Projet national **DVDC**

***D**urée de **V**ie **D**es **C**haussées*

RAPPORT DE RECHERCHE / LIVRABLE

Modèles d'évolution des chaussées

Auteur(s) / Organisme :

Tristan Lorino – Ifsttar

Thème de rattachement :

Thème 3

« Évaluation de la durée de vie résiduelle des chaussées »

DVDC/R/001

LC/17/DVDC/22

29/11/17

Site internet : www.dvdc.fr
Plateforme collaborative : www.omnispace.fr/dvdc
Président : Roger FRANK
Directeurs techniques : Sébastien BURLON et Philippe REIFFSTECK
Gestion administrative et financière : IREX (www.irex.asso.fr), 9 rue de Berri 75008 PARIS, contact irex.asso.fr

Sommaire

Introduction

Ce document est une première version d'un rapport plus ample, destiné à la revue des différents modèles capables de décrire, voire d'expliquer l'évolution des caractéristiques de chaussée. Il se compose en trois parties : la première traite de la modélisation statistique (compromis biais - variance notamment), la deuxième tend à modéliser l'évolution croissante d'une caractéristique continue (par exemple un indicateur de dégradation de surface), la troisième porte sur les modèles de classification de données discrètes (par exemple les différents niveaux d'entretien).

Les différents modèles sont succinctement présentés et mis en œuvre sur deux jeux de données : un jeu de données réelles provenant du Ministère des transports de Quebec et concernant la fissuration longitudinale (par fatigue), un jeu de données simulées de deux dégradations et de l'entretien programmé à partir des seuils atteints par ces deux dégradations.

En fin de document sont fournies toutes les procédures informatiques déployées dans le cadre de cette étude.

Première partie

Modélisation statistique et validation numérique

1. Modélisation statistique

Une base de données routières comportent généralement un grande nombre d'observations, correspondant à des relevés effectués à plusieurs reprises sur des sections de route de même longueur. Ces relevés correspondent aux mesures prises par un ou plusieurs indicateurs d'état de la chaussée, que cet état soit de type structurel (fissuration par exemple) ou surfacique (niveau d'adhérence). Il est possible que d'autres informations portant sur les sections soient disponibles dans la base de données : type de revêtement, nombre de couches, épaisseurs de ces couches, etc. Cependant il s'avère que ces informations concernant la typologie des chaussées sont souvent parcellaires ou erronées. Aussi nous intéressons-nous ici aux modèles susceptibles de rendre compte de l'évolution d'un indicateur sans la prise en compte de variables explicatives autres que le temps (qui peut être l'âge de la structure de chaussée).

Modélisation continue ou discrète

Les modèles d'évolution des caractéristiques de chaussées peuvent être de deux types différents, selon la nature de l'indicateur étudié : soit de type continu, si l'indicateur est de nature quantitative (par exemple s'il est exprimé en pourcentages), soit de type discret si l'indicateur est de nature qualitative (par exemple s'il se présente sous forme de classes ou de notes). Ces deux types de modèles seront tour à tour étudiés.

Modélisation pour l'inférence ou la prédiction

La formulation générique d'un modèle d'évolution est de la forme :

$$Y = f(X) + \epsilon \tag{1.1}$$

où Y est l'indicateur d'état de la chaussée (variable à expliquer ou à prédire), X est le vecteur des variables explicatives (contenant au moins le temps) et ϵ est un terme d'erreur de moyenne nulle et indépendant de X. Le modèle conduit à estimer f, c'est-à-dire à fournir des valeurs $\hat{f}(X) = \hat{Y}$ qui seront plus ou moins proches des valeurs réelles Y.

L'**inférence** consiste à comprendre le lien qui unit les variables explicatives à la variable à expliquer, à tester l'influence de ces variables explicatives sur la variable réponse ou encore à déterminer le sens de variation de Y en fonction de X, de manière à pouvoir extrapoler les résultats obtenus sur un échantillon donné (l'échantillon étudié) à l'ensemble de la population de laquelle il est issu. Ici, la fonction f ne peut être fournie sous la forme d'une « boîte noire », et elle nécessite de revêtir une forme paramétrique ou semi-paramétrique – les paramètres étant l'articulation entre variables à expliquer et variable explicative.

La **prédiction** consiste à déterminer la valeur de la variable réponse pour une valeur donnée des variables explicatives. Dans cette optique, la connaissance de la fonction f n'est pas indispensable, et celle-ci peut donc être considérée comme une « boîte noire ». Seule compte la justesse de l'ajustement, c'est-à-dire la proximité entre une valeur Y et sa prédiction $\hat{Y}$. L'erreur qui est commise en estimant f par $\hat{f}$ est une erreur réductible : dans le cas d'une méthode d'apprentissage par exemple,

Critère d'adéquation

Par la suite, le critère retenu pour comparer l'adéquation des modèles – qui fournissent les estimations notées $\hat{Y}_i$ – aux données (les Y_i) est la racine carrée de l'erreur quadratique

moyenne (EQM), qui est définie par :

$$\text{EQM} = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{Y}_i - Y_i\right)^2 \tag{1.2}$$

Dans le cas des modèles d'apprentissage, la procédure de leur ajustement aux données est la suivante [2] :

- phase d'apprentissage : sélection d'un échantillon d'apprentissage (*train dataset*) sur lequel le modèle est ajusté et conduisant au calcul d'une EQM dite d'apprentissage ;
- phase facultative de validation : sélection d'un échantillon de validation (*validation dataset*), permettant si nécessaire d'ajuster certains paramètres du modèle ;
- phase de test : sélection d'un échantillon de test (*test dataset*), sur lequel le modèle est ajusté et conduisant au calcul d'une EQM dite de test.

Compromis entre biais et variance

La moyenne (ou la valeur attendue, exprimée sous forme d'espérance) des carrés des écarts entre les valeurs réelles de Y et les valeurs prédites par le modèle $\hat{Y}$ vaut :

$$\begin{aligned}\mathbb{E}\left[(Y-\hat{Y})^2\right] &= \left(\mathbb{E}\left[\hat{f}(X)\right] - f(X)\right)^2 + \mathbb{E}\left[\left(\hat{f}(X) - \mathbb{E}\left[\hat{f}(X)\right]\right)^2\right] + \mathbb{V}(\epsilon) \\ &= \text{biais}^2_{\hat{f}} + \text{variance}_{\hat{f}} + \text{variance}_{\epsilon}\end{aligned} \tag{1.3}$$

La variance du terme d'erreur est dite « irréductible » car elle ne dépend pas de la justesse de l'estimation $\hat{f}$. En supposant même que $\hat{f}$ soit une estimation parfaite de f, et que donc la variable réponse estimée soit de la forme $\hat{Y} = f(X)+\epsilon$, la prédiction comporterait tout de même une composante d'erreur : il y aurait toujours une partie de la variabilité de Y qui ne pourrait être expliquée par X.

D'après l'éq. (1.3), la justesse de l'estimation $\hat{f}$ dépend de deux quantités positives ou nulles, et elle est optimale lorsque ces deux quantités sont minimales. La première de ces quantités est le **biais**, qui correspond à l'erreur d'approximation d'un problème réel par un modèle simplificateur. Plus le modèle sera complexe, plus il sera flexible et donc plus son adéquation aux données sera grande. La contrepartie d'une telle précision d'ajustement est le caractère de dépendance de l'ajustement aux données étudiées, et conséquemment d'une piètre capacité d'inférence : pour un autre échantillon, le modèle diffèrera grandement, d'une manière qu'il n'est pas possible de prévoir. La seconde quantité qui peut grever la précision du modèle est la **variance** associée à $\hat{f}$: il s'agit de la variabilité de $\hat{f}$ liée aux fluctuations aléatoires de l'échantillon d'apprentissage. Idéalement, même si l'estimation de f dépend de l'échantillon d'apprentissage, si cette estimation est juste elle ne devrait guère varier une fois appliquée à un autre échantillon d'apprentissage.

