Projet STA203 : Spectrométrie infrarouge pour la détermination de la teneur en sucre des cookies

Leonardo Martins Bianco - Guillaume Lambert

16 mai 2021

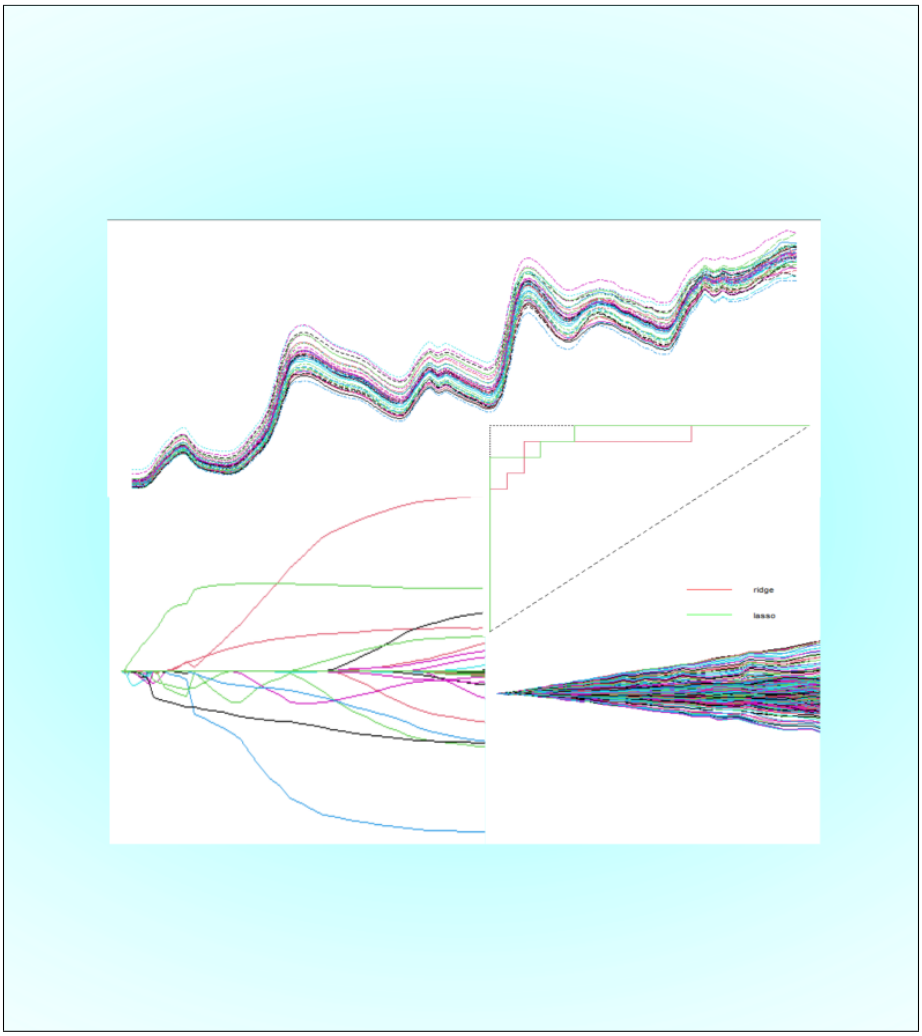


Table des matières

Introduction

Dans ce projet, on s'intéresse à l'étude du jeu de données `cookie` fourni par [1]. Il est composé d'une partie apprentissage de 40 observations et d'une partie test de 32 observations. A chaque observation est associé un spectre mesuré sur 700 fréquences séparées de 2 nanomètres, de 1100 à 2498 nanomètres. L'objectif de cette étude est d'étudier la teneur en sucre des cookies à partir de leur spectre, sans avoir à procéder à des études chimiques coûteuses.

