `std.fd` : de manière similaire à la fonction précédente, retourne les fonctions écart-type des données en paramètres.

Les courbes se représentent avec `plot`, comme toutes les données fonctionnelles étudiées jusqu'ici.

À titre d'exemple, on simule un échantillon de 10 courbes aléatoires comme combinaisons linéaires des fonctions d'une base de splines cubiques avec 21 noeuds équirépartis dans l'intervalle $[0, 2]$, avec pour coefficients des variables indépendantes de même loi gaussienne centrée réduite (résultat de l'***Exercice 3***, question **3.** du Chapitre 2). Les courbes sont représentées à la Figure 4.1, et on leur superpose la courbe représentative de leur fonction moyenne $t \mapsto \bar{X}_n(t)$, et aussi les courbes des fonctions $t \mapsto \bar{X}_n(t) - 2\sqrt{\mathrm{Var}_n(t)}$, et $t \mapsto \bar{X}_n(t) + 2\sqrt{\mathrm{Var}_n(t)}$. Attention, l'écart-type est calculé pour la collection de courbes et on n'a donc pas un intervalle de confiance pour chacune des courbes simulées.

On peut faire les mêmes calculs par exemple pour les données fonctionnelles bivariées `Handwrit`, décrites au Chapitre 2 et représentées à la Figure 1.6. On obtient une courbe de moyenne pour les abscisses d'écriture, un écart-type pour ces données, et la même chose pour les ordonnées. Certains résultats sont représentés à la Figure 4.2.

4.1.2 Covariance et corrélation

La fonction de covariance empirique, définie ci-dessous, synthétise la dépendance des enregistrements en différentes valeurs de T.

Définition 4.3 *La **fonction de covariance empirique** de l'ensemble des réalisations $\{X_1, \ldots, X_n\}$ de variables de $L^2(T)$ est l'application suivante*

$$\begin{aligned} C_n : T^2 &\longrightarrow \mathbb{R}, \\ (t_1, t_2) &\longmapsto C_n(t_1, t_2) = \frac{1}{n}\sum_{i=1}^{n}\left(X_i(t_1) - \bar{X}(t_1)\right)\left(X_i(t_2) - \bar{X}(t_2)\right). \end{aligned}$$

*La **fonction de corrélation** associée est*

$$\begin{aligned} Corr_n : T^2 &\longrightarrow \mathbb{R}, \\ (t_1, t_2) &\longmapsto Corr_n(t_1, t_2) = \frac{C_n(t_1, t_2)}{\sqrt{Var_n(t_1)\, Var_n(t_2)}}. \end{aligned}$$

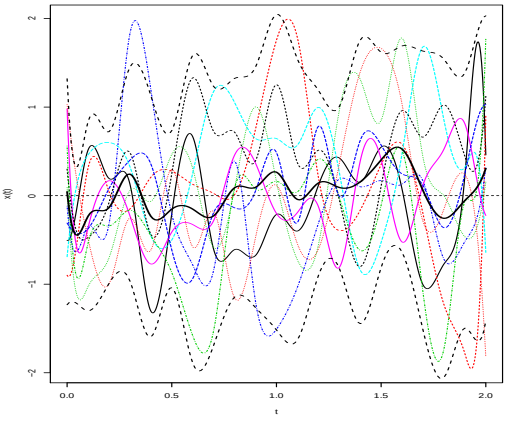


FIGURE 4.1 – 10 courbes simulées comme combinaisons linéaires des fonctions d'une base de splines cubiques à 21 noeuds équirépartis dans l'intervalle $[0, 2]$, avec pour coefficients des variables indépendantes de même loi $\mathcal{N}(0, 1)$ (lignes fines) avec leur fonction moyenne $t \mapsto \bar{X}_n(t)$ (ligne pleine, en gras) et les fonctions $t \mapsto \bar{X}_n(t) - 2\sqrt{\mathrm{Var}_n(t)}$, et $t \mapsto \bar{X}_n(t) + 2\sqrt{\mathrm{Var}_n(t)}$ (lignes pointillées, en gras).

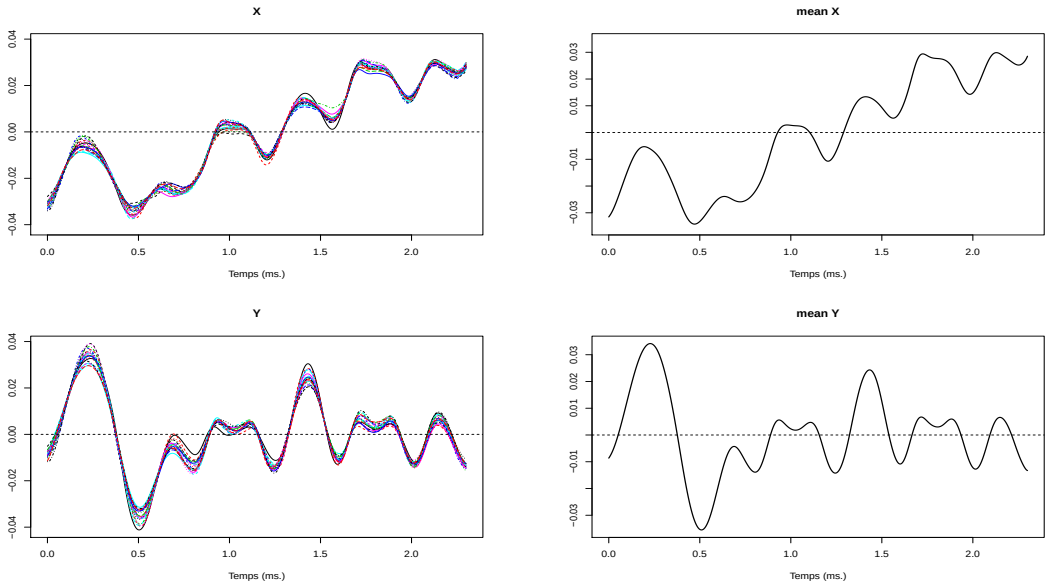
*L'**opérateur de covariance empirique** associé à l'échantillon $(X_1, \ldots, X_n)$ supposé centré est défini par*

$$\begin{array}{rccl} \Gamma_n : L^2(T) & \longrightarrow & L^2(T) \\ f & \longmapsto & \dfrac{1}{n}\displaystyle\sum_{i=1}^{n} \langle f, X_i \rangle X_i. \end{array}$$

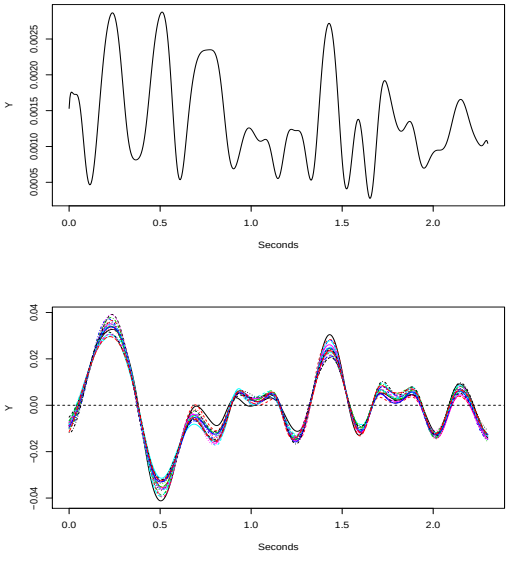
Le calcul de la fonction de covariance empirique avec R est implémenté dans le package `fda`, la description est reportée à la section suivante, puisque la fonction à utiliser sert également pour la covariance croisée.

4.1.3 Covariance et corrélation croisées

Habituellement, la covariance et la corrélation sont évaluées entre 2 variables. Les équivalents pour les variables fonctionnelles ne sont donc pas ceux de la Définition 4.3, mais plutôt les notions suivantes. On suppose ici que l'on observe des couples de variables aléatoires fonctionnelles $(X_1, Y_1), \ldots, (X_n, Y_n)$, *i.i.d.*, tels que $(X_i, Y_i) \in (L^2(T))^2$.



(a) (b)



(c)

FIGURE 4.2 – (a) Données `Handwrit` : abscisses (ligne 1) et ordonnées (ligne 2) en fonction du temps. (b) Fonctions moyennes des données `Handwrit`, abscisses (ligne 1) et ordonnées (ligne 2). (c) Fonction écart-type des données `Handwrit` ordonnées (ligne 1) et données associées (ligne 2).

Définition 4.4 *La **fonction de covariance croisée empirique** de l'ensemble des réalisations $\{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ de variables de $(L^2(T))^2$ est l'application suivante*

$$\begin{aligned} Cov_{n,X,Y} : T^2 &\longrightarrow \mathbb{R}, \\ (t_1, t_2) &\longmapsto Cov_{n,X,Y}(t_1, t_2) = \frac{1}{n}\sum_{i=1}^{n}\left(X_i(t_1) - \bar{X}(t_1)\right)\left(Y_i(t_2) - \bar{Y}(t_2)\right). \end{aligned}$$

*La **fonction de corrélation croisée** associée est*

$$\begin{aligned} Corr_{n,X,Y} : T^2 &\longrightarrow \mathbb{R}, \\ (t_1, t_2) &\longmapsto Corr_{n,X,Y}(t_1, t_2) = \frac{Cov_{n,X,Y}(t_1, t_2)}{\sqrt{Var_X(t_1)\, Var_Y(t_2)}}. \end{aligned}$$

Code R. Calcul de la fonction de covariance empirique, de la covariance et de la corrélation croisée Le package `fda` contient des fonctions permettant ces calculs, à partir d'échantillons de données fonctionnelles de la classe `fd`.

1. Fonction `var.fd` : selon l'utilisation, cette fonction retourne un ou plusieurs objets dont la classe est `bifd`, objets fonctionnels à deux variables (2 indices de temps par exemple).
   - **Premier cas :** appliquée à un objet fonctionnel univarié, ie. un échantillon de courbes $t \mapsto X_i(t)$, $i = 1, \ldots, n$, cette fonction renvoie $(t_1, t_2) \mapsto C_n(t_1, t_2)$ sous forme d'un objet fonctionnel qui doit ensuite être évalué en une grille de points pour pouvoir être tracé.
   - **Second cas :** appliquée à un objet bivarié, ie. un échantillon de courbes $t \mapsto (X_i(t), Y_i(t))$, $i = 1, \ldots, n$, cette fonction renvoie la fonction de covariance des X_i, $(t_1, t_2) \mapsto C_{n,X}(t_1, t_2)$, la covariance croisée des (X_i, Y_i) $(t_1, t_2) \mapsto \mathrm{Cov}_{n,X,Y}(t_1, t_2)$ et la fonction de covariance des Y_i, $(t_1, t_2) \mapsto C_{n,X}(t_1, t_2)$.
2. Fonction `cor.fd` : fonction calculant la corrélation empirique entre un ou deux objets fonctionnels, sous forme d'une matrice.
   - **Premier cas :** appliquée à une grille $\{t_j,\, j = 1, \ldots, d\}$ (vecteur), et un objet fonctionnel univarié, ie. un échantillon de courbes $t \mapsto X_i(t)$, $i = 1, \ldots, n$ (objet de la classe `fd`), cette fonction renvoie la matrice $C = (\mathrm{Corr}_{n,X}(t_j, t_k))_{1 \le j,k \le d}$.
   - **Second cas :** appliquée à une grille $\{t_j,\, j = 1, \ldots, d\}$ (vecteur), un premier échantillon de courbes $t \mapsto X_i(t)$, $i = 1, \ldots, n$, une seconde grille $\{t'_k,\, k = 1, \ldots, d'\}$ et un second échantillon de courbes $t \mapsto Y_i(t)$, $i = 1, \ldots, n$, cette fonction renvoie la matrice $C = (\mathrm{Corr}_{n,X,Y}(t_j, t'_k))_{1 \le j \le d, 1 \le k \le d'}$.

Code R. Représentation de courbes en 3D On donne ici quelques fonctions permettant de représenter une courbe $(t_1, t_2) \mapsto z(t_1, t_2)$.

1. Fonction `eval.bifd` du package `fda` : prend en arguments une première grille $\{t_j,\, j = 1, \ldots, d\}$ (vecteur), une seconde grille $\{t'_k,\, k = 1, \ldots, d'\}$, un (ou plusieurs) objet(s) fonctionnel(s) de la classe `bifd` et retourne un tableau contenant les évaluations de ces objets aux points (t_j, t'_k).
2. Outils de tracé d'une surface $(t_1, t_2) \mapsto z(t_1, t_2)$, à partir d'une grille de discrétisation pour t_1, une pour t_2, et la matrice d'évaluation de z en (t_1, t_2).

— Fonction `persp` : tracé 3D de la surface
— Fonction `contour` : tracé plan des lignes de niveau.
— Fonction `filled.contour` : tracé plan des lignes de niveau colorées (la couleur dépendant du niveau).
— Fonction `levelplot` du package `lattice` : tracé plan des lignes de niveau et couleurs entre elles.