Un compromis doit donc s'opérer entre biais et variance : plus le modèle sera flexible (il aura un degré de liberté élevé), moins il sera biaisé, mais plus sa variance sera grande. À titre d'illustration, considérons la partie gauche de la figure 1.1 : des données, représentées par des cercles noirs, ont été simulées à partir d'une fonction f connue, représentée par la courbe en noir. Trois ajustements ont été faits : l'un par régression linéaire (courbe orange), les deux

autres par courbe spline (en bleu avec une faible flexibilité, en vert avec une forte flexibilité). Nous remarquons que la courbe verte passe au plus près des points, mais pas au plus près de la fonction f : en raison de sa forte flexibilité, cet ajustement sera fortement dépendant de l'échantillon d'apprentissage.

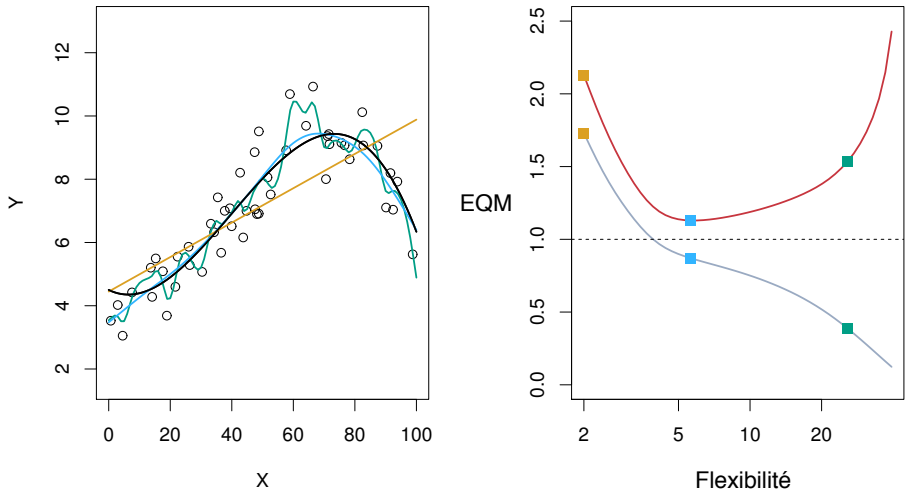

Figure 1.1 – *Gauche : données simulées à partir de f, en noir. Trois estimations de f : droite de régression linéaire (orange), deux courbes splines (bleue et verte). Droite : EQM d'apprentissage (gris), EQM de test (rouge) et minimum possible de l'EQM de test (droite pointillée). Les carrés représentent les EQM pour les trois ajustements de gauche. Figure tirée de l'ouvrage « An Introduction to Statistical Learning, with applications in R » (Springer, 2015), avec l'autorisation des auteurs G. James, D. Witten, T. Hastie et R. Tibshirani.*

Sur la partie droite de la figure sont représentées les EQM d'apprentissage (courbe grise), de test (courbe rouge) ainsi que l'EQM minimale de test (droite horizontale pointillée, correspondant à $\mathbb{V}(\epsilon)$), en fonction du degré de flexibilité du modèle (qui est en réalité le nombre de degrés de liberté).

L'ajustement linéaire à 2 degrés de liberté conduit à des EQM d'apprentissage et de test élevés : cet ajustement, dont l'adéquation aux données d'apprentissage est faible, s'avère également insatisfaisant pour l'échantillon de test. La régression linéaire souffre de son manque de flexibilité pour ajuster une courbe qui n'est clairement pas une droite.

L'ajustement par une courbe spline à haute flexibilité (courbe verte) se caractérise par une excellente adéquation aux points (faible EQM d'apprentissage), mais pas à la courbe réelle f, d'où une EQM élevée en phase de test : il s'agit souvent, dans le cas d'une flexibilité trop importante, d'un problème de sur-ajustement (ou sur-paramétrisation) des données d'apprentissage, qui conduit à un modèle ne rendant pas fidèlement compte du processus que nous cherchons à modéliser.

2. Validation numérique

Outil logiciel

La mise en œuvre des modèles d'évolution des caractéristiques de chaussées est menée avec le logiciel R car d'une part ce logiciel est gratuit et multi-plateforme, d'autre part il bénéficie d'un grand nombre de librairies qui donnent accès à tous les modèles présentés. Tous les codes de programmation sont fournis en annexe.

Jeux de données

Les modèles sont illustrés par leur application à deux jeux de données.

Le premier provient d'une base de suivi de 45 sections-tests, qui constituent des chaussées bitumineuses (souples ou épaisses). Ces données sont issues du Ministère des Transport du Quebec. L'indicateur d'état de la chaussée (exprimé en pourcentages) est ici la fissuration de fatigue. Chaque section a été inspectée entre 4 et 12 fois. La figure 2.1 illustre ces données.

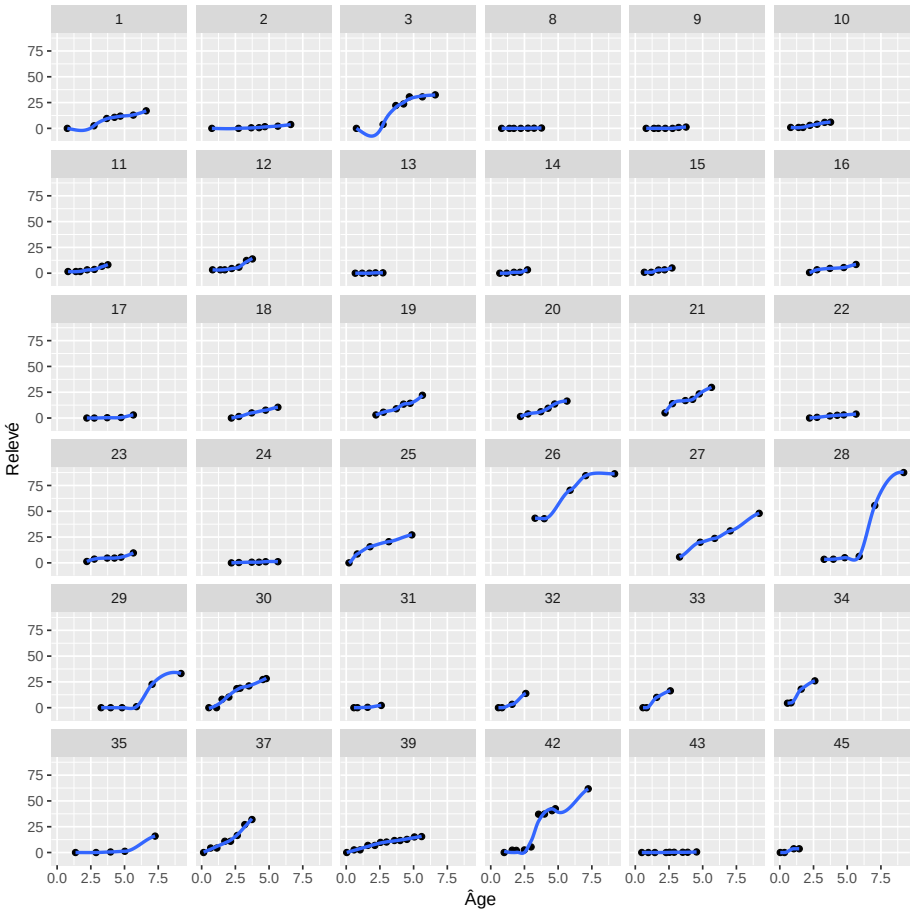

Figure 2.1 – *Fissuration de fatigue pour les 41 sections (points), avec courbe de lissage (en bleu).*

Le second jeu de données consistent en simulations de deux niveaux de dégradations, notées D1 et D2, et en la mise en place de travaux d'entretien de trois degrés différents, suivant les valeurs croisées de D1 et D2 comme indiqué dans le tableau 2.1. À titre d'exemple, si la dégradation D1 est de 60 % et si D2 est de 35 %, alors l'entretien réalisé est de type « b ».

Les valeurs des dégradations D1 et D2 sont issues d'une loi d'équation :

$$D = 1 - \exp\Big(- (\frac{t}{\lambda})^k \Big) \times 100$$

où t est l'âge de la section au moment de relever le niveau de dégradation D et $(\lambda,\, k)$ est un couple de paramètres fixés.