La particularité de ce problème est que les nombres d'observations $n = 40$ et $n = 32$ sont beaucoup plus faibles que le nombre de variables $p = 700$. On se situe donc dans un problème de *grande dimension* où $n << p$. L'intérêt de ce projet est donc de trouver des modèles d'étude et de prédiction dans ce cadre particulier où les modèles de base ne sont pas applicables, comme la régression linéaire. Dans notre projet, on s'intéresse tout particulièrement à la régression régularisée *ridge* et *lasso*.

Dans la première partie, on s'intéressera au choix de la régression ridge pour traiter les données en grande dimension. On étudiera également des notions théoriques derrière la régression ridge pour mieux l'appréhender. Dans la deuxième partie, on effectuera une analyse exploratoire de nos données par étude graphique et par analyse par composantes principales (ACP). On s'intéressera également à la qualité de reconstruction du nuage par l'ACP. Dans la troisième partie, on étudie la régression ridge et on retrouve des résultats théoriques de la partie 1. Dans la quatrième partie, on ne s'intéresse plus à la teneur en sucre quantitative mais à un dépassement de seuil qualitatif de 18. On appliquera donc une régression logistique que l'on pénalisera en ridge et en lasso. On expliquera le choix de la pénalisation et on étudiera les résultats pour désigner le meilleur des deux. Enfin, la cinquième partie est un appendice où on prouve la décomposition en valeurs singulières d'une matrice.

1 Un peu de théorie

Dans un premier temps, on va voir que l'on ne peut pas utiliser la régression linéaire dans le cadre de la grande dimension. Cela va motiver l'utilisation de la régression ridge. Ensuite, on va étudier le choix de la pénalisation de l'intercept et du centrage des données dans cette régression. Enfin, on verra comment se comporte l'estimateur ridge lorsque λ tend vers 0. Pour cette partie théorique, [2] et [3] sont des bonnes références.

1.1 Le choix de la régression ridge dans le cadre de la grande dimension

Les données d'apprentissage contiennent $n = 40$ observations. Chaque observation est décrite par $p = 700$ variables, c'est-à-dire, la dimension du problème est $p = 700$. Ces variables correspondent à une longueur d'onde, séparées par 2nm entre 1100 – 2500nm. On voit que la dimension est beaucoup plus grande que la taille de l'échantillon d'apprentissage, ce qui nous met dans le cas dit de *grande dimension*.

Si l'on appelle $X \in \mathcal{M}_{n,p}(\mathbb{R})$ la matrice du plan d'expérience, chacune des $p = 700$ colonnes peut être vue comme un vecteur dans un espace de dimension $n = 40$. Dans cet espace, on peut avoir alors jusqu'à $n = 40$ vecteurs linéairement indépendants maximum, donc on a au moins $p - n = 660$ vecteurs linéairement dépendants dans les colonnes de X. C'est-à-dire, le rang de X est beaucoup plus petit que son nombre de colonnes p. Une conséquence de ce fait est que la matrice de corrélation $X'X$ (de taille $p \times p$) sera singulière, i.e., elle aura des valeurs propres nulles.

Proposition 1.1. *Soit X une matrice de taille $n \times p$ de rang r. On a*

$$\operatorname{rang}(X'X) = \operatorname{rang}(XX') = \operatorname{rang}(X) = r.$$

Démonstration. Soit A une matrice avec les mêmes caractéristiques que X (même taille et même rang). Montrons d'abord que $Ax = 0 \iff A'Ax = 0$. Si $Ax = 0$, par linéarité $A'Ax = 0$. Si $A'Ax = 0$, alors $x'A'Ax = (Ax)'Ax = \|Ax\|_2^2 = 0$, et cela implique $Ax = 0$. Donc les noyaux de A et $A'A$ sont égaux, et par la formule rang-noyau on conclut que $\operatorname{rang}(A) = \operatorname{rang}(A'A)$. Si l'on prend $A = X$ on obtient l'égalité $\operatorname{rang}(X'X) = \operatorname{rang}(X)$, et si l'on prend $A = X'$ et l'on utilise le fait que $\operatorname{rang}(X) = \operatorname{rang}(X')$, on obtient l'autre égalité. □

Donc si $r < p$, alors il existent des colonnes linéairement dépendantes dans $X'X$, et cela implique $\det(X'X) = 0$. Le lemma 1 de la partie bonus donne une décomposition spectrale valide pour $X'X$ d'où on conclut l'existence de valeurs propres nulles.