Nous donnons maintenant différents exemples d'utilisation des fonctions ci-dessus, sur des données par ailleurs déjà utilisées dans ce document : la fonction de covariance empirique des données de précipitations canadiennes (logarithme des précipitations relevées en millimètres) est représentée à la figure 4.3. Pour les températures et précipitations du même jeu de données, dont un extrait a déjà été représenté à la Figure 1.1, on représente également la covariance et la corrélation, à la Figure 4.4. Quant aux données `Handwrit`, la Figure 4.5 leur est consacrée.

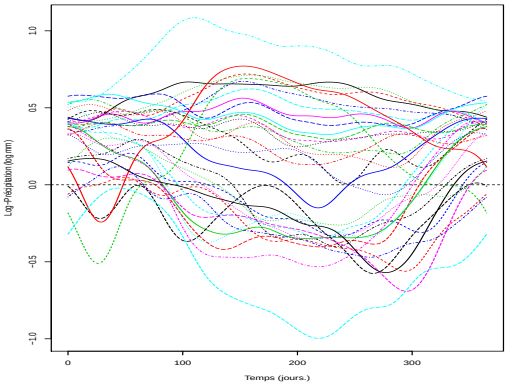
4.2 Analyse en composantes principales fonctionnelle

4.2.1 Introduction - rappel sur l'ACP multivariée.

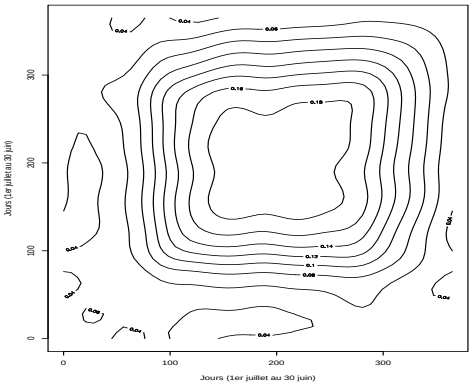
L'Analyse en Composantes Principales (ACP dans la suite) est une méthode de statistique exploratoire visant à produire une synthèse visuelle de données multivariées (observations de plusieurs variables pour plusieurs individus) : elle fait plus précisément partie des méthodes dites factorielles, dont le but est d'exploiter les aspects géométriques et représentations graphiques pour obtenir le "meilleur" résumé possible des données. En particulier, ses résultats sont de nature purement descriptive : en statistique multivariée, elle ne repose sur aucun modèle probabiliste, mais uniquement sur des considérations géométriques.

L'objectif est classiquement le suivant : il s'agit de représenter les variables étudiées dans un espace de dimension réduite par rapport à celle de l'espace de départ, tout en conservant le maximum d'information possible portée par la structure de ces variables. Supposons que l'on s'intéresse à la loi d'un vecteur aléatoire $X = {}^t(X_1, \ldots, X_d) \in \mathbb{R}^d$. Dès que $d > 3$, la représentation de X dans un repère devient impossible. On va alors chercher les sous-espaces de $\mathbb{R}^d$ qui résument le mieux l'information. La théorie prouve que cela revient à diagonaliser en base orthonormée la matrice de covariance Σ de X, et à choisir les sous-espaces engendrés par les vecteurs propres. Comme cette matrice est généralement inconnue, on s'appuie sur une version empirique, puisque l'on dispose généralement d'un échantillon d'observations supposées suivre la loi de X, et l'on considère la réduction de la matrice de covariance empirique associée. Cette présentation, à partir d'un échantillon *i.i.d.* est la plus classique, et nous résumons les grandes lignes ci-dessous.

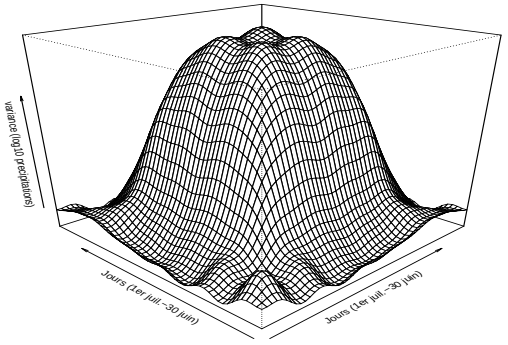
Supposons qu'on dispose d'observations $X_i = {}^t(X_{i,1}, \ldots, X_{i,d}) \in \mathbb{R}^d$, pour $i \in \{1, \ldots, n\}$. La mesure $X_{i,j}$ représente la valeur de la $j-$ième variable pour le $i-$ième individu. L'ensemble



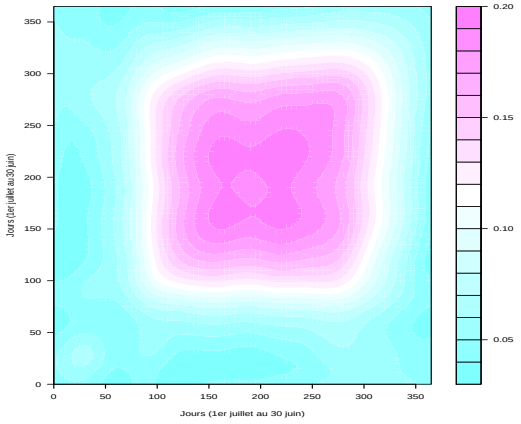
(a)



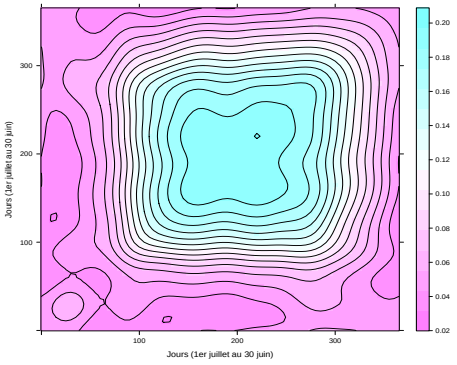
(b)



(c)

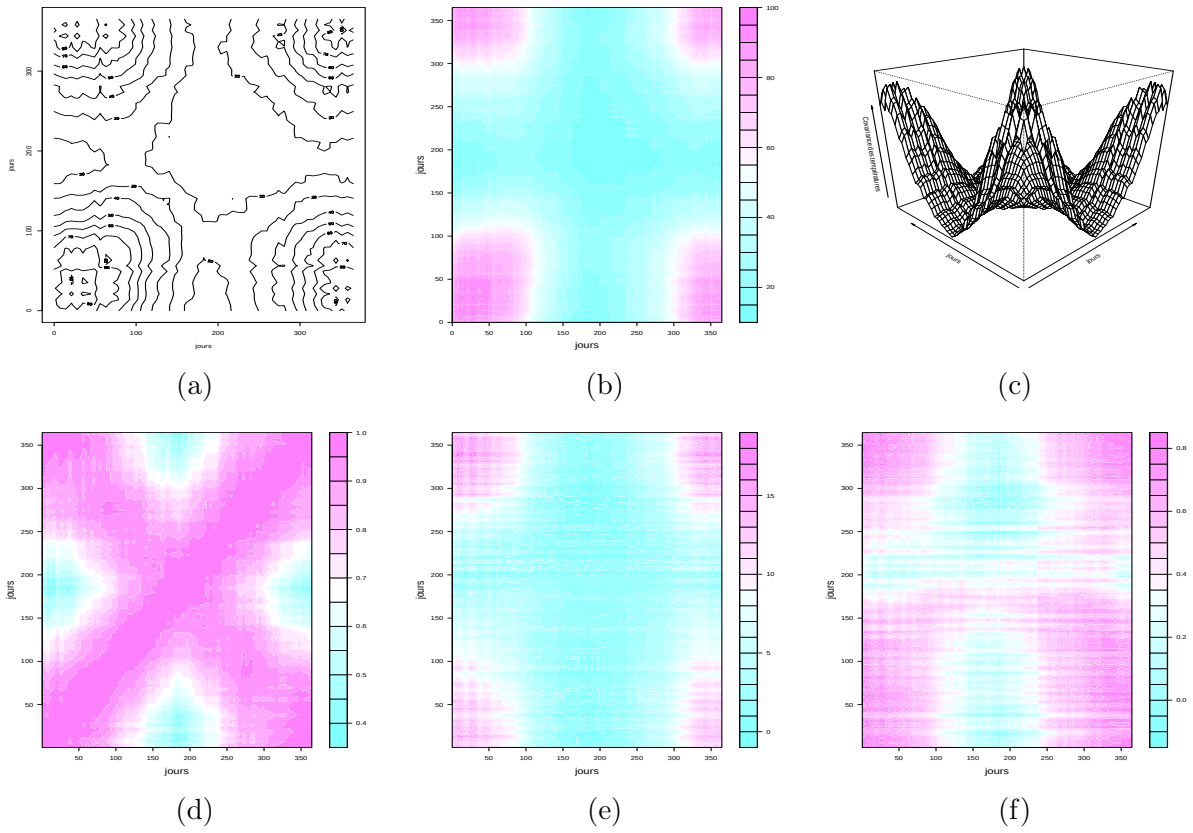


(d)



(e)

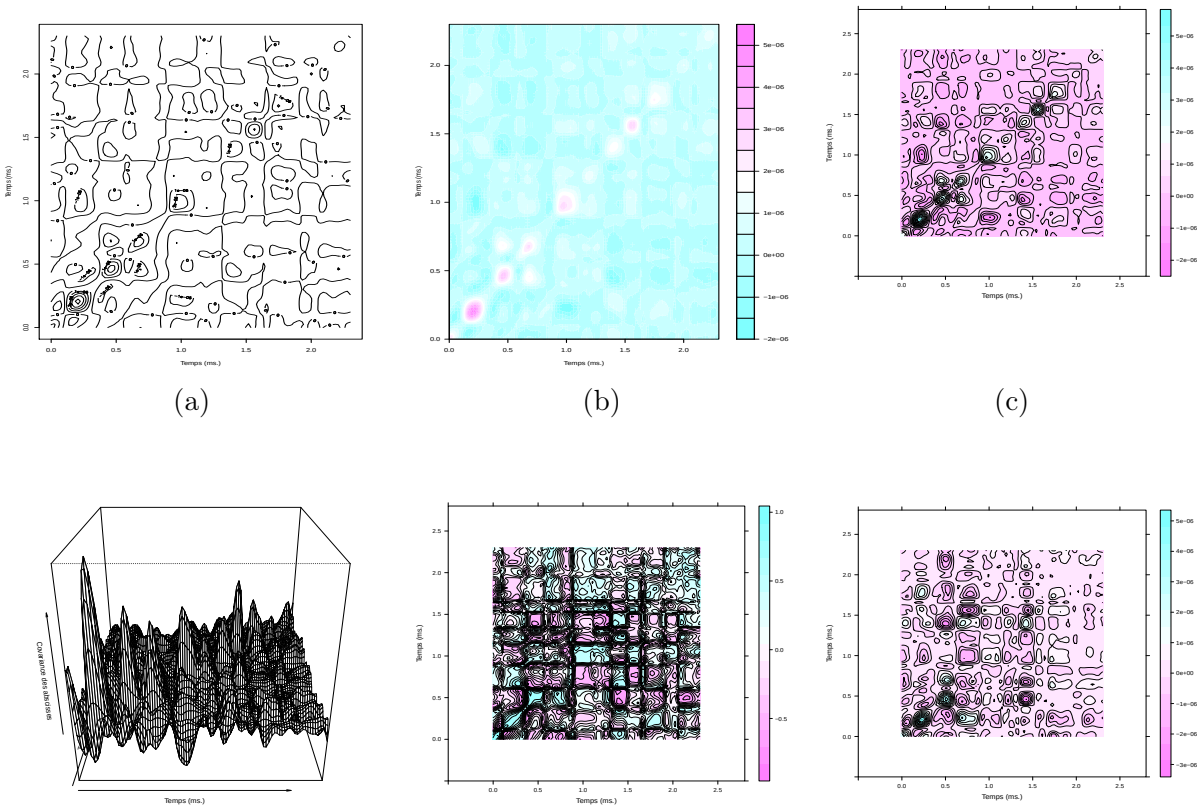
FIGURE 4.3 – (a) Données de log-précipitations `Canadian Weather`, lissées par moindres carrés pénalisé par pénalité impliquant l'opérateur d'accélération harmonique en base de Fourier de période 365 (une courbe par station météo). (b) (d) et (e) Lignes de niveau de la fonction de covariance empirique des données : (b) fonction `contour` de R, (d) fonction `filled.contour` (e) fonction `levelplot`. (c) Fonction de covariance empirique des données en 3D (fonction `persp` de R).



(a) (b) (c)

(d) (e) (f)

FIGURE 4.4 – Données `Canadian Weather`, de températures (en degrés) et de précipitations (en mm.) journalières dans des stations météorologiques canadiennes. Première ligne : Représentation de la fonction de covariance empirique des données de température (a) Fonction `contour` (b) Fonction `filledcontour` (c) Fonction `persp`. Seconde ligne (fonction `filledcontour`) : (d) Fonction de corrélation des données de températures. (e) et (f) Fonctions de covariance (graphique (e)) et de corrélation (graphique (f)) croisées des données de températures et précipitations.