Tableau 2.1 – *Simulation de travaux de maintenance à trois niveaux suivant les valeurs de deux dégradations : nature des travaux (gauche) et effectifs (droite).*

	D1 (%)		
D2 (%)	0-50	50-70	70-100
0-30	a	a	b
30-50	a	b	c
50-100	b	c	c

	D1 (%)		
D2 (%)	0-50	50-70	70-100
0-30	n_{11}	n_{12}	n_{13}
30-50	n_{21}	n_{22}	n_{23}
50-100	n_{31}	n_{32}	n_{33}

Deuxième partie

Modèles pour données continues

3. Modèle paramétrique non linéaire mixte

Le modèle paramétrique non linéaire mixte choisi ici est le modèle logistique mixte. Le terme « mixte » signifie que les paramètres du modèle sont de deux nature : l'une fixe qui correspond à des valeurs moyennes des paramètres (c.-à-d. des valeurs pour l'ensemble des sections étudiées), l'autre aléatoire pour affecter à chaque section une variabilité qui lui est propre et qui découle de la répétition des mesures effectuées sur elle (cette répétition entraînant une corrélation entre mes mesures que le modèle statistique doit prendre en compte).

En l'absence de paramètre aléatoire, le modèle logistique (à effets uniquement fixes) s'écrit :

$$y = \frac{\theta_1}{1 + \exp\left(-\frac{x-\theta_2}{\theta_3}\right)} + \epsilon \tag{3.1}$$

où, comme l'illustre la figure 3.1 :

- y est la valeur de l'indicateur lors du relevé x ;
- θ_1 est l'asymptote horizontale (généralement fixée à 100 % pour un indicateur exprimé en pourcentages) ;
- θ_2 est la valeur de x pour laquelle y atteint la moitié de la valeur de l'asymptote ;
- θ_3 est l'intervalle entre θ_2 et la valeur de x pour laquelle y atteint les 3/4 de la valeur asymptotique ;
- ϵ est le terme d'erreur supposé supposé suivre une loi normale centrée.

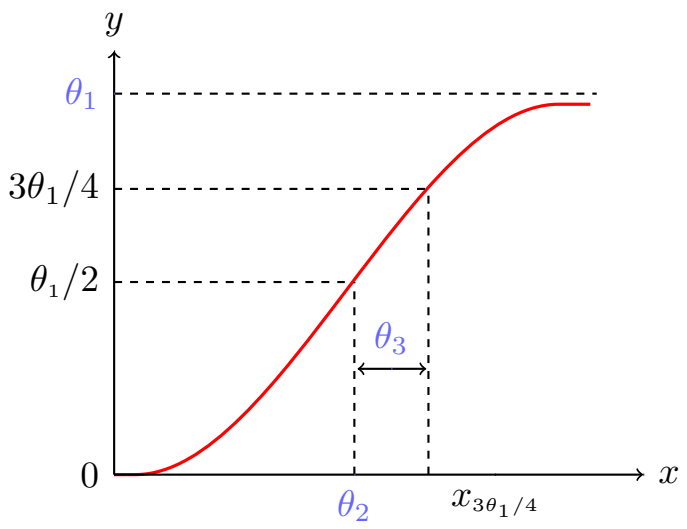

Figure 3.1 – *Régression logistique (courbe rouge) et paramètres associés (en bleu).*

Notons que si l'on dispose de variables explicatives, leur effet peut être introduit au travers des paramètres à effets fixes θ_1, θ_2 et θ_3.

L'extension du modèle logistique par ajout d'une variabilité intra-section, qui devient alors modèle logistique mixte, s'écrit sous la forme :

$$y_{ij} = \frac{\theta_1 + \theta_{1i}}{1 + \exp\left(-\frac{x-\theta_2+\theta_{2i}}{\theta_3+\theta_{3i}}\right)} + \epsilon_{ij} \tag{3.2}$$

où, en plus des paramètres précédemment définis :

- θ_{i1}, θ_{i2} et θ_{i3} sont respectivement les paramètres aléatoires associés à la section i ;
- ϵ_{ij} est le terme d'erreur supposé supposé suivre une loi normale centrée.

La figure 3.2 illustre les courbes d'évolution de la fissuration pour la population générale (courbe noire) et l'évolution propre à chaque section (courbe bleue). Les RMSE sont respectivement égaux à 12 % et 1 % dans le cas des régressions moyenne (population générale) et individuelle (propre à chaque section).

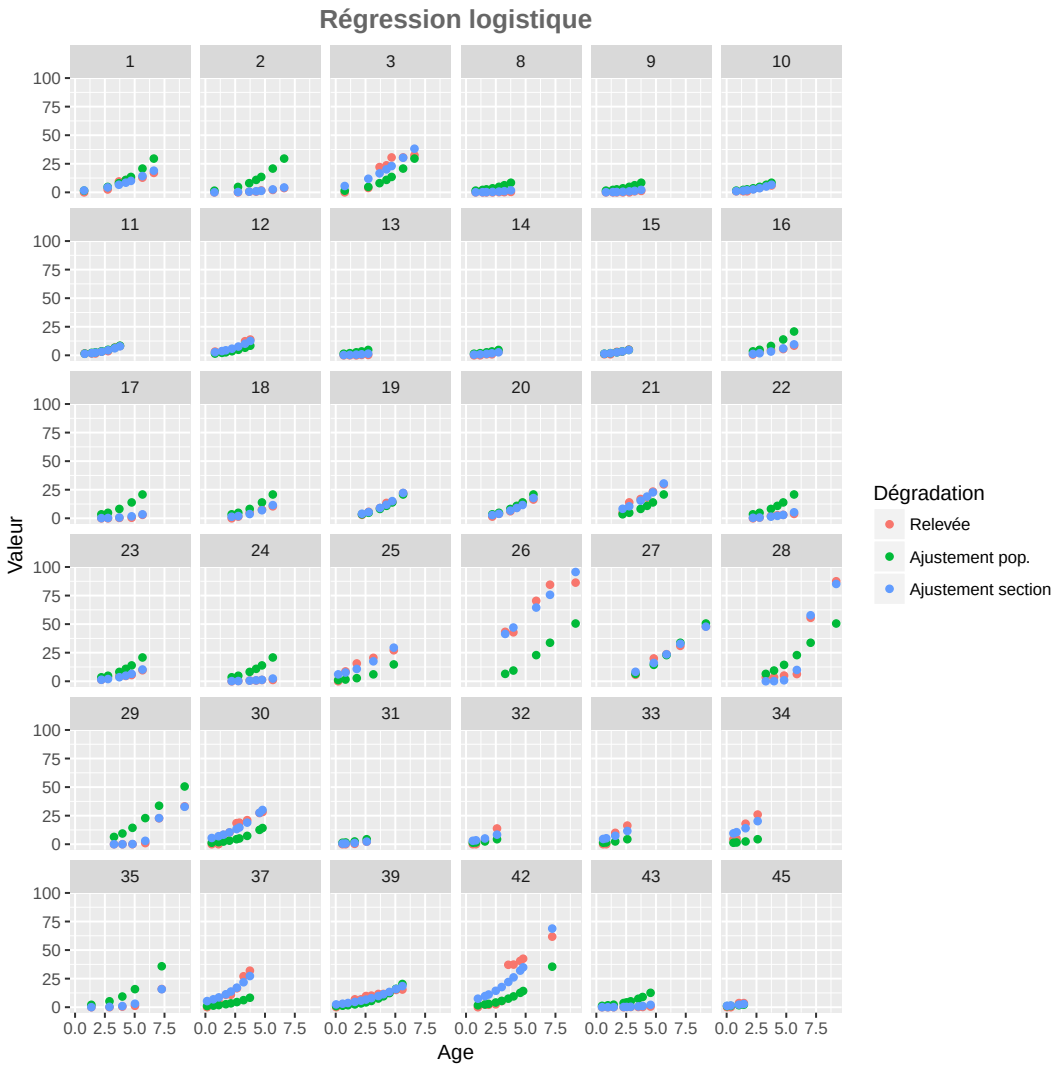

Figure 3.2 – *Régression logistique mixte pour chaque section (points bleus) et pour la population moyenne (points verts).*