On rappelle que pour un modèle de régression linéaire $Y = \theta_0 1_n + X\theta + \varepsilon$, si la matrice X est injective alors l'estimateur des coefficients est

$$\hat{\theta} = (X'X)^{-1} X'Y$$

Donc si la matrice $X'X$ n'est pas inversible, on ne peut pas utiliser ce résultat et le problème devient plus compliqué. Un autre problème proche de ce cas est quand les valeurs propres ne sont pas nulles mais très petites. On rappelle que la variance de l'estimateur $\hat{\theta}$ indiqué ci-dessus est donnée par

$$\mathbb{V}(\hat{\theta}) = \sigma^2 (X'X)^{-1}$$

où σ est la variance des résidus assumés homoscédastiques. Si l'on met la matrice $X'X$ dans la base de vecteurs propres en utilisant le lemma 1, alors son inverse est donné par la matrice $\text{diag}(1/\lambda_1, \dots, 1/\lambda_p)$. Si les valeurs propres sont très petites, alors on voit que la variance explose. Cela implique une mauvaise erreur de prédiction pour cet estimateur.

Une des premières solutions satisfaisantes à ce problème a été proposée par [4]. L'idée se base sur les 2 propositions suivantes.

Proposition 1.2. *Les matrices $X'X$ et $X'X + \tau\mathbb{I}$ ont les mêmes vecteurs propres $\{u_j\}_{j=1,\dots,p}$ et si on note λ_j la j-ème valeur propre de $X'X$ et $\tilde{\lambda}_j$ la j-ème valeur propre de $X'X + \tau\mathbb{I}$, alors $\tilde{\lambda}_j = \lambda_j + \tau$ pour $j = 1, \dots, p$.*

Démonstration. D'une part, si u_j est un vecteur propre de $X'X$ avec valeur propre λ_j, alors $(X'X + \tau\mathbb{I})u_j = \lambda_j u_j + \tau u_j = (\lambda_j + \tau)u_j$. Autrement dit, u_j est vecteur propre de $X'X + \tau\mathbb{I}$ de valeur propre $\tilde{\lambda}_j = \lambda_j + \tau$. D'autre part, si v_j est un vecteur propre de $X'X + \tau\mathbb{I}$ avec valeur propre $\tilde{\lambda}_j$, alors $X'Xv_j = (X'X + \tau\mathbb{I})v_j - \tau\mathbb{I}v_j = (\tilde{\lambda}_j - \tau)v_j$, d'où v_j est vecteur propre de $X'X$ de valeur propre $\lambda_j = \tilde{\lambda}_j - \tau$. □

Proposition 1.3. *Les valeurs propres de $X'X \in \mathcal{M}_{p,p}(\mathbb{R})$ sont positives ou nulles.*

Démonstration. Soit $v \in \mathcal{M}_{p,1}(\mathbb{R})$. Alors $v'X'Xv = (Xv)'Xv = \|Xv\|_2^2 \geq 0$. De plus, $X'X$ est symétrique : $(X'X)_{ij} = \sum_k \tilde{x}_{ik}x_{kj} = \sum_k x_{ki}x_{kj} = (X'X)_{ji}$. Donc la matrice $X'X$ est symétrique positive et ses valeurs propres sont bien positives ou nulles. □

On peut donc ranger les valeurs propres de $X'X$ de manière décroissante et observer une potentielle dégénérescence du rang, c'est-à-dire le cas où les valeurs propres deviennent très proches de 0. La régression ridge permet de remplacer la matrice $X'X$ par la matrice $X'X + \tau\mathbb{I}$. Ainsi, on conserve les mêmes vecteurs propres mais on augmente uniformément de τ les valeurs propres de $X'X$ pour se débarrasser de la dégénérescence du rang. Finalement, on obtient une matrice inversible $X'X + \tau\mathbb{I}$ qui nous permettra de calculer l'estimateur ridge :

$$\hat{\theta}_{ridge}(\tau) = (X'X + \tau\mathbb{I})^{-1}X'Y$$

Il est la solution du problème de minimisation suivant, qui définit la régression ridge. [5]