(a) (b) (c)

(d) (e) (f)

FIGURE 4.5 – Données `Handwrit`, d'abscisses et d'ordonnées au cours du temps lors de 20 tracés d'écriture. Première ligne et début seconde : Représentation de la fonction de covariance empirique des données d'abscisses (a) Fonction `contour` (b) Fonction `filledcontour` (c) Fonction `level plot` (d) Fonction `persp`. Fin seconde ligne : (e) Fonction de corrélation des données d'abscisses (fonction `levelplot`). (f) Fonctions de covariance croisées des données d'abscisses et d'ordonnées.

des données est synthétisé par la matrice suivante, à n lignes et d colonnes

$$X = \begin{pmatrix} X_{1,1} & X_{1,2} & \cdots & X_{1,d} \\ X_{2,1} & X_{2,2} & \cdots & X_{2,d} \\ & \ddots & & \\ \vdots & & \ddots & \vdots \\ & & & \ddots \\ X_{n,1} & \cdots & & X_{n,d} \end{pmatrix},$$

avec $V_j = {}^t(X_{1,j}, \ldots, X_{n,j}) \in \mathbb{R}^n$ le vecteur de la $j-$ième variable (représenté par un point dans un espace affine de $\mathbb{R}^n$), et $U_i = {}^t(X_{i,1}, \ldots, X_{i,d}) \in \mathbb{R}^d$ le vecteur des variables du $i-$ième individu (représenté par un point dans un espace affine de $\mathbb{R}^d$). Sachant que dès que d ou n est supérieur à 3, la représentation de ces vecteurs sous forme de nuages de points devient impossible, l'idée est de trouver un système d'axes et de plans tels que les projections des nuages sur ces axes et ces plans permettent d'avoir une bonne image des données, avec la déformation la plus petite possible. Pour choisir l'origine de l'espace, on considère généralement le centre de gravité G du nuage des individus (avec poids $1/n$), de coordonnées ${}^t(X_{.1}, \ldots, X_{.d})$ avec $X_{.j} = \sum_{i=1}^n X_{i,j}/n$. Ceci revient à travailler sur les données centrées :

$$X_c = \begin{pmatrix} X_{1,1} - X_{.1} & & \cdots & X_{1,d} - X_{.d} \\ X_{2,1} - X_{.1} & X_{2,2} - X_{.2} & \cdots & X_{2,d} - X_{.d} \\ & \ddots & & \\ \vdots & & \ddots & \vdots \\ & & & \ddots \\ X_{n,1} - X_{.1} & \cdots & & X_{n,d} - X_{.d} \end{pmatrix}. \tag{4.1}$$

La matrice de covariance (empirique) des données est $\Sigma = (\mathrm{Cov}(V_j, V_{j'}))_{1 \leq j,j' \leq d}$, avec

$$\mathrm{Cov}(V_j, V_{j'}) = \frac{1}{n} \sum_{i=1}^n (X_{i,j} - X_{.j})(X_{i,j'} - X_{.j'}),$$

et elle peut s'exprimer sous la forme $\Sigma = {}^t X_c X_c / n$. Enfin, on mesure la dispersion du nuage des individus par l'inertie du nuage,

$$I = \frac{1}{n} \sum_{i=1}^n d^2(G, U_i) = \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^n (X_{i,j} - X_{.j})^2 = \sum_{j=1}^n \mathrm{Var}(V_j) = Tr(\Sigma),$$

d^2 désignant le carré de la distance euclidienne usuelle. La déformation du nuage des individus, projeté sur un sous espace S est mesurée par l'inertie du nuage autour du sous espace S,

$$I_S = \frac{1}{n} \sum_{i=1}^n d^2(\Pi_S U_i, U_i),$$

où $\Pi_S U_i$ est le projeté orthogonal de U_i sur S. On peut décomposer $I = I_S + I_{S^\perp}$, où $S^\perp$ est l'orthogonal de S dans l'espace affine considéré et

$$I_{S^\perp} = \frac{1}{n} \sum_{i=1}^n \|\Pi_S U_i\|^2.$$

Si S est une droite (un axe), $I_{S^\perp}$ mesure l'allongement du nuage selon cet axe, c'est l'inertie expliquée par S.

Le principe de l'ACP peut se résumer ainsi. On cherche un sous-espace S permettant d'obtenir, par projection, la moindre déformation du nuage des individus de départ. Il s'agit donc de trouver S qui minimise l'inertie I_S ou de façon équivalente maximise l'inertie du nuage projetée $I_{S^\perp}$:

$$\arg \min_{\dim S=k} I_S = \arg \max_{\dim S=k} I_{S^\perp}.$$

L'inertie du nuage projeté se décomposant comme somme des inerties moyennes du nuage projeté sur des droites orthogonales, la recherche de S se fait "séquentiellement", par récurrence (finie).

1. On cherche d'abord Δ_1, axe dirigé par un vecteur a_1 normé (de norme 1, ${}^t a_1 a_1 = 1$), tel que $I_{\Delta_1^\perp}$ est maximal. On peut montrer, en exprimant $I_{\Delta_1^\perp} = {}^t a_1 \Sigma a_1$, que a_1 doit être un vecteur propre unitaire associé à λ_1 la plus grande valeur propre de Σ.
2. On cherche ensuite Δ_2, orthogonal à Δ_1, dirigé par a_2 normé et d'inertie minimale...

On construit ainsi, par récurrence, des axes

$$\Delta_1 \perp \Delta_2 \perp \ldots \perp \Delta_d$$

dirigés par les vecteurs propres $a_1, \ldots, a_d$ associé aux valeurs propres

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$$

de Σ et tels que

$$I_{\Delta_1^\perp} \geq I_{\Delta_2^\perp} \geq \cdots \geq I_{\Delta_d^\perp} .$$

On a ainsi une suite de sous espace $S_k = \text{Vect}\{a_1, \ldots, a_k\}$, avec a_k vecteur propre unitaire associé à λ_k, $k-$ième plus grande valeur propre de Σ, et $I_{S_k^\perp} = \sum_{l=1}^k \lambda_l$. Comme Σ est diagonalisable, les nouveaux axes forment une base de l'espace, et sont appelés axes principaux. On représente les individus dans cette nouvelle base. La matrice Σ s'écrit comme $\Sigma = PD^tP$, où D est la matrice diagonale dont les éléments diagonaux sont les valeurs propres λ et P la matrice de passage de l'ancienne base à la nouvelle base, dont les colonnes sont les coordonnées des vecteurs propres a_j. On représente les individus U_i dans cette nouvelle base, sous la forme $U_i = \mu + \sum_{j=1}^k \xi_j a_j$, où μ est la moyenne empirique des observations.

L'ACP fonctionnelle va constituer, comme son nom l'indique, une extension de la méthode précédente au cas de données fonctionnelles. Son premier intérêt sera, comme en statistique multivariée de fournir un outil pour visualiser la répartition des données, mais, et c'est là l'une des forces de l'approche, elle sera également très utilisée pour réduire la dimension des données, par exemple dans le cas du modèle linéaire fonctionnel. Les premiers travaux sur l'analyse de données fonctionnelles concernent d'ailleurs l'ACP, comme on l'a déjà noté à la Section 1.3 : ce sont ceux de Deville (1974); Dauxois et Pousse (1976); Dauxois *et al.* (1982); Besse et Ramsay (1986) par exemple, et la question a largement été approfondie depuis, voir par exemple Hall (2011).

4.2.2 Théorie de l'ACP fonctionnelle d'une courbe aléatoire

Supposons que l'on s'intéresse à une courbe aléatoire X à valeurs dans $L^2(T)$, admettant un opérateur de covariance Γ_X. Pour comprendre comment passer de l'ACP multivariée au cas fonctionnel, mettons en parallèle ce dont on dispose, c'est l'objectif de la Table 4.1.

	ACP multivariée	ACP fonctionnelle
Données	$X \in \mathbb{R}^d$ $X = {}^t(X_1, \ldots, X_d)$	$X \in L^2(T)$ $X = \{X(t), t \in T\}$
Moyenne	vecteur de moyenne $\mathbb{E}[X] = {}^t(\mathbb{E}[X_1], \ldots, \mathbb{E}[X_d])$	courbe de la moyenne $\mathbb{E}[X] = \{\mathbb{E}[X(t)], t \in T\}$
Covariance	matrice $\Sigma_X = \mathrm{Cov}(X, X)$	fonction de covariance (surface) C_X ou opérateur Γ_X $C_X(s,t) = \mathrm{Cov}(X(s), X(t))$

Table 4.1 – Cas fonctionnel vs. multivarié

Comme pour l'ACP multivariée, on va chercher à construire un sous-espace vectoriel S de $L^2(T)$ de dimension finie, et capturant le plus d'information sur la variable X. La construction se fait, comme en multivarié, de manière séquentielle : on cherche une base hilbertienne $(\psi_j)_{j\in\mathbb{N}\setminus\{0\}}$ de $L^2(T)$ de telle sorte que l'espace $S_k = \mathrm{Vect}\{\psi_1, \ldots, \psi_k\}$ minimise en moyenne la distance L^2 entre X et sa projection orthogonale sur S_k, notée, $\Pi_{S_k} X = \sum_{j=1}^k \langle X, \psi_j\rangle \psi_j$. La construction se fait par récurrence.

Étape 1. On cherche d'abord $\psi_1 \in L^2(T)$, de norme $\|\psi_1\|^2 = \int_T \psi_1^2(t)dt = 1$, telle que

$$\psi_1 \in \arg\min_{f\in L^2(T)} \mathbb{E}[\|X - \langle X, f\rangle f\|].$$

Étape 2. On cherche ensuite $\psi_2 \in L^2(T)$, de norme $\|\psi_2\|^2 = 1$ et $\langle \psi_2, \psi_1\rangle = 0$, telle que

$$\psi_2 \in \arg\min_{f\in L^2(T)} \mathbb{E}[\|X - \langle X, \psi_1\rangle\psi_1 - \langle X, f\rangle f\|].$$

Étape 3. $\cdots$

Étape k+1. Si on a construit $\psi_1, \ldots, \psi_k$, on cherche $\psi_{k+1} \in L^2(T)$, de norme $\|\psi_{k+1}\|^2 = 1$ et $\langle \psi_{k+1}, \psi_j\rangle = 0$ pour tout $j \leq k$, telle que

$$\psi_{k+1} \in \arg\min_{f\in L^2(T)} \mathbb{E}[\|X - \Pi_{S_k} X - \langle X, f\rangle f\|].$$

De manière similaire au cas multivarié, on obtient le résultat suivant :

Proposition 4.1 *Avec les notations précédentes, la famille $(\psi_k)_k$ construite est au plus dénombrable et est constituée de fonctions propres de l'opérateur de covariance Γ_X associées aux valeurs propres $(\lambda_k)_{k\geq 1}$ rangées par ordre décroissant. Ainsi, la base de l'ACP fonctionnelle est, au signe près, la base de Karhunen-Loève de X.*

	ACP multivariée	ACP fonctionnelle
Données	$X \in \mathbb{R}^d$ $X = {}^t(X_1, \ldots, X_d)$	$X \in L^2(T)$ $X = \{X(t), t \in T\}$
Base de l'ACP	vecteurs propres $(a_j)_{j=1,\ldots,d}$ de la matrice de covariance	fonctions propres $(\psi_j)_{j\geq 1}$ de l'opérateur de covariance
Représentation ACP	$X = \mathbb{E}[X] + \sum_{j=1}^d \xi_j a_j$	$X(t) = \mathbb{E}[X](t) + \sum_{j=1}^d \xi_j \psi_j(t)$

Table 4.2 – Cas fonctionnel vs. multivarié, ACP.

Ainsi, chaque ψ_j satisfait

$$\Gamma_X \psi_j = \lambda_j \psi_j \text{ ou encore } \int_T C_X(s,\cdot)\psi_j(s)ds = \lambda_j \psi(\cdot), \tag{4.2}$$

où C_X désigne la fonction de covariance de X.

Remarquons que, comme en multivarié, il y a une ambiguïté dans la définitions des fonctions ψ_k. En effet si ψ_k satisfait les conditions ci-dessus, alors $-\psi_k$ aussi. La base de l'ACP est donc connue uniquement au signe près. On peut mettre en parallèle ACP multivariée et fonctionnelle, c'est le but de la Table 4.2

4.2.3 ACP fonctionnelle à partir d'un échantillon de courbes

En pratique, comme nous l'avons déjà noté, la base de Karhunen-Loève est généralement inconnue, puisque la loi du processus dont les observations sont supposées être des réalisations l'est. Se pose alors la question du calcul de l'ACP en pratique. D'abord, il existe quelques rares situations où, même si la loi et l'opérateur de covariance de X sont inconnus, la base de Karhunen-Loève est connue : c'est le cas des données circulaires ($T = [a, b]$, $X(a) = X(b)$ et X stationnaire au second ordre). Dans le cas général, on utilise, comme en multivarié, une version empirique des arguments précédents.