4. SVM

Les machines à vecteurs supports (*Support Vector Machine*) sont des estimateurs non linéaires de fonctions, dont les fondements reposent sur la théorie statistique de l'apprentissage [1, 7, 5]. Supposons que nous cherchions à estimer une fonction $y = f(x)$ à partir d'un nombre limité d'échantillons $S = \{(x_1,y_1), \ldots, (x_n,y_n)\} \in \mathbb{R}^d \times \mathbb{R}$ et d'hypothèses faites sur les propriétés de $\mathcal{H}$, espace de Hilbert muni d'un produit scalaire. Nous définissons le risque lié à l'estimation par :

$$R(h) = \int L\big(y, h(x)\big)\, \mathrm{d}P(x, y) \tag{4.1}$$

où $L(.,.)$ est une fonction de coût. Le meilleur estimateur $\hat{f}(x) = h^*(x) \in \mathcal{H}$ est celui qui minimise (4.1). La distribution jointe $P(x, y)$ étant inconnue *a priori*, il faut approximer (4.1) : la théorie statistique de l'apprentissage fournit alors des résultats de la forme :

$$R(h) \leqslant \sum_{(x_i,\, y_i)\in S} L\big(y_i, h(x_i)\big) + Q(n, h, \delta) \tag{4.2}$$

où $Q(n, h, \delta)$ consiste en un terme de confiance qui est fonction du nombre d'observations, de la capacité de la sous-classe $\mathcal{H}_k \subset \mathcal{H}$ qui contient l'hypothèse h et de la probabilité $1 - \delta$ avec laquelle l'inégalité (4.2) est valide. La procédure de minimisation du risque structurel procède par partage de $\mathcal{H}$ en sous-classes emboîtées $\mathcal{H}_k$ ($\mathcal{H}_1 \subset \mathcal{H}_2 \subset \cdots \subset \mathcal{H}_M$) de capacités croissantes, ce qui permet de chercher l'hypothèse $h^* \in \mathcal{H}$ qui minimise (4.2).

Dans le cadre des SVM, l'hyperplan (w^*, b^*) optimal noté $\mathcal{H}_{\text{SVM}}$ est obtenu via la résolution de :

$$h^*(w^*, b^*, x) = \min_{(w,\, b)} \; C \cdot \sum_{(x_i,\, y_i)\in S} L\big(y_i, h(w, b, x_i)\big) + \frac{1}{2}\|w\|^2 \tag{4.3}$$

où C est une constante à choisir.

Dans le cas de la régression, des fonctions de coût linéaire ou quadratique L_ϵ sont utilisées, ne prenant en compte que les déviations $|y_i - h(x_i)| > \epsilon$, où ϵ devient un autre paramètre à fixer. L'équation (4.3) peut être réécrite comme un problème dual de Lagrange, et cette reformulation se ramène à un problème de programmation quadratique, de solution unique et pour lequel des méthodes de résolution efficaces existent. La solution de l'éq. (4.3) prend la forme :

$$h^*(x) = \hat{f}(x) = \sum_{(x_i,\, y_i)\in S} \alpha_i^* K(x_i, x) + b^* \tag{4.4}$$

où les α_i^* sont les multiplicateurs de Lagrange impliqués dans la résolution du problème dual de Lagrange de (4.3) et $K(.,.)$ est un noyau. Quand $\epsilon > 0$, un nombre important des α_i^* sont égaux à zéro, et (4.4) fournit alors une représentation creuse. Les x_i correspondant aux $\alpha_i^* \neq 0$ sont appelés *vecteurs de support*.

La régression de type SVM conduit à un RMSE de 11 % et la courbe est présentée à la figure 4.1.

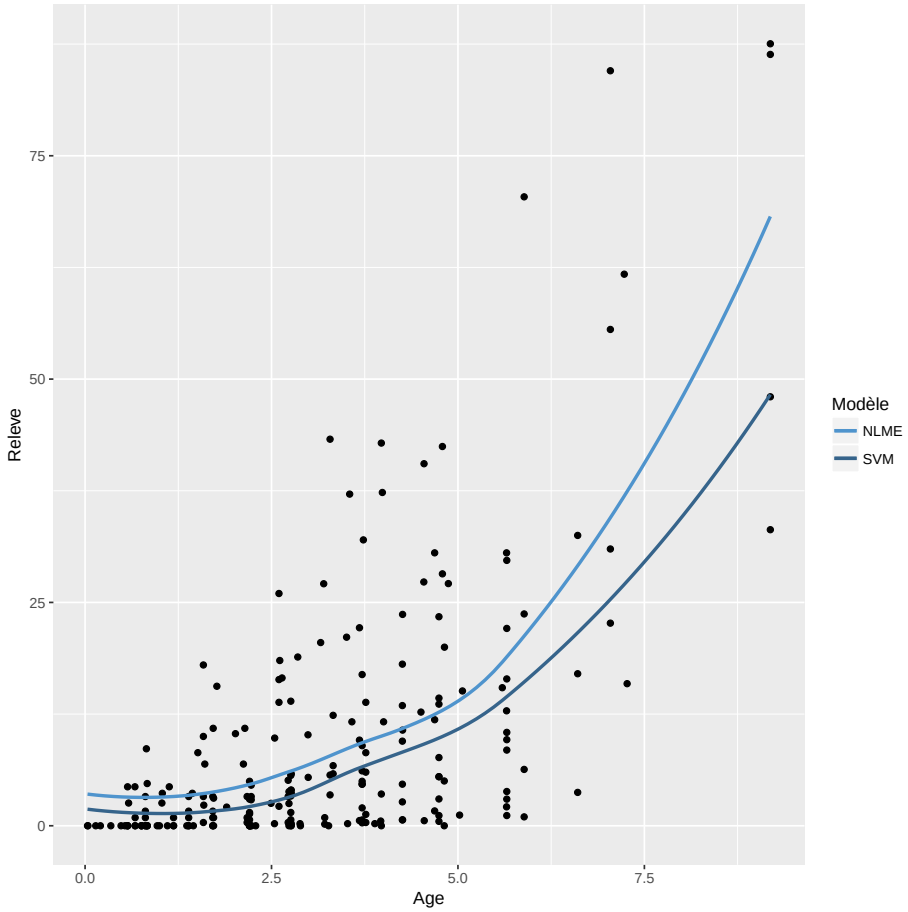

Figure 4.1 – *Régression logistique mixte et régression de type SVM.*

Il est possible de faire appel à des noyaux différents, comme par exemple des noyaux linéaire et polynomial (fig. 4.2), pour lesquels les RMSE sont respectivement de 12 % et 11 %. cependant, dans tous les cas, et à l'inverse du modèle de régression non linéaire, il n'existe pas de calcul simple de l'équation de la courbe SVM, ce qui en limite l'intérêt.