Definition 1.1 (Régression ridge)**.** On appelle la régression ridge la minimisation du critère pénalisé :

$$\tilde{\theta}_{ridge}(\tau) = \underset{\theta\in\mathbb{R}^p}{\text{argmin}}\|Y - X\theta\|^2 + \tau\|\theta\|^2$$

Ce problème est équivalent au problème sous contrainte :

$$\tilde{\theta}(\delta) = \underset{\theta \in \mathbb{R}^p, \|\theta\|^2 \leq \delta}{\operatorname{argmin}} \|Y - X\theta\|^2$$

Finalement, l'atout de la régression ridge est de pallier le problème de non inversabilité de $X'X$ qui nous empêche d'utiliser les résultats de la régression linéaire. Dans notre cas particulier, celui de la grande dimension, la matrice $X'X$ n'est pas inversible, on ne peut donc pas utiliser la régression linéaire et c'est pourquoi on utilise la régression ridge.

1.2 Pénalisation de l'intercept et transformation des données

L'estimateur ridge solution du problème de minimisation de la définition 1.1 est le suivant :

Theorem 1.4 (Estimateur ridge). *Soit le problème de la régression ridge comme défini précédemment. Alors*

$$\tilde{\theta} = (X'X + \tilde{\tau}\mathbb{I})^{-1} X'Y$$

où $\tilde{\tau}$ est un multiplicateur de Lagrange optimal du problème pénalisé.

Il est important de noter que la régression ridge définie dans la définition 1.1 correspond à un problème pénalisé de régression linéaire sans intercept. Donc pour utiliser cet estimateur on doit centrer Y avant d'appliquer la régression ridge, car cela élimine l'intercept. On pourra le récupérer à la fin car l'intercept sera la moyenne des observations $\bar{Y}$. Il est aussi important de noter que cet estimateur n'est pas invariant par changement d'échelle des observations X. C'est pourquoi on normalise toujours les observations avant de faire la régression ridge. C'est-à-dire que l'on divise chaque colonne par l'écart-type de la colonne.

On écrit maintenant formellement les différents éléments. Soit $\bar{X}$ la matrice où chaque colonne est le vecteur de composantes constantes, égales à la moyenne de la colonne correspondante dans X. On peut centrer les données en soustrayant les moyennes de la matrice originale, i.e., $X - \bar{X}$ sont les variables explicatives centrées, et $Y - \bar{Y}1_n$ sont les observations centrées. Ensuite, on note $\hat{\sigma}_Y$ l'écart-type de Y et $\hat{\sigma}_X$ le vecteur des écarts-types des colonnes de X. Ainsi, dans $(X - \bar{X})/\hat{\sigma}_X$, on divise chaque colonne $(X - \bar{X})_i$ par $(\hat{\sigma}_X)_i$.

On peut maintenant définir les nuages centrés réduits $\tilde{X} := (X - \bar{X})/\hat{\sigma}_X$ et $\tilde{Y} := (Y - \bar{Y}1_n)/\hat{\sigma}_Y$. On note le modèle $Y = \theta_0 1_n + X\theta + \varepsilon$ ajusté sur les données originales, et le modèle $\tilde{Y} = \tilde{\theta}_0 1_n + \tilde{X}\tilde{\theta} + \tilde{\varepsilon}$ ajusté sur les données centrées réduites.

$$\tilde{Y} = \tilde{\theta}_0 1_n + \tilde{X}\tilde{\theta} + \tilde{\varepsilon} \iff \frac{(Y - \bar{Y}1_n)}{\hat{\sigma}_Y} = \tilde{\theta}_0 1_n + \frac{(X - \bar{X})}{\hat{\sigma}_X}\tilde{\theta} + \tilde{\varepsilon}$$