Supposons donc maintenant que l'on observe n réalisations *i.i.d.* $(X_1, \ldots, X_n)$ d'une variable $X \in L^2(T)$, centrée. On va utiliser la version empirique de l'opérateur de covariance, introduit à la Définition 4.3. La fonction de covariance empirique est, dans le cas centré,

$$\begin{aligned} C_n : T \times T &\longrightarrow \mathbb{R} \\ (s,t) &\longmapsto C_n(s,t) = \mathrm{Cov}_X(s,t) = \frac{1}{n}\sum_{i=1}^n X_i(t)X_i(s). \end{aligned}$$

Le lien avec l'opérateur de covariance empirique, dans notre cas (processus de L^2), est le suivant :

$$\Gamma_n f(s) = \int_T C_n(s,t)f(t)dt, \quad s \in T, \quad f \in L^2(T).$$

L'opérateur Γ_n est de rang fini, donc, comme l'opérateur de covariance, il est compact. Par ailleurs, il est également auto-adjoint. Ainsi, il est diagonalisable. On note $(\widehat{\psi}_j)_{j\geq 1}$ ses fonctions propres, associées aux valeurs propres $(\hat{\lambda}_j)_{j\geq 1}$. Comme Γ_n est de rang fini (rang inférieur ou égal à n), la suite des $(\hat{\lambda}_j)_{j\geq 1}$ est nulle à partir d'un rang $J_0 := \text{rg}(\Gamma_n) + 1$. Les propriétés asymptotiques (consistance, normalité asymptotique) des suites $(\widehat{\psi}_j)_{j\geq 1}$ et $(\hat{\lambda}_j)_{j\geq 1}$ comme estimateurs respectifs des fonctions propres et valeurs propres de l'opérateur de covariance ont été étudiées par Dauxois *et al.* (1982); Hall et Hosseini-Nasab (2006); Cardot *et al.* (2007). Par ailleurs des versions lissées des $(\widehat{\psi}_j)_{j\geq 1}$ ont également été étudiées du fait du peu régularité des courbes $(\widehat{\psi}_j)$ lorsque les données sont peu régulières (voir Rice et Silverman 1991; Lee *et al.* 2002, par exemple).

4.2.4 Mise en pratique de l'ACP fonctionnelle

Comme pour toute implémentation, nous sommes limités quant aux possibilités de l'ordinateur de stocker des données en dimension infinie, et de faire des calculs en dimension infinie. Le problème de calcul des fonctions propres de l'opérateur de covariance empirique doit donc être discrétisé. Supposons toujours que l'on observe n réalisations *i.i.d.* $(X_1, \ldots, X_n)$ d'une variable $X \in L^2(T)$. On suppose qu'elles ont été lissées et centrées.

4.2.4.1 Première possibilité : utilisation des données discrétisées

La première solution pour un calcul approché consiste à utiliser les données discrétisées selon une grille suffisamment fine $(t_1, \ldots, t_d)$. Pour chaque X_i, $i \in \{1, \ldots, n\}$, on dispose alors de $X_i(t_1), \ldots, X_i(t_d)$. On a donc en fait une matrice X_c de taille $n \times d$, comme en (4.1). La matrice de covariance empirique associée est $\Sigma = n^{-1t}X_cX_c$, dont les éléments sont les $C_n(t_j, t_k)$, pour $1 \leq j, k \leq d$. On peut alors appliquer une ACP multivariée classique, décrite à la section (4.2.1). On obtient des vecteurs propres $(a_j)_{j=1,\ldots,d}$, formant une base dans laquelle on peut représenter les données discrétisées.

La question qui se pose alors est la suivante : comment récupérer les fonctions propres $(\widehat{\psi}_j)_{j\geq 1}$ de l'opérateur de covariance empirique Γ_n à partir de ces vecteurs propres ? Le point de départ est le suivant. Pour toute fonction f suffisamment régulière, l'intégrale sur un intervalle borné peut être approchée par une somme de Riemann, en particulier,

$$\Gamma_n f(t_j) = \int_T C_n(t_j, t) f(t) dt \approx \frac{|T|}{d} \sum_{k=1}^{d} C_n(t_j, t_k) f(t_k) = \frac{|T|}{d} \Sigma \tilde{f}, \tag{4.3}$$

où $\tilde{f} = {}^t(f(t_1), \ldots, f(t_d))$ et $|T|$ est la longueur de T. Ainsi, si ψ est une fonction propre de Γ_n associée à la valeur propre λ, l'équation $\Gamma_n \psi = \lambda \psi$ peut être approchée par

$$\frac{|T|}{d} \Sigma \tilde{\psi} = \lambda \widetilde{\psi} \iff \Sigma \tilde{\psi} = \frac{d}{|T|} \lambda \widetilde{\psi}.$$

Donc, si a est vecteur propre de Σ associé à la valeur propre ρ, alors $\widetilde{\psi} = (T/d)^{-1/2} a$ est une discrétisation d'une fonction propre normalisée (grâce au coefficient $(T/d)^{-1/2}$) de Γ_n associé à la valeur propre $\lambda = (|T|/d)\rho$. Pour passer de $\widetilde{\psi}$ à ψ, il reste à utiliser une méthode d'interpolation (puisque la grille est supposée fine, cela suffit). Remarquons que l'approximation de l'intégrale par une somme de Riemann peut être remplacée par des méthodes dites de quadrature (méthodes de calcul numérique d'intégrales) plus fines, telle la méthode des trapèzes.

4.2.4.2 Seconde possibilité : utilisation du développement dans une base données

La méthode précédente est simple, mais tend à méconnaître le caractère fonctionnel des données que l'on souhaite justement exploiter. Une seconde méthode consiste à utiliser le développement des fonctions X_i, $1 \leq i \leq n$ dans une base de fonctions classiques $\{\varphi_1, \ldots, \varphi_D\}$ donnée. On notera $W = (w_{j,k})_{1 \leq j,k \leq D}$ la matrice $D \times D$ telle que $W_{j,k} = \langle \varphi_j, \varphi_k \rangle = \int_T \varphi_j(t)\varphi_k(t)dt$. Remarquons que W est la matrice identité si la base est orthonormée. On écrit chaque fonction X_i comme

$$X_i(t) = \sum_{j=1}^{D} \theta_{i,j}\varphi_j(t), \quad t \in T, \quad i \in \{1, \ldots, n\}.$$

Si on note X le "vecteur" dont les composantes sont les fonctions $X_1, \ldots, X_n$ et Φ celui dont les composantes sont $\varphi_1, \ldots, \varphi_D$, on peut résumer ceci par

$$X = \Theta\Phi, \quad \Theta = (\theta_{i,j})_{\substack{1 \leq i \leq n \\ 1 \leq j \leq D}}.$$

La fonction de covariance empirique s'écrit alors

$$C_n(s,t) = \frac{1}{n} {}^t\Phi(s) {}^t\Theta\Theta\Phi(t).$$

Supposons alors que $\widehat{\psi} = \sum_{j=1}^{D} b_j\varphi_j = {}^t\Phi(t)b$ est fonction propre de l'opérateur de covariance empirique Γ_n associé à la valeur propre λ. On note $b = {}^t(b_1, \ldots, b_D)$, et on impose $\|\widehat{\psi}\| = 1$, ce qui signifie ${}^tbWb = 1$. L'équation $\Gamma_n\widehat{\psi} = \lambda\widehat{\psi}$ se traduit "matriciellement" : en effet,

$$\Gamma_n\widehat{\psi}(s) = \int_T C_n(s,t)\widehat{\psi}(t)dt = \int_T \frac{1}{n} {}^t\Phi(s) {}^t\Theta\Theta\Phi(t) {}^t\Phi(t)bdt = {}^t\Phi(s)\frac{1}{n} {}^t\Theta\Theta Wb, \quad s \in T$$

et donc

$${}^t\Phi(s)\frac{1}{n} {}^t\Theta\Theta Wb = \lambda {}^t\Phi(s)b, \quad s \in T,$$

ce qui matriciellement devient $n^{-1} {}^t\Theta\Theta Wb = \lambda b$. En posant $a = W^{1/2}b$, on obtient donc le problème de recherche de valeurs propres/vecteurs propres matriciel

$$\frac{1}{n} W^{1/2 t}\Theta\Theta W^{1/2}a = \lambda a.$$

Quand on a trouvé a, il reste ensuite à poser $b = W^{-1/2}a$, pour déterminer $\widehat{\psi}$. Remarquons que dans le cas $W = I$, le problème est ramené à celui de l'ACP multivariée du tableau de coefficients Θ. Dans tous les cas, le nombre maximal de fonctions propres que l'on peut déterminer avec cette méthode est D, la dimension de l'espace d'approximation $\text{Vect}\{\varphi_1, \ldots, \varphi_D\}$.

4.2.5 Représentations graphiques des résultats et exemples

On peut faire différents types de graphiques pour illustrer les résultats d'un calcul d'ACP fonctionnelle.

1. Nous pouvons tout d'abord tracer les premières fonctions propres $(\widehat{\psi}_j)_{j \geq 1}$ (ou composantes principales), et observer la manière dont elles varient en fonction de leur paramètre. On trace généralement les premières, de manière à obtenir une proportion de variance expliquée supérieure ou égale à 90%.

2. On peut ensuite, comme en multivarié, représenter, dans un plan, les coordonnées des courbes de l'échantillon de départ selon la première et la seconde composantes principales (ou la seconde et la troisième, ...). Ce qui permet d'avoir un nuage de point (un point = une courbe) dans un plan.
3. Si l'on suit Ramsay et Silverman (2005), il est également possible de représenter les composantes principales comme perturbation de la fonction moyenne : on superpose alors sur un même graphique, la fonction moyenne empirique de l'échantillon de courbes, et les deux courbes dont les équations sont calculées à partir de la fonction moyenne à laquelle on ajoute ou on soustrait un multiple raisonnable d'une des fonctions propres. L'idée est qu'une fonction propre représente une variation autour de la moyenne.

Code R. ACP. La fonction `pca.fd` permet de mettre en oeuvre l'ACP fonctionnelle.

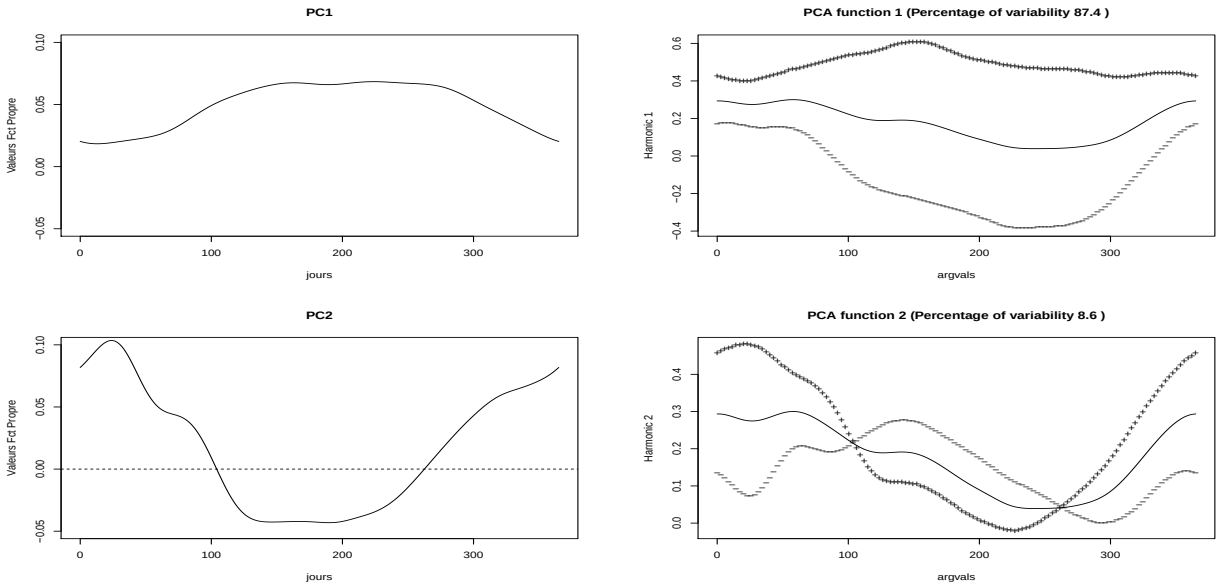
1. **Arguments principaux.**
   - `fdobj` : un objet de la classe `fd` (données fonctionnelles après lissage), éventuellement multivarié,
   - `nharm` : le nombre de fonctions propres à calculer (aussi appelées harmoniques),
   - `centerfns` : booléen, si la valeur est `TRUE`, centrage des données avant le calcul de l'ACP.
2. **Sortie.** La fonction retourne un objet de la classe `pca.fd`, dont les entrées principales sont les suivantes
   - `harmonics` : objet fonctionnel (classe `fd`) contenant les fonctions propres,
   - `values` : l'ensemble complet des valeurs propres de l'opérateur de covariance empirique,
   - `scores` : matrice des scores ie. des coordonnées des individus (courbes) dans la nouvelle base, ie. selon chacune des fonctions propres,
   - `varprop` : un vecteur donnant la proportion de variance expliquée par chaque fonction propre,
   - `meanfd` : objet fonctionnel (classe `fd`) contenant la fonction moyenne.

   La fonction `plot.pca.fd` permet d'obtenir le 3ème type de représentation graphique évoqué ci-dessus.