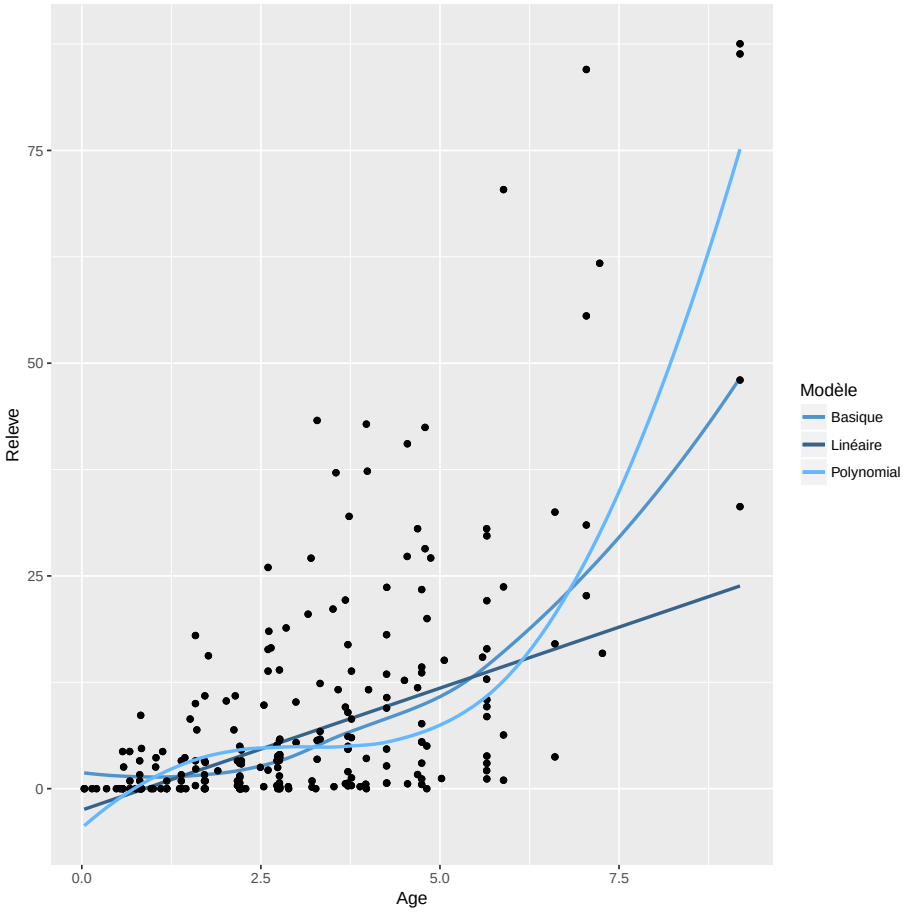

Figure 4.2 – *Régression de type SVM avec différents noyaux.*

Troisième partie

Modèles pour données discrètes

5. Classification par apprentissage

Tableau 5.1 – *Tableau croisé des effectifs.*

		Entretien observé		
		a	b	c
Entretien prédit	a	n_{11}	n_{12}	n_{13}
	b	n_{21}	n_{22}	n_{23}
	c	n_{31}	n_{32}	n_{33}

La qualité de la répartition entre les différentes classes se mesure grâce à deux indicateurs [4]. Le premier est **l'indicateur de précision globale**, et il est par définition égal au nombre de données concordantes, soit :

$$\frac{1}{n}\sum_{i=1}^{3} n_{ii}, \quad \text{avec} \quad n = \sum_{i,j} n_{ij}$$

Cet indicateur présente deux inconvénients : d'une part il ne tient pas compte de la fréquence naturelle des classes, d'autre part il traite à égalité toutes les classes. Aussi est-il employé comme indicateur du minimum de précision : une répartition aléatoire entre k classes étant égale à $1/K$, si l'indicateur de précision globale est inférieur à cette quantité, alors la concordance doit être invalidée.

Le second indicateur de précision est le **coefficient kappa**, défini par :

$$\kappa = \frac{\mathbb{P}_o - \mathbb{P}_p}{1 - \mathbb{P}_p}$$

où $\mathbb{P}_o$ et $\mathbb{P}_p$ sont respectivement les fréquences observées et prédites. En reprenant l'exemple 5.1, on obtient :

$$\mathbb{P}_o = \frac{1}{n}\sum_{i=1}^{3} n_{ii}$$

et

$$\mathbb{P}_p = \frac{1}{n^2}\sum_{i=1}^{3} n_{i.} \times n_{.i}$$

où $n_{i.} = \sum_j n_{ij}$ et $n_{.i} = \sum_i n_{ij}$. En effet, si la répartition était purement aléatoire, alors la probabilité qu'observation et prédiction soit dans la même classe i serait égale à

$$\frac{n_{i.}}{n} \times \frac{n_{.i}}{n}.$$

On considère généralement qu'un coefficient κ supérieur à $0{,}60$ (resp. $0{,}80$) correspond à un accord satisfaisant (resp. excellent) entre valeurs observées et valeurs prédites.

Les tableaux 5.2 et 5.3 fournissent les valeurs de l'indicateur de précision globale et du κ pour des nombres de sections et de relevés par section différent, ainsi que deux différents taux d'apprentissage. Pour chaque configuration des effectifs, 100 itérations ont été réalisées. L'algorithme testé ici est celui de l'analyse discriminante linéaire (`lda`).

Les résultats sont similaires et très satisfaisants, quels que soient le taux d'entrainement et le nombre de relevés par section.

Tableau 5.2 – *Mesure de la précision pour un taux d'entraînement de 50 % de l'effectif total.*

		Précision globale			
Sections	Relevés	Valeur	Inf.	Sup.	Indice κ
500	3	0,929	0,908	0,946	0,806
1 000	3	0,928	0,914	0,941	0,803
5 000	3	0,930	0,924	0,935	0,808
10 000	3	0,929	0,925	0,933	0,806
500	6	0,930	0,916	0,942	0,807
1 000	6	0,928	0,918	0,937	0,803
5 000	6	0,929	0,925	0,933	0,807
10 000	6	0,929	0,926	0,932	0,806
500	9	0,929	0,917	0,939	0,806
1 000	9	0,929	0,921	0,936	0,807
5 000	9	0,929	0,926	0,932	0,806
10 000	9	0,929	0,927	0,932	0,807

Tableau 5.3 – *Mesure de la précision pour un taux d'entraînement de 80 % de l'effectif total.*

		Précision globale			
Sections	Relevés	Valeur	Inf.	Sup.	Indice κ
500	3	0,928	0,893	0,955	0,803
1 000	3	0,929	0,906	0,949	0,806
5 000	3	0,929	0,919	0,938	0,806
10 000	3	0,929	0,922	0,936	0,807
500	6	0,929	0,905	0,948	0,806
1 000	6	0,928	0,912	0,942	0,804
5 000	6	0,930	0,923	0,936	0,808
10 000	6	0,929	0,925	0,934	0,807
500	9	0,930	0,911	0,945	0,807
1 000	9	0,929	0,917	0,941	0,807
5 000	9	0,930	0,924	0,935	0,808
10 000	9	0,929	0,926	0,933	0,807

La librairie `caret` [3] possède de nombreux outils pour la classification. Nous comparons ici 8 méthodes [1], que nous testons sur un jeu de données de 10 000 sections comportant chacune 6 observations :

- `lda` : analyse discriminante linéaire ;
- `rpart` : partitionnement récursif ;
- `rda` : analyse discriminante régularisée ;
- `fda` : analyse discriminante flexible ;
- `knn` : partitionnement suivant les k plus proches voisins ;
- `svm` : machine à support vecteur ;
- `gbm` : arbres boostés ;
- `rf` : forêt aléatoire.

1. Ces méhodes seront détaillées dans la prochaine du présent rapport.

Tableau 5.4 – *Mesure de la précision pour différentes méthodes (croisement valeurs observées - valeurs prédites).*

	Précision globale			Indice κ		
Méthode	Min	Moy.	Max	Min	Moy.	Max
lda	0,903	0,912	0,921	0,737	0,760	0,785
rpart	0,934	0,958	0,983	0,814	0,880	0,952
rda	0,944	0,952	0,958	0,848	0,869	0,883
fda	0,966	0,974	0,979	0,906	0,927	0,941
cart	0,933	0,959	0,982	0,812	0,884	0,948
knn	0,995	0,997	0,999	0,987	0,992	0,997
svm	0,995	0,997	0,999	0,987	0,993	0,997
gbm	0,999	1,000	1,000	0,999	0,999	1,000
rf	0,999	1,000	1,000	0,997	0,999	1,000

Tableau 5.5 – *Tableau croisé des effectifs pour lda (gauche), gbm (milieu) et rf (droite).*

		Entretien observé			**Entretien observé**			**Entretien observé**		
		a	b	c	a	b	c	a	b	c
Entretien	a	9 104	45	0	9 953	0	0	9 953	0	0
prédit	b	377	618	256	0	960	0	0	960	1
	c	72	297	1 229	0	0	1 485	0	0	1 484

6. Arbre de décision

Un arbre de décision est un algorithme d'apprentissage automatique qui ne fait aucune hypothèse sur la relation entre la variable à expliquer et les variables explicatives. À la fin du processus décisionnel, un ensemble de règles reliant le résultat aux variables explicatives est dérivé de l'arbre. L'algorithme commence par définir un nœud racine, qui contient toutes les données d'apprentissage, et le divise en deux nouveaux nœuds basés sur la variable explicative la plus importante, c'est-à-dire celle qui sépare le résultat en deux groupes [6]. Puis, pour chaque nouveau nœud, une division est testée, toujours sur la base de la variable explicative qui divise le mieux ce nœud particulier (cette variable explicative n'étant pas forcément celle du nœud précédent). le processus continue jusqu'à ce que un critère d'arrêt d'arrêt soit rencontré – par défaut, le critère retenu est la nécessité que chaque nœud ait au moins 20 observations, sinon une nouvelle division n'est pas tentée). Un arbre de décision a tendance à sur-ajuster les données : il se caractérise donc par de très bonnes performances pour les données d'entraînement, mais de faibles performance sur l'ensemble des données de test.

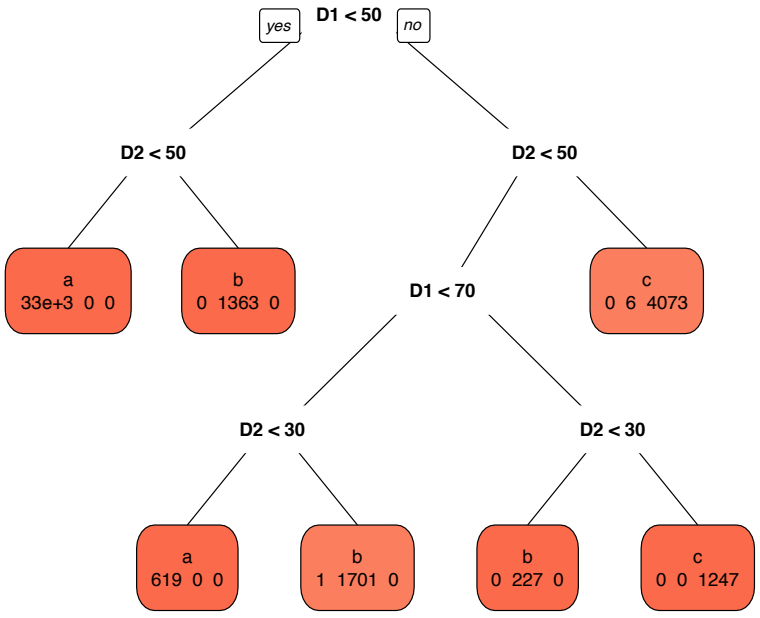

Figure 6.1 – *Arbre de décision optimal.*

Pour remédier à cela, l'algorithme repose sur la validation croisée : l'arbre réalise en interne une validation croisée de 10 cycles ; pour lesquels 90 % des données d'entraînement sont utilisées pour construire le modèle, et les 10 % restants -- même s'ils font partie des données d'entraînement -- servent uniquement à prédire et évaluer les résultats. En d'autres termes, ces 10 % constituent momentanément un ensemble de test. Après les 10 cycles, 10 échantillons différents de données sont utilisés pour tester le modèle, ce qui signifie que chaque observation

est utilisée, non seulement pour entraîner, mais aussi pour tester le modèle. Une mesure d'erreur est quantifiée à chaque apparition d'un nœud, qui permet de déterminer le nombre optimal de divisions. Cette erreur s'appelle l'erreur de validation croisée (*erreur-x*) et, dans le logiciel R, le paramètre de complexité.

Les données étudiées ici consistent en un jeu de 60 000 observations, à raison de 6 observations par section. Les données d'entraînement constituent environ 70 % de l'effectif total, et sont donc au nombre de 42 001, tandis que les données de test représentent les 17 999 observations restantes.

La figure 6.1 illustre l'arbre de décision obtenu avec la librairie `rpart` de R. Les effectifs indiqués dans les rectangles correspondent à ceux présentés dans le tableau 2.1 p. 9 et se répartissent comme indiquée sur la figure 6.2.

On constate que la classification fournit d'excellents résultats, puisque 7 observations seulement sont mal classées.

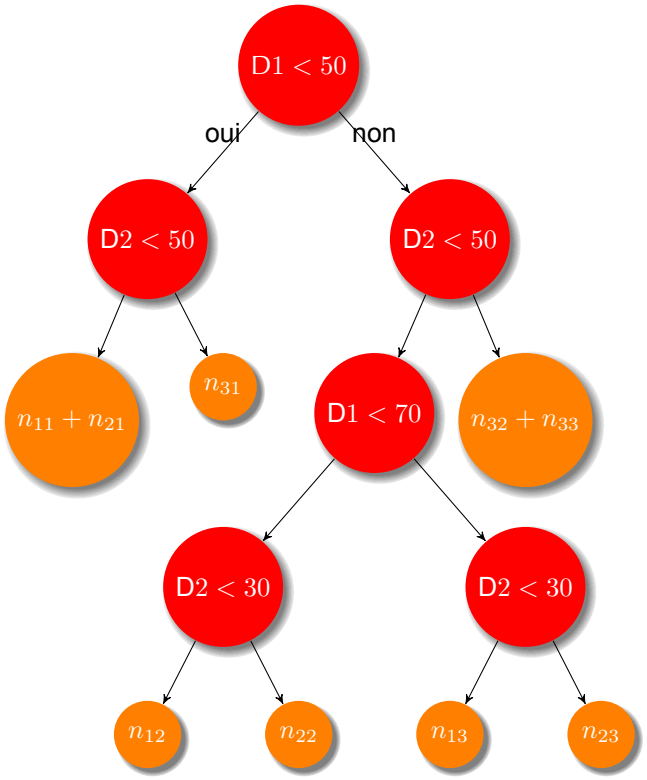

Figure 6.2 – *Arbre de décision donnant les effectifs.*

Le tableau 6.1 fournit le croisement entre les observations réelles du jeu des données de test, et les prédictions fournies par l'arbre pour ces mêmes observations. Un test du chi-deux d'indépendance entre les deux types d'entretien (observé et prédit) conduit à un rejet de l'hypothèse d'indépendance, ce qui était prévisible étant donné la très forte répartition des effectifs sur la diagonale du tableau – seules 3 observations sont mal classées.

Tableau 6.1 – *Tableau de contingence entre valeurs réelles de test et valeurs prédites pour l'arbre de décision.*

	Entretien observé (%)		
Entretien prédit	a	b	c
a	14 305	0	0
b	2	1 412	1
c	0	0	2 278

Notons que l'arbre de décision peut être élagué de manière à restreindre le nombre de nœuds (cf. fig. 6.3).

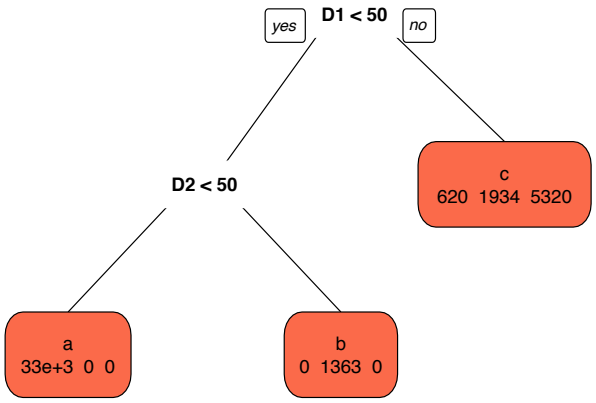

Figure 6.3 – *Arbre de décision à 2 divisions (3 nœuds).*

7. Classifieur bayésien

Nous supposons ici que l'appartenance d'une observation à une classe est une réalisation d'une variable aléatoire dont la valeur est le numéro de la classe (variable discrète). Les variables explicatives X sont supposées indépendantes conditionnellement à la variable à expliquer (appartenance à une classe). Nous scindons l'échantillon d'entraînement en m sous-ensembles contenant chacun tous les observations d'une même classe, puis nous entraînons un estimateur de la densité de chacun, soit $\mathbb{P}(X \mid \mathcal{C}_k)$. Nous connaissons par ailleurs les probabilités *a priori* $\mathbb{P}(\mathcal{C}_k)$. D'après le théorème de bayes, nous en déduisons la probabilité d'affectation d'une nouvelle observation à un classe :

$$\mathbb{P}(\mathcal{C}_k \mid X) = \frac{\mathbb{P}(X \mid \mathcal{C}_k) \times \mathbb{P}(\mathcal{C}_k)}{\sum_{k=1}^{K} \mathbb{P}(X \mid \mathcal{C}_k) \times \mathbb{P}(\mathcal{C}_k)}.$$

La règle de décision conduit à affecter l'observation X à la classe pour laquelle la probabilité *a posteriori* $\mathbb{P}(\mathcal{C}_k \mid X)$ est maximale.