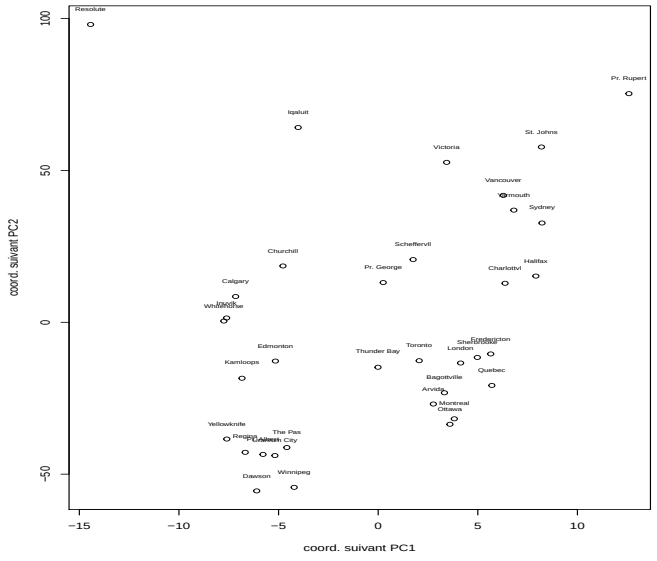
Nous illustrons l'utilisation de ces fonctions sur quelques jeux de données du package `fda` de R déjà utilisés. Le première exemple concerne les log-précipitations canadiennes (données `CanadianWeather`). La proportion de variabilité expliquée par les 2 premières fonctions propres est déjà supérieure à 90%. Celles-ci sont représentées à la figure 4.6 : d'abord seules, puis comme variations de la moyenne. Sur cette même figure, on représente également les différentes stations météorologiques dans le plan défini par les deux premières fonctions propres. L'ACP des données de températures du même jeu de données fait l'objet de la Figure 4.7.

Notons enfin que les possibilités de l'ACP sont multiples. Tout d'abord, on a décrit l'ACP pour des échantillons de courbes univariés. Or, on peut également, en modifiant légèrement la procédure, l'appliquer pour des données fonctionnelles multivariées (en définissant un produit scalaire sur $L^2(T) \times L^2(T)$ dans le cas de données bivariées par exemple). Sans donner de détails, les deux premières fonctions propres (bivariées donc) obtenues pour les données d'écriture `Handwrit` sont représentées comme variations de la moyenne à la Figure 4.8.

Par ailleurs, la méthode garantit que la base $(\widehat{\psi}_j)_{j\geq 1}$ est la famille de fonctions orthonormales minimisant un critère quadratique intégré, et joue ainsi le rôle de meilleur famille

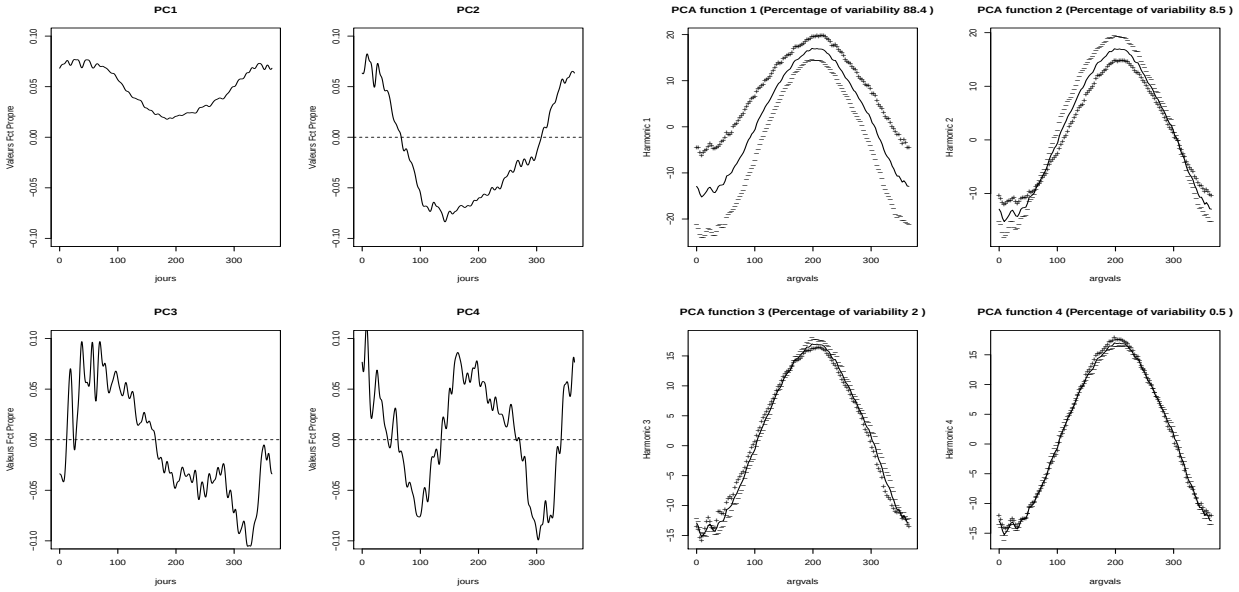


(a) (b)

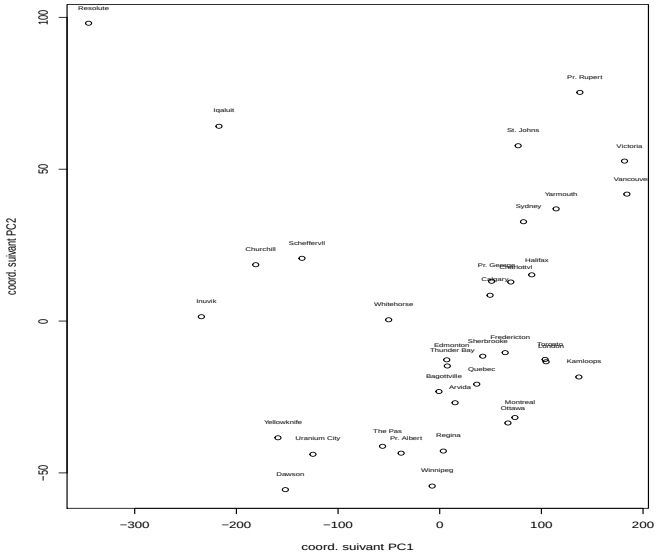


(c)

FIGURE 4.6 – ACP pour les données de log-précipitations canadiennes (données `CanadianWeather`) (a) Représentations des deux premières fonctions propres. (b) Représentation des deux fonctions propres comme variations autour de la moyenne (trait plein : fonction moyenne, traits pointillés : fonction moyenne modifié d'un multiple de la fonction propre). (c) Représentation des villes dans le plan défini par les deux premières fonctions propres.

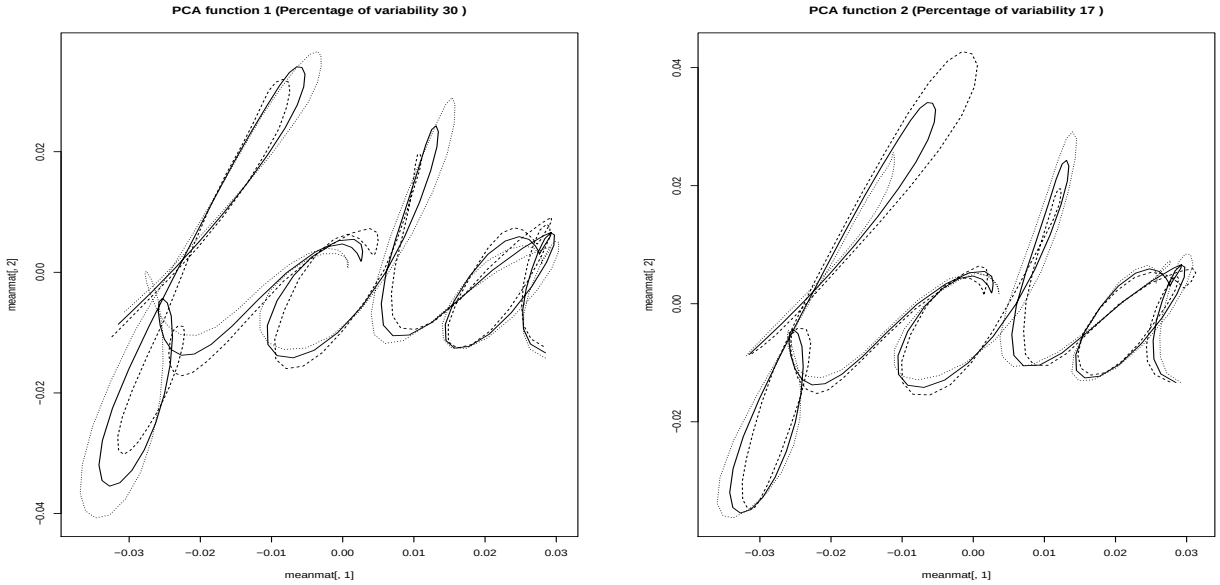


(a) (b)

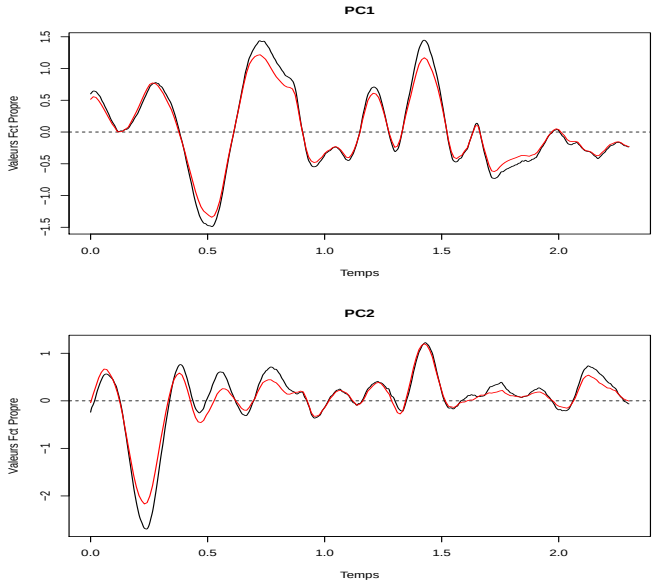


(c)

FIGURE 4.7 – ACP pour les données de températures canadiennes (données `CanadianWeather`) (a) Représentations des quatre premières fonctions propres. (b) Représentation des quatre fonctions propres comme variations autour de la moyenne (trait plein : fonction moyenne, traits pointillés : fonction moyenne modifié d'un multiple de la fonction propre). (c) Représentation des villes dans le plan défini par les quatre premières fonctions propres.

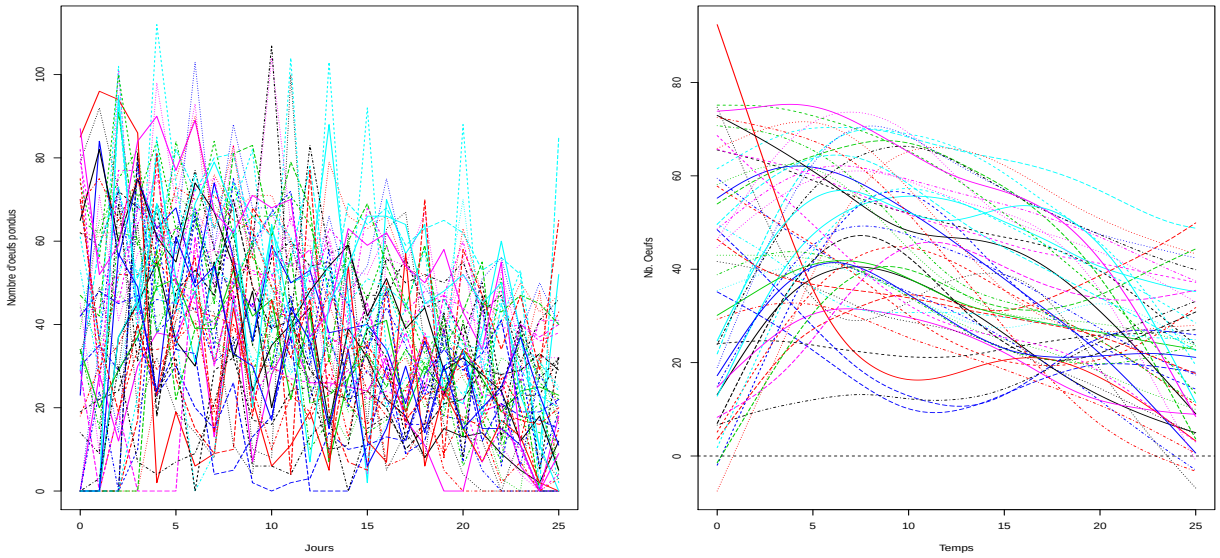


(a) (b)



(c)

FIGURE 4.8 – ACP pour les données `Handwrit` (a) et (b) Représentations de la première (graphique (a)) et de la seconde (graphique (b)) fonction propre comme variation de la moyenne (trait plein : fonction moyenne, traits pointillés : fonction moyenne modifiée d'un multiple de la fonction propre). (c) Comparaison des fonctions propres obtenues pour les ordonnées par une ACP univariée (ligne noire) ou en extrayant la seconde composante de l'ACP fonctionnelle pour données multivariées (ligne rouge).



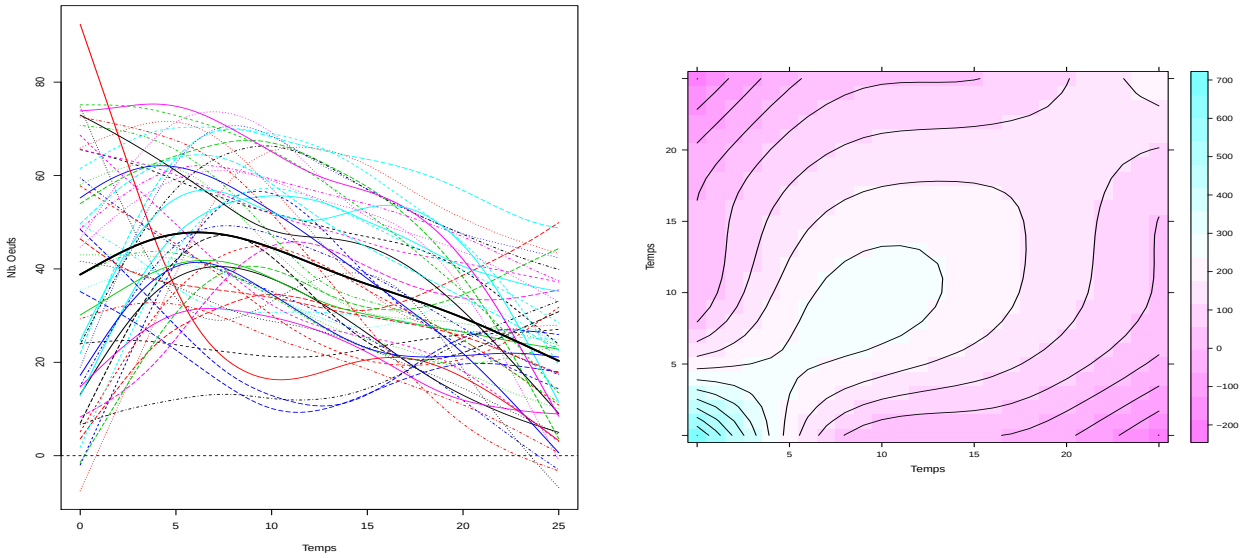
(a) (b)

FIGURE 4.9 – Données `Medfly`. (a) Données brutes (b) Données lissées.

approximant les courbes de départ. Ceci ne signifie pas qu'il n'existe pas d'autres systèmes de fonctions vérifiant ces propriétés : toute transformation orthogonale, ie. toute rotation des fonctions de bases les transforme en une autre famille de fonctions qui peuvent s'avérer utile pour représenter les courbes de l'échantillon. Ainsi, certaines stratégies, comme la stratégie VARIMAX, permettent de choisir une matrice de rotation permettant de passer des $(\widehat{\psi}_j)_{j\geq 1}$ à des fonctions mieux interprétables. Nous ne rentrerons pas dans les détails ici, et renvoyons à la Section 8.3.3 de Ramsay et Silverman (2005) par exemple.

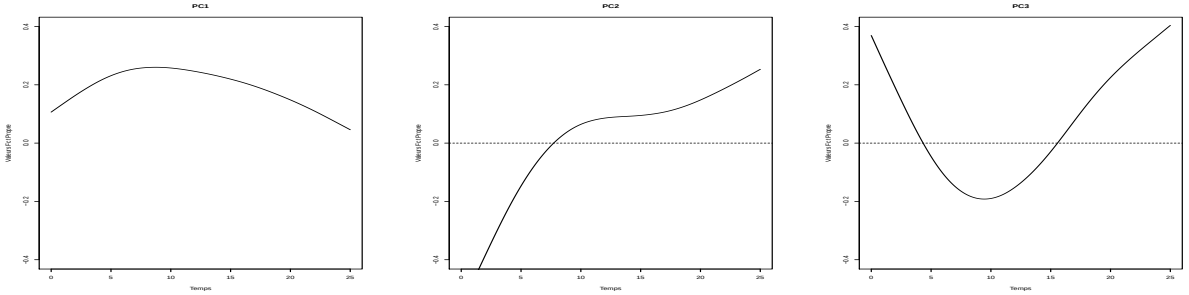
Exercice 7. Étude des données `medfly`. Il s'agit de données concernant le nombre d'oeufs pondus par 50 mouches méditerranéennes des fruits pendant 25 jours.

1. Charger le fichier de données `medfly.Rdata` sur l'espace *MyCourse* du cours, représenter les données `medfly$eggcount`.
2. Proposer un lissage des données par moindres carrés pénalisés dans une base de splines cubiques sur $[0, 25]$, avec noeuds en chaque entier. On ajustera le paramètre λ de la pénalité usuelle par validation croisée généralisée, en choisissant dans l'ensemble $\{e^{-10}, e^{-9}, \ldots, e^{9}, e^{10}\}$. Après ces deux premières question, la Figure 4.9 devrait être obtenue.
3. Représenter les fonctions lissées, superposer sur le même graphique la fonction moyenne empirique (Figure 4.10).
4. Calculer et représenter la covariance empirique des données (Figure 4.10).
5. Effectuer l'ACP de ces données. Combien de fonctions propres sont nécessaire pour expliquer 90% de la variabilité ? Les représenter (Figure 4.11).



(a) (b)

FIGURE 4.10 – Données `Medfly` - statistique descriptive (a) Données lissées (traits fins) et fonction moyenne empirique (trait épais noir) (b) Lignes de niveau de la fonction de covariance empirique.



(a) (b) (c)

FIGURE 4.11 – Données `Medfly` - ACP. Représentation des trois premières fonctions propres.

Exercice 8. Étude des données `pinch` du package `fda`. Déjà décrites à la Section 1.1.1 et représentées à la Figure 1.4, il s'agit de 20 enregistrements de l'évolution au cours d'une manipulation de pincement de la force exercée entre le pouce et l'index d'un sujet.

1. Représenter les données `pinchraw` et `pinch`. Quelle est la différence entre les deux ?
2. Proposer un lissage des données par moindres carrés pénalisés dans une base de splines cubiques sur $[0, 150]$, avec noeuds en chaque entier. On ajustera le paramètre λ de la pénalité usuelle par validation croisée généralisée, en choisissant dans l'ensemble $\{e^{-10}, e^{-9}, \ldots, e^{9}, e^{10}\}$.
3. Effectuer l'ACP de ces données. Combien de fonctions propres sont nécessaire pour expliquer 90% de la variabilité ? Les représenter.

Appendice : Code R de certaines figures

Les lignes de commande R permettant d'obtenir la plupart des figures de ce document sont fournies dans cette section. Les codes sont enregistrés dans des fichiers d'extension ".R", dont les premières lignes sont les suivantes :

```
rm(list=ls())
library('fda')
palette("default")
```

Codes pour les figures du Chapitre 1

Figure 1.1 (a)

```
op <- par(mar=c(5, 4, 4, 5)+.1)
stations <- c("Pr. Rupert", "Montreal", "Edmonton", "Resolute")
matplot(day.5, CanadianWeather$dailyAv[, stations, "Temperature.C"],
        type="l", axes=FALSE, xlab="", ylab="Temperature moyenne (deg C)")
axis(2, las=1)

axis(1, monthBegin.5, labels=FALSE)
axis(1, monthEnd.5, labels=FALSE)
axis(1, monthMid, monthLetters, tick=FALSE)
matpoints(monthMid, CanadianWeather$monthlyTemp[, stations])
mtext(stations, side=4,
      at=CanadianWeather$dailyAv[365, stations, "Temperature.C"],
      las=1)
```

Code pour la Figure 1.1 (b)

```
logprecav<-CanadianWeather$dailyAv[dayOfYearShifted,,'log10precip']
plot(c(1,365),range(logprecav),type='n',main='Précipitations au Canada',
      xlab='Jours (du 1er juillet au 30 juin)',ylab='log en base 10 des précipitations')
for (i in c(1,15,20)){
  points(seq(1,365,1),logprecav[,i],type='l',col=i)
}
legtxt <-  c("St Johns", "London", "Regina")
legend(1,-0.5,legtxt,col=c(1,15,20),lty=1, cex=0.8)
```

Figure 1.3

```
plot(range(growth$age),range(growth$hgtf),type='n',
       main='Courbes de croissance',xlab='Age (annees)',ylab='Taille (cm)')
for (i in 1:10){
  points(growth$age,growth$hgtf[,i],type="l",col=i)
  points(growth$age,growth$hgtf[,i],type="o",col=i)
}
```

Figure 1.4

```
matplot (pinchtime, pinchraw, type="l", lty=1, cex=2,
         col=1, lwd=1, xlab = "Temps (sec.)", ylab="Force (Newton)",
         main='Données de force du groupe Pouce-Index')
abline(h=2, lty=2)
```

Figure 1.5

```
data<-as.matrix(read.table("npfda-spectrometric.dat"))
X<-data[,1:100]
Y<-data[,101]
n<-length(X)
p<-100
x<-seq(850,1050,length.out=p)
palette(rainbow(50,start=0,end=4/6))
plot(range(x),range(X),type='n',xlab='longueur d onde (nm)',
          ylab='absorbance',main='Donnees spectrometriques')
for (i in 1:100){
  points(x,X[i,],type='l',col=50-floor(Y[i]))
}
```

Figure 1.6 (a)

```
matplot(handwritTime,handwrit[,,1],type='l',
        xlab='Temps (ms.)',ylab='Abscisse d ecriture',
        main='Representation d une des coordonnees')
```

Figure 1.6 (b)

```
plot(range(handwrit[,,1]), range(handwrit[,,2]),
         type="n",main='Echantillons d écriture à la main',xlab='x',ylab='y')
for (i in 1:20){
  points(handwrit[,i,1],handwrit[,i,2],type="l")
}
```

Figure 1.7 (a)

```
plot(range(refinery$Time),range(refinery$Tray47),type='n',
        main='Donnees de production du niveau 47 de la raffinerie',
        xlab='Temps (min.)',ylab='Quantité de petrole produit')
points(refinery$Time,refinery$Tray47,type='o')
```

Figure 1.7 (b)

```
plot(range(refinery$Time),range(refinery$Reflux),type='n',
        main='',xlab='Temps (min.)',ylab='Flux de vapeur au niveau 47')
points(refinery$Time,refinery$Reflux,type='o')
```

Figure 1.8

```
plot(melanoma[,1],melanoma[,2],type='o',
       main='Taux d incidence du melanome',xlab='Temps (années)',ylab='Taux')
```

Codes pour les figures du Chapitre 2

Figures 2.1 (a) et (b)

Création d'une fonction simulant un mouvement Brownien (sauvegarde dans un fichier `brownien.R`)

```
MvB=function(n,p=100)
{
 X=rnorm(n*(p-1))/sqrt(p-1)
 X=matrix(X,ncol=p-1)
 MvB=matrix(0,ncol=p,nrow=n)
 for (i in 1:n)
 {
  MvB[i,]=c(0,cumsum(X[i,]))
 }
 MvB
}
```

Utilisation de la fonction pour obtenir la figure (a) :

```
source("brownien.R")
n <- 5
p <- 100
X <- MvB(n,p)
plot(c(0,1),range(X),type='n',main='Mouvement brownien',xlab='t',ylab='X(t)')
t <- seq(0,1,length.out=p)
for (i in 1:5){
  points(t,X[i,],type='l',col=i)
}
```

Ensuite, pour obtenir la figure (b) :

```
W<-array(0,c(n,p))
for (i in 1:5){
  W[i,]<-X[i,]-t*X[i,p]
}
matplot(t,t(W),type='l', main='Pont brownien',xlab='t',ylab='W(t)')
```

Figures 2.1 (c)

```
ksi=matrix(rnorm(n*2),n,2)
vectt=seq(0,1,length.out=p)
base=matrix(rep(0,p*2),2,p)
base[1,]=1
base[2,]=sqrt(2)*sin((pi/2)\%*\%t(vectt))
X=ksi\%*\%base
plot(c(0,1),range(X),type='n',main='Processus plus régulier',xlab='t',ylab='X(t)')
for (i in 1:n){
  points(vectt,X[i,],type='l',col=i)
}
```

Figure 2.2

Création d'une fonction servant à construire les processus des 3 figures

```
simuX=function(n,p=100,J=150,lambda=(1:J)^(-2),loi='norm',...)
{
  J <- length(lambda)
  if (loi=='norm')  ksi=rnorm(n*J) else ksi=runif(n*J,-sqrt(3),sqrt(3))
  ksi=matrix(ksi,n,J)*matrix(rep(sqrt(lambda),n),n,J,byrow=TRUE)
  x=matrix(rep(seq(0,1,length.out=p),J),J,p,byrow=TRUE)
  Jm=matrix(rep(seq(0.5,J-0.5,by=1),p),J,p)
  base=sqrt(2)*sin(pi*Jm*x)
  pertinit <- matrix(rep(rnorm(n),p),n,p,byrow=FALSE)
  ksi%*%base + pertinit
}
```

Utilisation de la fonction pour obtenir la première des 3 figures par exemple :

```
n <- 5
p <- 100
t <- seq(0,1,length.out=p)
X <- simuX(n,p) # lambda_j=j^(-2) (defaut)
plot(c(0,1),range(X),type='n',xlab='t',ylab='X(t)')
for (i in 1:5){
  points(t,X[i,],type='l',col=i)
}
```

Figure 2.3

```
base_Fourier <- create.fourier.basis(c(0,2*pi),9) #période par défaut: 1
plot(base_Fourier,main="Base de Fourier",xlab='t',ylab='x(t)')
```