Le tableau 7.1 fournit le croisement entre les observations réelles du jeu des données de test, et les prédictions fournies par le classifieur bayésien. Nous constatons que par rapport à d'autres méthodes de classification, les résultats sont ici nettement moins satisfaisants pour deux classes d'entretien sur les trois.

Tableau 7.1 – *Tableau de contingence entre valeurs réelles de test et valeurs prédites pour le classifieur bayésien.*

	Entretien observé (%)		
Entretien prédit	a	b	c
a	8 907	10	0
b	646	817	127
c	0	133	1 358

Quatrième partie

Conclusions et perspectives

Pour modéliser l'évolution d'une caractéristique (continue) de chaussée, le modèle non linéaire mixte s'avère un outil extrêmement performant : son formalisme acceptant tout type de fonction non linéaire, il est à même d'ajuster parfaitement tout type de donnée continue. De plus, il est possible d'introduire dans le modèle des variables explicatives du processus modélisé : c'est là une piste privilégiée pour d'ultérieurs ajouts au modèle statistique d'une composante mécanique (comportement des structures de chaussées).

Pour modéliser l'évolution d'une caractéristique (discrète) de chaussée, la classification par apprentissage et les arbres de décisions s'avèrent de précieux outils. D'autres types de modèles d'apprentissage seront abordés dans la version suivante de ce rapport – notamment les réseaux de neurones.

A. Code R

A.1 Préliminaires

```
# Définition de l'espace de travail
setwd("/Users/tristan/Documents/_Synchro/projets/Projets\ nationaux/DVDC/R")

# Fonction calculant le RMSE
rmse <- function(error)
{
  sqrt(mean(error^2))
}

# Importation des données
dato0=read.csv("/UsersDocuments/DVDC/R/data2.csv", header = TRUE, sep= ";", quote="\"", dec=".")

# exclusion des sections à 2 relevés
dato=dato0[!(dato0$Section %in% c(4,5,6,7,36,38,40,41,44)),]
dato$Releve=dato$Releve*100

# Graphique de l'évolution des sections
gdgraph=ggplot(dato,aes(Age,Releve)) + geom_point() + geom_smooth(se=FALSE)
+ facet_wrap(~Section, nrow=6) + scale_x_continuous('Âge') + scale_y_continuous('Relevé')
ggsave(gdgraph, file="donnesbrutes.pdf", width=8, height=8)
```

A.2 Modèle non linéaire mixte

```
# Chargement de la librairie pour le modèle non linéaire mixte
library(nlme)

# Déclaration des données groupées (par section)
mtqgrouped=groupedData(Releve~Age|Section,dato)

# régression non linéaire mixte
fit1.nlme=nlme(Releve~SSlogis(Age,Asym,a,b), data=mtqgrouped, fixed=Asym+a+b~1)
fit1.nlme

# Stockage des prédictions
pred=predict(fit1.nlme,level=0:1)

# Représentation graphique NLME population
donnesfilt=data.frame(Section=dato$Section,Age=dato$Age,ReleveP=dato$Releve,
PredP=pred[,2],PredS=pred[,3])
dutt.m <- melt(donnesfilt, id = c("Section","Age"))
head(dutt.m)

levels(dutt.m$variable)=c("Relevée","Ajustement pop.","Ajustement section")
names(dutt.m)=c("Section","Age","Dégradation","Valeur")
```

```
myuuplot=ggplot(dutt.m,aes(x = Age,y = Valeur, colour=Dégradation)) + geom_point()
+ facet_wrap(~Section) + ggtitle("Régression logistique")
+ theme(plot.title = element_text(color="#666666", face="bold", size=14, hjust=0.5))
ggsave(myuuplot, file="nlme-population.pdf", width=8, height=8)
```

A.3 SVM

```
# Chargement de la librairie SVM
library(e1071)

# Déclaration du modèle
model <- svm(Releve ~ Age , dato)
summary(model)

# Stockage des prédictions
predictedY <- predict(model, dato)

tuneResult <- tune(svm, Releve ~ Age,  data = dato,
                   ranges = list(epsilon = seq(0,1,0.1), cost = 2^(2:9))
)

# Résultats
print(tuneResult)
plot(tuneResult)

# Stockage des valeurs ajustées
timp=data.frame(Tps=dato$Age,Pred=predictedY)

# Graphique du SVM
ggplot(dato,aes(Age,Releve)) + geom_point()
+ geom_smooth(data=timp,aes(Tps,Pred,colour="VSM"),se=FALSE)

# Graphique conjoint modèle non linéaire mixte et SVM
bothgraph=ggplot(dato,aes(Age,Releve)) + geom_point()
+ geom_smooth(data=timp,aes(Tps,Pred,colour="SVM"),se=FALSE)
+ geom_smooth(data=pred2,aes(Age,Releve,colour="NLME"),se=FALSE)
+ scale_colour_manual(name="Modèle",values=c(NLME="steelblue3",SVM="steelblue4"))
ggsave(bothgraph, file="nlme-svm.pdf", width=8, height=8)

# Détermination du RMSE
errornlmef <- pred3$Releve - dato$Releve
errornlmer <- pred2$Releve - dato$Releve
errorsvm=predictedY-dato$Releve
RMSEnlmef <- rmse(errornlmef)
RMSEnlmef
RMSEnlmer <- rmse(errornlmer)
RMSEnlmer
RMSEsvm <- rmse(errorsvm)
RMSEsvm

## autres noyaux
model_lin <- svm(Releve ~ Age , dato, kernel="linear")
```

```
model_poly <- svm(Releve ~ Age , dato, kernel="polynomial")

predicted_lin <- predict(model_lin, dato)
predicted_poly <- predict(model_poly, dato)

rmse(predicted_lin-dato$Releve)
rmse(predicted_poly-dato$Releve)

bothygraph=ggplot(dato,aes(Age,Releve)) + geom_point()
+ geom_smooth(data=timp,aes(Tps,Pred,colour="Basique"),se=FALSE) + geom_smooth(data=timp_lin,aes(Tps,Pred,
+ scale_colour_manual(name="Modèle",
values=c(Basique="steelblue3",Linéaire="steelblue4",Polynomial="steelblue1"))
ggsave(bothygraph, file="diff-svm.pdf", width=8, height=8)
```