Figure 2.4

```
base_monomes <- create.monomial.basis(rangeval=c(-1,1),nbasis=10)
plot(base_monomes,main="Base des monômes",xlab='t',ylab='x(t)')
```

Figure 2.5 (a)

```
base_Bsplines2 <- create.bspline.basis(rangeval=c(0,10), nbasis=13, norder=2)
plot(base_Bsplines2,main="B-splines d'ordre 2 de base",xlab='t',ylab='x(t)')
```

Figure 2.5 (b)

```
base_Bsplines <- create.bspline.basis(rangeval=c(0,10),nbasis=13) #norder=4
plot(base_Bsplines,main="B-splines cubiques de base (ordre 4)",xlab='t',ylab='x(t)')
```

Codes pour les figures du Chapitre 3

Figure 3.1

```
Donnees_brutes_temp<- t(MontrealTemp[, 16:47]) #matrice 32*34
Jours <-((16:47)+0.5)

Fct_base_temp <- create.bspline.basis(c(16,48),7)
MatPhi_temp <- eval.basis(Jours,Fct_base_temp)

Coeff_MC_temp <- solve(crossprod(MatPhi_temp),crossprod(MatPhi_temp,Donnees_brutes_temp))
Donnees_lisseesMC_temp<-fd(Coeff_MC_temp,Fct_base_temp,
                                                    list("Jours","années","Température (deg.

# Figure de la 1ère ligne : toutes les courbes lissees
plot(Donnees_lisseesMC_temp,lty=1,col=1)


# Figure (a)  donnees brutes de 1990
plot(Jours,t(Donnees_brutes_temp[,30]),"b", lwd=2,
                 xlab="Jours",ylab='Températures (deg. C)',
                 main="Température journalière en 1990")

# Figure (b)
plotfit.fd(Donnees_brutes_temp[,30],Jours,Donnees_lisseesMC_temp[30],lty=1,lwd=2,
                  main='Donnees lissees vs. brutes',
                  xlab="Jours",ylab='Températures (deg. C)')
```

Figures 3.2 et 3.3

```
Noeuds_Refin<-c(seq(0,67,length.out=3),67,seq(67,193,length.out=3)); #7 noeuds
Fct_base_Refin <- create.bspline.basis(c(0,193),norder=4,breaks=Noeuds_Refin) #9 fct base



MatPhi_Refin <-predict(Fct_base_Refin,refinery$Time) #matrice 194 lignes 9 colonnes
 # ou M<-eval.basis(refinery$Time,Fct_base_Refin)
Coeff_MC_Refin<- solve(crossprod(MatPhi_Refin),crossprod(MatPhi_Refin,refinery$Tray47))
Donnees_lisseesMC_Refin <-fd(Coeff_MC_Refin,Fct_base_Refin)
```

```
#Figure 1 (a)
plot(Fct_base_Refin,xlab='Temps (min.)')

#Figure 1 (b)
plotfit.fd(refinery$Tray47,refinery$Time,Donnees_lisseesMC_Refin,lty=1,lwd=1,
         main='Donnees lissees vs. brutes',col=1,
         xlab='Temps (min.)',ylab='Production de pétrole')

#Figure 2 (a)
plot(Donnees_lisseesMC_Refin,xlab='Temps (min.)',ylab='Production de pétrole')

#Figure 2 (b)
plot(Donnees_lisseesMC_Refin,Lfdobj=1,xlab='Temps (min.)',ylab='')
```

Figure 3.4

```
#Figures des 2 premieres lignes
pts_rupture_handwrit1 = seq(0,2.3,length.out=21)
nb_Fct_base_handwrit1 = 23 #19+4
Fct_base_handwrit1 = create.bspline.basis(c(0,2.3),norder=4,breaks=pts_rupture_handwrit1)
tps_handwrit  <- seq(0, 2.3, len=1401)
noms_variables = list('Seconds',NULL,c('X','Y'))
Donnees_lissees_handwrit1 = smooth.basis(tps_handwrit,handwrit,
                           Fct_base_handwrit1,fdnames=noms_variables)


par(mfrow=c(2,1))
plot(Donnees_lissees_handwrit1$fd) # premiere colonne
plotfit.fd(handwrit,tps_handwrit,Donnees_lissees_handwrit1$fd,type='p')
            # seconde colonne (on ne prend que le graphe 18 !)

#Figures des 2 secondes lignes
pts_rupture_handwrit2 = seq(0,2.3,length.out=51)
Fct_base_handwrit2 = create.bspline.basis(c(0,2.3),norder=6,breaks=pts_rupture_handwrit2)
plot(Fct_base_handwrit2)
Donnees_lissees_handwrit2 = smooth.basis(tps_handwrit,handwrit,
             Fct_base_handwrit2,fdnames=noms_variables)

plotfit.fd(handwrit,tps_handwrit,Donnees_lissees_handwrit2$fd,nfine=1001,
                 col=4,lwd=2,cex.pch=0.2)  # premiere colonne
par(mfrow=c(2,1))
plot(Donnees_lissees_handwrit2$fd)
     # seconde colonne (on ne prend que le graphe 18 !)
```

Figure 3.5 (le code fait suite à celui de la figure précédente)

```
plot(Donnees_lissees_handwrit2$fd,Lfdobj=1) # premiere colonne
plot(Donnees_lissees_handwrit2$fd,Lfdobj=2) # seconde colonne
```

```
par(mfrow=c(1,1)) # pour revenir à une fenêtre classique pour les graphes suivants
```

Figure 3.6

```
Noeuds_refin<-sort(c(refinery$Time,67,67))
Fct_Base_refin<-create.bspline.basis(c(0,193),norder=4,breaks=Noeuds_refin)

fdpar_refin1<-fdPar(fdobj=Fct_Base_refin,Lfdobj = 2,lambda=0.00001)
Donnees_lissees_refin1 <-smooth.basis(argvals=refinery$Time,
                         y=refinery$Tray47,fdParobj=fdpar_refin1)
plotfit.fd(refinery$Tray47,refinery$Time,Donnees_lissees_refin1$fd,
                        xlab='Temps (min.)',ylab="Quantite de pétrole produit")
                        # courbe noire
fdpar_refin2<-fdPar(fdobj=Fct_Base_refin,Lfdobj = 2,lambda=10^6)
Donnees_lissees_refin2 <-smooth.basis(argvals=refinery$Time,
                        y=refinery$Tray47,fdParobj=fdpar_refin2)
lines(Donnees_lissees_refin2$fd,col=2)
                        # courbe rouge
```

Figures 3.7 (températures) et 3.8 (précipitations)

```
temperature = CanadianWeather$dailyAv[,,'Temperature.C']
precipitation = CanadianWeather$dailyAv[,,'Precipitation.mm']
jours = 1:365 - 0.5

#definition de la penalite et de la base de lissage
op_harmonique = vec2Lfd( c(0,(2*pi/365)^2,0),c(0,365) )
Fct_base_CanadianW= create.fourier.basis(c(0,365),nbasis=365)

#optimisation du parametre de penalite par gcv pour le lissage des temp. et precip.
gcvtemp = matrix(0,31,35)
gcvprec = matrix(0,31,35)
for(i in 1:31){
  lambda = exp(i-10) # param. de penalite
  fdPar_CanadianW = fdPar(Fct_base_CanadianW,op_harmonique,lambda)
  gcvtemp[i,] = smooth.basis(jours,temperature,fdPar_CanadianW)$gcv
            #valeur du critere gcv pour le lissage des temp.
  gcvprec[i,] = smooth.basis(jours,precipitation,fdPar_CanadianW)$gcv
            #valeur du critere gcv pour le lissage des precip.
}

## Cas du lissage des courbes de temp.
# recherche du minimum
i = which.min(mgcvtemp)
lambda_gcv_temp = exp(i-10)

# lissage avec param. optimal
fdPar_CanadianTemp_gcv = fdPar(Fct_base_CanadianW,op_harmonique,lambda_gcv_temp)
Donnees_lissees_temp = smooth.basis(jours,temperature,fdPar_CanadianTemp_gcv)$fd
```

```
## Cas du lissage des courbes de precip.
# recherche du minimum
i = which.min(mgcvprec)
lambda_gcv_precip = exp(i-10)

# lissage avec param. optimal
fdPar_CanadianPrecip_gcv = fdPar(Fct_base_CanadianW,op_harmonique,lambda_gcv_precip)
Donnees_lissees_precip = smooth.basis(jours,precipitation,fdPar_CanadianPrecip_gcv)$fd


##Figures
# Premiere ligne de la premiere figure (critere gcv en fonction du param de penalite)
par(mfrow=c(1,1))
matplot(-10:20,gcvtemp,type='l',xlab='log lambda',ylab='gcv') # courbes en trait fin
mgcvtemp = apply(gcvtemp,1,mean)
lines(-10:20,mgcvtemp,lwd=2,col=4) # courbe moyenne

# Seconde ligne de la premiere figure
op <- par(mar=c(5, 4, 4, 5)+.1)
stations <- c("Pr. Rupert", "Montreal", "Edmonton", "Resolute")
matplot(day.5, CanadianWeather$dailyAv[, stations, "Temperature.C"],
        type="l", axes=FALSE, xlab="", ylab="Temperature moyenne (deg C)")
axis(2, las=1)
axis(1, monthBegin.5, labels=FALSE)
axis(1, monthEnd.5, labels=FALSE)
axis(1, monthMid, monthLetters, tick=FALSE)
matpoints(monthMid, CanadianWeather$monthlyTemp[, stations])
mtext(stations, side=4,
      at=CanadianWeather$dailyAv[365, stations, "Temperature.C"],
      las=1)
op <- par(mar=c(5, 4, 4, 5)+.1)
plot(Donnees_lissees_temp[stations],axis=FALSE,xlab="", ylab="Temperature moyenne (deg C)")


#Seconde figure (a)
plot(precipitation[,"St. Johns"],type='l',col=1)
points(precipitation[,"London"],type='l',col=15)
points(precipitation[,"Regina"],type='l',col=20)

#Seconde figure (b)
plot(Donnees_lissees_precip[c("St. Johns","London","Regina")],col=c(1,15,20),
               main='Précipitations au Canada',
               xlab='Jours (du 1er juillet au 30 juin)',ylab='précipitations (mm)')
```

Figure 3.9

```
Fct_base_Growth<-create.bspline.basis(c(1,18),nbasis=12,norder=6)

#optimisation du param. de penalite par gcv
```

```
log_lambda=seq(-6,0,0.25)
gcv_growth<-rep(0,length(log_lambda))
for (i in 1:length(log_lambda)){
  lambda_i<-10^log_lambda[i]
  fdPar_Growth_i<-fdPar(Fct_base_Growth,Lfdobj=4,lambda=lambda_i)
  gcv_growth[i]<-sum(smooth.basis(growth$age,growth$hgtf,fdPar_Growth_i)$gcv)
}
i = which.min(gcv_growth)
lambda_gcv_growth = 10^log_lambda[i]

#lissage avec param. optimal
fdPar_Growth_i<-fdPar(Fct_base_Growth,Lfdobj=4,lambda=lambda_i)
Donnees_lissees_Growth<-smooth.basis(growth$age,growth$hgtf,fdPar_Growth_i)$fd

#Figure (a) (somme du critere gcv en fonction du param. de penalite)
plot(log_lambda,gcv_growth,type='b',xlab='log lambda',ylab='gcv(lambda)',ylim=c(15,25))
#Figure (b) donnees brutes et lissees pour le 2nd individu de l'echantillon
plotfit.fd(growth$hgtf[,2],growth$age,Donnees_lissees_Growth[2],lty=1,lwd=1,
           main='Donnees lissees vs. brutes',
           xlab="Temps (années)", ylab="Taille (cm.)",col=2)
```

Codes pour les figures du Chapitre 4

Il faut ajouter la commande `library('lattice')` au début des fichiers utilisant la commande `levelplot`.