A.4 Simulations et classification par apprentissage

```
##############
### Fonction de création des données simulées
# n_act = nombre de sections
# rp : nombre de relevés par section
# tm = âge maximal de la chaussée au moment des relevés
# lambda_1 : paramètre d'échelle de la loi de Weibull pour la dégradation D1
# ka_1 : paramètre de forme de la loi de Weibull pour la dégradation D1
# lambda_2 : paramètre d'échelle de la loi de Weibull pour la dégradation D2
# ka_2 : paramètre de forme de la loi de Weibull pour la dégradation D2

follow_up = function(n=10,rp=5)
{
  tm=10
  lambda_1=9
  ka_1=5
  lambda_2=10
  ka_2=3.5

  # Data.frame avec identifiant de section et temps de relevé
  donnees=data.frame(Id=rep(1:n, rep(rp,n)),Tps=runif(n*rp,1,tm))

  # Bruit gaussien sur les paramètres de D1
  donnees[,3]=rep(rnorm(n,0,1), rep(rp,n))
  donnees[,4]=rep(rnorm(n,0,1), rep(rp,n))
  donnees[,5]=lambda_1+donnees[,3]
  donnees[,6]=ka_1+donnees[,4]
  dimnames(donnees)[[2]][5]=c("lambda_1_eps")
  dimnames(donnees)[[2]][6]=c("ka_1_eps")

  # Bruit gaussien sur les paramètres de D2
  donnees[,7]=rep(rnorm(n,0,1), rep(rp,n))
  donnees[,8]=rep(rnorm(n,0,1), rep(rp,n))
  donnees[,9]=lambda_2+donnees[,7]
  donnees[,10]=ka_2+donnees[,8]
  dimnames(donnees)[[2]][9]=c("lambda_2_eps")
  dimnames(donnees)[[2]][10]=c("ka_2_eps")
```

```
# Tri des données suivant Id puis Tps
donnees=donnees[order(donnees[,"Id"],donnees[,"Tps"]),]

# D1
donnees[,11]=(1-exp(-(donnees$Tps/donnees$lambda_1_eps)^donnees$ka_1_eps))*100
dimnames(donnees)[[2]][11]=c("D1")

# D2
donnees[,12]=(1-exp(-(donnees$Tps/donnees$lambda_2_eps)^donnees$ka_2_eps))*100
dimnames(donnees)[[2]][12]=c("D2")

# Entretien : 1=rien, 2=léger, 3=lourd
# Seuils pour D1 : 0-50 / 50-70 / 70-100
# Seuils pour D2 : 0-30 / 30-50 / 50-100
donnees[,13]=ifelse((donnees$D1 <= 50),1,ifelse((donnees$D1 > 70),3,2))
dimnames(donnees)[[2]][13]=c("D1c")
donnees[,14]=ifelse((donnees$D2 <= 30),1,ifelse((donnees$D2 > 50),3,2))
dimnames(donnees)[[2]][14]=c("D2c")

# Entretien après croisement des valeurs de D1 et D2
donnees[(donnees$D2c == 1),15]=c("a")
donnees[(donnees$D2c == 1) & (donnees$D1c == 3),15]=c("b")
donnees[(donnees$D2c == 2) & (donnees$D1c == 1),15]=c("a")
donnees[(donnees$D2c == 2) & (donnees$D1c == 2),15]=c("b")
donnees[(donnees$D2c == 2) & (donnees$D1c == 3),15]=c("c")
donnees[(donnees$D2c == 3),15]=c("c")
donnees[(donnees$D2c == 3) & (donnees$D1c == 1),15]=c("b")
dimnames(donnees)[[2]][15]=c("Entretien")
donnees$Entretien = factor(donnees$Entretien)

donnees_trait=donnees[,c("Id","Tps","D1","D2","Entretien")]

# Classification par apprentissage
# p = 0,80 : taux d'entraînement des données
validation_index <- createDataPartition(donnees_trait$Entretien, p=0.80, list=FALSE)

# 20 % pour la validation
validation <- donnees_trait[-validation_index,]
# 80 % restants pour l'entraînement et le test
dataset <- donnees_trait[validation_index,]

# Stockage des résultats
x <- donnees_trait[,c("D1","D2")]
y <- donnees_trait[,c("Entretien")]

# Algorithme pour la validation croisée
control=trainControl(method="repeatedcv", number=10, repeats=3)
metric <- "Accuracy"

fit.lda <- train(Entretien~., data=dataset, method="lda", trControl=control)
```

```
 # Stockage des prédictions
  predictions <- predict(fit.lda, validation)
  exit_res=confusionMatrix(predictions, validation$Entretien)
  resultats=c(exit_res$overall["Accuracy"],exit_res$overall["AccuracyLower"],
  exit_res$overall["AccuracyUpper"],exit_res$overall["Kappa"])
  return(resultats)
}

##############
# Nombre de sections
n=c(500,1000,5000,10000)
# Nombre de mesures par section
rp=c(3,6,9)
# Préparation du fichier des résultats
p=expand.grid(n=n,rp=rp)

# Vectorisation du résultat de l'ajustement
follow_up_vectorized <- Vectorize(follow_up)
# 100 répétitions de l'ajustement
sims <- replicate(100, follow_up_vectorized(n=p$n, rp=p$rp))
# Calcul des moyennes sur les 100 répétitions
res <- apply(sims, c(1, 2), mean)
# Stockage des résultats
cbind(p, t(res))

# Exportation des résultats sous format LaTeX
xt=cbind(p, t(res))
xt=sapply(xt, format, decimal.mark = ',',digits=3)
tli.table=xtable(xt,digits=3)
print(tli.table, include.rownames = FALSE)

# Classification par apprentissage : différentes méthodes
control <- trainControl(method="repeatedcv", number=10, repeats=3)
set.seed(7)
fit.lda <- train(Entretien~D1+D2, data=data_train, method="lda", trControl=control)
predictions_lda <- predict(fit.lda, validation)
set.seed(7)
fit.rf <- train(Entretien~D1+D2, data=data_train, method="rf", trControl=control)
predictions_rf <- predict(fit.rf, validation)
set.seed(7)
fit.cart <- train(Entretien~D1+D2, data=data_train, method="rpart", metric=metric,
trControl=control)
predictions_cart <- predict(fit.cart, validation)
set.seed(7)
fit.knn <- train(Entretien~D1+D2, data=data_train, method="knn", metric=metric,
trControl=control)
predictions_knn <- predict(fit.knn, validation)
set.seed(7)
fit.svm <- train(Entretien~D1+D2, data=data_train, method="svmRadial", metric=metric,
trControl=control)
predictions_svm <- predict(fit.svm, validation)
set.seed(7)
```

```
fit.gbm <- train(Entretien~D1+D2, data=donnees, method="gbm", metric=metric,
trControl=control)
predictions_gbm <- predict(fit.gbm, validation)

fit.lda$results
cm_lda=confusionMatrix(predictions_lda, validation$Entretien)
cm_rf=confusionMatrix(predictions_rf, validation$Entretien)
cm_cart=confusionMatrix(predictions_cart, validation$Entretien)
cm_knn=confusionMatrix(predictions_knn, validation$Entretien)
cm_svm=confusionMatrix(predictions_svm, validation$Entretien)

results <- resamples(list(lda=fit.lda, cart=fit.cart,knn=fit.knn,svm=fit.svm,gbm=fit.gbm,rf=fit.rf))
summary(results)

rbind(c("lda",round(cm_lda$overall[1:2],3)),c("rf",round(cm_rf$overall[1:2],3)),
c("cart",round(cm_cart$overall[1:2],3)),c("knn",round(cm_knn$overall[1:2],3)),
c("svm",round(cm_svm$overall[1:2],3)))

# Classifieur bayésien
nb.model <- naiveBayes(Entretien~D1+D2, data = data_train)
predict_bay <- predict(object = nb.model, newdata = validation)
Test.mod <- cbind(validation, predict_bay)
head(Test.mod)
table(Test.mod$Entretien, Test.mod$predict_bay)
```

Bibliographie

[1] C. Cortes et V. Vapnik, "Support-Vector Networks", in *Machine Learning* 20 (1995), p. 273–297.

[2] G. James, D. Witten, T. Hastie et R. Tibshirani, *An Introduction to Statistical Learning*, Springer, 2015, p. 426.

[3] M. Kuhn, "Building Predictive Models in R Using the caret Package", in *Journal of Statistical Software* 28 (2005), p. 1–26.

[4] M. Kuhn, "A Short Introduction to the caret Package", in *Journal of Statistical Software* 1 (2016), p. 1–10.

[5] D. Meyer, "Package e1071", in *R Documentation* 1.6-7 (2015), p. 1–62.

[6] T. Therneau et E. Atkinson, "A Introduction to Recursive Partitioning Using the RPART Routines", in *R Documentation* 1 (2017), p. 1–62.

[7] H. Wendt, P. Flandrin et P. Abry, "Régressions par machines à vecteurs supports pour la prédiction de séries chaotiques", in *GRETSI* (2005), p. 153–156.