Figure 4.1

```
#simulation des courbes aléatoires  (CL d'une base de splines avec coeff. gaussiens)
pts_rupture = seq(0,2,length.out=21)
extremites=c(0,2)
ordre = 4
nbre_fctBase =length(pts_rupture)+ordre-2
fct_base = create.bspline.basis(extremites,norder=ordre,breaks=pts_rupture)
coefficients_ech = matrix(rnorm(10*nbre_fctBase),nbre_fctBase,10)
objetfd2 = fd(coefficients_ech,fct_base)

#tracé des courbes et de la fonction moyenne
par(mfrow=c(1,1))
plot(objetfd2,xlab="t",ylab="x(t)")
lines(mfd2,lwd=3)
```

Figure 4.2

```
#Données Handwrit  : lissage par moindres carrés en base de splines
tps_handwrit  <- seq(0, 2.3, len=1401)
noms_variables = list('Seconds',NULL,c('X','Y'))
pts_rupture_handwrit = seq(0,2.3,length.out=51)
Fct_base_handwrit = create.bspline.basis(c(0,2.3),norder=6,breaks=pts_rupture_handwrit2)
```

```
Donnees_lissees_handwrit = smooth.basis(tps_handwrit,handwrit,Fct_base_handwrit2,
              fdnames=noms_variables)

#calcul de la courbe moyenne
moy_handwrit = mean(Donnees_lissees_handwrit$fd)

#calcul de la courbe écart type pour les courbes des ordonnées
std_handwrit = std.fd(Donnees_lissees_handwrit$fd[,2])

##Figures
#Figure (a)
plot(Donnees_lissees_handwrit$fd,xlab='Temps (ms.)')

#Figure (b)
plot(moy_handwrit,lwd=2,xlab='Temps (ms.)')

#Figure (c)
plot(std_handwrit)
plot(Donnees_lissees_handwrit$fd[,2])
```

Figure 4.5 (le code fait suite à celui de la Figure 4.2)

```
#Calcul de la fonction de covariance empirique
varfda = var.fd(Donnees_lissees_handwrit2$fd)
varfda_bis=var.fd(Donnees_lissees_handwrit2$fd[,1])

#Evaluation en certains temps seulement
tps_handwrit_reduit = tps_handwrit[14*(0:100)+1]
varfdavals = eval.bifd(tps_handwrit_reduit,tps_handwrit_reduit,varfda)
       #bifd=bivariate funct data object (avec 2 bases une pr t_1, une pr t_2)
varfdavals_bis=eval.bifd(tps_handwrit_reduit,tps_handwrit_reduit,varfda_bis)
      #vardfavals est un tableau 101*101 répété sur 3 niveaux:
        #niveau(1,1): covariance X (meme chose que varfda_bis),
           # ainsi varfdavals[,,1,1] peut être remplacé par varfdavals_bis
        #niveau(1,2) covariance croisée (X,Y),
        #niveau (1,3) covariance Y)

#Calcul de la corrélation des données d'abscisses
corfda = cor.fd(tps_handwrit_reduit,Donnees_lissees_handwrit2$fd[,1],
                 tps_handwrit_reduit,Donnees_lissees_handwrit2$fd[,2])

##Figures
par(mfrow=c(1,1))

#Figure (a)
contour(tps_handwrit_reduit,tps_handwrit_reduit,varfdavals_bis,
               xlab="Temps (ms.)", ylab="Temps (ms.)")

#Figure (b)
```

```
filled.contour(tps_handwrit_reduit,tps_handwrit_reduit,varfdavals_bis,
               xlab="Temps (ms.)", ylab="Temps (ms.)")

#Figure (c)
levelplot(row.values=tps_handwrit_reduit,column.values=tps_handwrit_reduit,
          x=varfdavals_bis,contour=TRUE,labels=FALSE,
          xlab="Temps (ms.)", ylab="Temps (ms.)")

#Figure (d)
persp(tps_handwrit_reduit,tps_handwrit_reduit,varfdavals_bis,
           xlab="Temps (ms.)", ylab="Temps (ms.)",
           zlab="Covariance des abscisses", phi=25, r=3)

#Figure (e)
levelplot(row.values=tps_handwrit_reduit,column.values=tps_handwrit_reduit,
          x=corfda,contour=TRUE,labels=FALSE,xlab="Temps (ms.)", ylab="Temps (ms.)")

#Figure (f)
levelplot(row.values=tps_handwrit_reduit,column.values=tps_handwrit_reduit,
            x=varfdavals[,,1,2],contour=TRUE,labels=FALSE,
            xlab="Temps (ms.)", ylab="Temps (ms.)")
```

Figure 4.7

On reprend les données lissées `Donnees_lissees_temp` obtenues avec le code de la Figure 3.7.

```
Temp_ACP<-pca.fd(Donnees_lissees_temp,nharm=4)#,fdPar_CanadianTemp_gcv)

# Figure (a)
op=par(mfrow=c(2,2))
plot(Temp_ACP$harmonics[1],xlab="jours",ylab="Valeurs Fct Propre",
        ylim=c(-0.1,0.1), main="PC1")
plot(Temp_ACP$harmonics[2],xlab="jours",ylab="Valeurs Fct Propre",
        ylim=c(-0.1,0.1), main="PC2")
plot(Temp_ACP$harmonics[3],xlab="jours",ylab="Valeurs Fct Propre",
        ylim=c(-0.1,0.1), main="PC3")
plot(Temp_ACP$harmonics[4],xlab="jours",ylab="Valeurs Fct Propre",
        ylim=c(-0.1,0.1), main="PC4")

# Figure (b)
op=par(mfrow=c(2,2))
plot.pca.fd(Temp_ACP, cex.main=0.9,xlab="jours",ylab='température (deg. C.)')

#Figure (c)
par(mfrow=c(1,1))
plot(Temp_ACP$scores[,1],Temp_ACP$scores[,2],
          xlab="coord. suivant PC1", ylab="coord. suivant PC2")
text(Temp_ACP$scores[,1]+5,Temp_ACP$scores[,2]+5,CanadianWeather$place,cex=0.6)
```

Figure 4.8

On reprend les données lissées `Donnees_handwrit_lissees` obtenues avec le code de la Figure 4.2.

```
#ACP bivariee des abscisses et ordonnees
Handwrit_ACP = pca.fd(Donnees_handwrit_lissees$fd,nharm=10)
#ACP des ordonnées uniquement
HandwritY_ACP = pca.fd(Donnees_handwrit_lissees$fd[,2],nharm=8)

#Figure (a)
par(mfrow=c(1,1))
plot.pca.fd(Handwrit_ACP,harm=1,pointplot=FALSE,expand=0.004,cycle=TRUE)

#Figure (b)
plot.pca.fd(Handwrit_ACP,harm=2,pointplot=FALSE,expand=0.004,cycle=TRUE)

#Figure (c)
par(mfrow=c(2,1))
plot(HandwritY_ACP$harmonics[1],main="PC1",xlab="Temps",ylab="Valeurs Fct Propre")
lines(-Handwrit_ACP$harmonics[1,2],col=2)
plot(HandwritY_ACP$harmonics[2],main="PC2",xlab="Temps",ylab="Valeurs Fct Propre")
lines(-Handwrit_ACP$harmonics[2,2],col=2)
```

Bibliographie

A. Ait-Saï di, F. Ferraty, R. Kassa et P. Vieu : Cross-validated estimations in the single-functional index model. *Statistics*, 42(6):475–494, 2008.

R. B. Ash et M. F. Gardner : *Topics in stochastic processes*. Academic Press [Harcourt Brace Jovanovich, Publishers], New York-London, 1975. Probability and Mathematical Statistics, Vol. 27.

P. Besse et J. O. Ramsay : Principal components analysis of sampled functions. *Psychometrika*, 51(2):285–311, 1986.

H. Brezis : *Analyse fonctionnelle*. Masson, Halsted Press, 1983.

M. Briane et G. Pages : *Théorie de l'intégration : licence de mathématiques ; cours et exercices*. Vuibert, 2000.

H. Cardot, F. Ferraty et P. Sarda : Functional linear model. *Statist. Probab. Lett.*, 45 (1):11–22, 1999.

H. Cardot, A. Mas et P. Sarda : CLT in functional linear regression models. *Probab. Theory Related Fields*, 138(3-4):325–361, 2007.

F. Comte : *Estimation non-paramétrique*. Spartacus-IDH, 2015.

J. Dauxois et A. Pousse : *Les analyses factorielles en calcul des probabilités et en statistique : essai d'étude synthétique*. Thèse de doctorat, Université Toulouse III, 1976.

J. Dauxois, A. Pousse et Y. Romain : Asymptotic theory for the principal component analysis of a vector random function : some applications to statistical inference. *J. Multivariate Anal.*, 12(1):136–154, 1982.

C. de Boor : *A practical guide to splines*, vol. 27 de *Applied Mathematical Sciences*. Springer-Verlag, New York, revised édn, 2001.

P. Deheuvels : A Karhunen-Loève expansion for a mean-centered Brownia bridge. *Statist. Probab. Lett.*, 77(12):1190–1200, 2007.

J.-C. Deville : Méthodes statistiques et numériques de l'analyse harmonique. *Ann. I.N.S.É.É.*, (15):3–101, 1974.

N. Dunford et J. T. Schwartz : Linear operators, vol. i. *Interscience, New York*, 1963, 1958.

T. Dunker, M. A. Lifshits et W. Linde : Small deviation probabilities of sums of independent random variables. *In High dimensional probability (Oberwolfach, 1996)*, vol. 43 de *Progr. Probab.*, p. 59–74. Birkhäuser, Basel, 1998.

F. Ferraty : Special issue on statistical methods and problems in infinite dimensional spaces. *J. Multivariate Anal.*, 101(2):305–490, 2010.

F. Ferraty, A. Mas et P. Vieu : Nonparametric regression on functional data : inference and practical aspects. *Aust. N. Z. J. Stat.*, 49(3):267–286, 2007.

F. Ferraty et Y. Romain, éds. *The Oxford handbook of functional data analysis*. Oxford University Press, Oxford, 2011.

F. Ferraty et P. Vieu : The functional nonparametric model and application to spectrometric data. *Comput. Statist.*, 17(4):545–564, 2002.

F. Ferraty et P. Vieu : *Nonparametric functional data analysis*. Springer Series in Statistics. Springer, New York, 2006. Theory and practice.

F. Ferraty et P. Vieu : Richesse et complexité des données fonctionnelles. *Revue Modulad*, 43:25–43, 2011.

A. Goia et P. Vieu : Special issue on models and methods for high or infinite dimensional spaces. *J. Multivariate Anal.*, 146:1–352, 2016.

W. González Manteiga et P. Vieu : Introduction to the special issue on statistics for functional data. *Comput. Statist. Data Anal.*, 51(10):4788–4792, 2007.

P. J. Green et B. W. Silverman : *Nonparametric regression and generalized linear models*, vol. 58 de *Monographs on Statistics and Applied Probability*. Chapman & Hall, London, 1994. A roughness penalty approach.

P. Hall : Principal component analysis for functional data : methodology, theory, and discussion. *In The Oxford handbook of functional data analysis*, p. 210–234. Oxford Univ. Press, Oxford, 2011.

P. Hall et M. Hosseini-Nasab : On properties of functional principal components analysis. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 68(1):109–126, 2006.

W. Härdle, G. Kerkyacharian, D. Picard et A. Tsybakov : *Wavelets, approximation, and statistical applications*, vol. 129 de *Lecture Notes in Statistics*. Springer-Verlag, New York, 1998.

F. Hirsch et G. Lacombe : *Eléments d'analyse fonctionnelle : cours et exercices*. Dunod, 1997.

J. Hoffmann-Jørgensen, L. A. Shepp et R. M. Dudley : On the lower tail of Gaussian seminorms. *Ann. Probab.*, 7(2):319–342, 1979.

I. A. Ibragimov et Y. A. Rozanov : *Gaussian random processes*, vol. 9 de *Applications of Mathematics*. Springer-Verlag, New York-Berlin, 1978. Translated from the Russian by A. B. Aries.

S.-Y. Lee, W. Zhang et X.-Y. Song : Estimating the covariance function with functional data. *British J. Math. Statist. Psych.*, 55(2):247–261, 2002.

W. V. Li et Q.-M. Shao : Gaussian processes : inequalities, small ball probabilities and applications. *In Stochastic processes : theory and methods*, vol. 19 de *Handbook of Statist.*, p. 533–597. North-Holland, Amsterdam, 2001.

M. A. Lifshits : On the lower tail probabilities of some random series. *Ann. Probab.*, 25 (1):424–442, 1997.

I. B. MacNeill : Properties of sequences of partial sums of polynomial regression residuals with applications to tests for change of regression at unknown times. *Ann. Statist.*, 6 (2):422–433, 1978.

A. Mas : Lower bound in regression for functional data by representation of small ball probabilities. *Electron. J. Stat.*, 6:1745–1778, 2012.

P. Massart : *Concentration inequalities and model selection*, vol. 1896 de *Lecture Notes in Mathematics*. Springer, Berlin, 2007. Lectures from the 33rd Summer School on Probability Theory held in Saint-Flour, July 6–23, 2003, With a foreword by Jean Picard.

J. O. Ramsay et C. J. Dalzell : Some tools for functional data analysis. *J. Roy. Statist. Soc. Ser. B*, 53(3):539–572, 1991. With discussion and a reply by the authors.

J. O. Ramsay et B. W. Silverman : *Applied functional data analysis*. Springer Series in Statistics. Springer-Verlag, New York, 2002. Methods and case studies.

J. O. Ramsay et B. W. Silverman : *Functional data analysis*. Springer Series in Statistics. Springer, New York, second édn, 2005.

J. O. Ramsay, G. Hooker et S. Graves : *Functional data analysis with R and MATLAB*. Springer Science & Business Media, 2009.

C. H. Reinsch : Smoothing by spline functions. I, II. *Numer. Math.*, 10:177–183 ; ibid. 16 (1970/71), 451–454, 1967.

J. A. Rice et B. W. Silverman : Estimating the mean and covariance structure nonparametrically when the data are curves. *J. Roy. Statist. Soc. Ser. B*, 53(1):233–243, 1991.

A. Roche : *Modélisation statistique pour données fonctionnelles : approches non asymptotiques et méthodes adaptatives*. Thèse de doctorat, Université Montpellier II, 2014.

M. Valderrama : Introduction to the special issue on modelling functional data in practice. *Comput. Statist.*, 22(3):331–334, 2007.