Or, on a

$$\left(\frac{X - \bar{X}}{\hat{\sigma}_X}\right)\tilde{\theta} = X\left(\frac{\tilde{\theta}}{\hat{\sigma}_X}\right) - \bar{X}\left(\frac{\tilde{\theta}}{\hat{\sigma}_X}\right)$$

où $\left(\frac{\tilde{\theta}}{\hat{\sigma}_X}\right)_j := \tilde{\theta}_j/(\hat{\sigma}_X)_j$. Ainsi

$$\boxed{Y = \left(\bar{Y} + \hat{\sigma}_Y \tilde{\theta}_0\right) 1_n - \hat{\sigma}_Y \bar{X} \left(\frac{\tilde{\theta}}{\hat{\sigma}_X}\right) + \hat{\sigma}_Y X \left(\frac{\tilde{\theta}}{\hat{\sigma}_X}\right) + \tilde{\varepsilon}}$$

Ce modèle est équivalent au modèle $Y = \theta_0 1_n + X\theta + \varepsilon$. Donc a on

$$\begin{cases} (\bar{Y} + \hat{\sigma}_Y \tilde{\theta}_0)\mathbb{1}_n - \tilde{\sigma}_Y \bar{X} \left(\frac{\tilde{\theta}}{\hat{\sigma}_X}\right) = \theta_0 \mathbb{1}_n \\ X\theta = \tilde{\sigma}_Y X \left(\frac{\tilde{\theta}}{\hat{\sigma}_X}\right) \end{cases}$$

$$\iff \begin{cases} \theta_0 = \bar{Y} + \hat{\sigma}_Y \tilde{\theta}_0 - \hat{\sigma}_Y \sum_{j=1}^{700} \bar{X}_j \frac{\tilde{\theta}_j}{(\hat{\sigma}_X)_j} \\ \theta = \hat{\sigma}_Y \left(\frac{\tilde{\theta}}{\hat{\sigma}_X}\right) \end{cases}$$

avec $\tilde{\theta} = \left(\tilde{X}'\tilde{X} + \tilde{\tau}\mathbb{I}\right)^{-1} \tilde{X}'\tilde{Y}$. Grâce à ce système, noté $(*)$, on va pouvoir étudier les estimateurs des coefficients selon la pénalisation de l'intercept et la transformation des données.

Estimation des coefficients sans pénalisation de l'intercept. Ce cas particulier est celui de la pénalisation ridge d'un modèle linéaire avec intercept sans pénalisation de ce dernier. On ne peut donc pas utiliser directement le théorème 1.4.. On commence par centrer et réduire les données. Ainsi, on n'inclue pas l'intercept dans le modèle et on estime $\tilde{\theta}$ grâce au théorème 1.4.. On utilise ensuite le système $(*)$ pour trouver les coefficients θ et θ_0.

Cas 1 : X centré. Dans ce cas, le système $(*)$ devient

$$\begin{cases} \tilde{\theta} = \theta \\ \tilde{\theta_0} = \theta_0 + \left(\overline{X}\tilde{\theta}\right)_i \quad \text{for} \quad i = 1, \ldots, n \end{cases}$$

La deuxième équation ne représente pas un système d'équations, mais une seule équation, car toutes les composantes du produit $\overline{X}\tilde{\theta}$ sont les mêmes. Si on note $\left(\bar{X}_k\right)_{k=1,\ldots,p}$ les éléments de n'importe quelle ligne de $\bar{X}$, on a

$$\tilde{\theta_0} = \theta_0 + \sum_{k=1}^{p} \overline{X}_k \tilde{\theta}_k$$

On remarque que, par rapport au système $(*)$, les coefficients $\tilde{\theta}$ ne sont pas modifiés, mais le coefficient de l'intercept est différent. On conclut que l'effet du centrage porte seulement sur le changement du coefficient de l'intercept.

Cas 2 : Y centré. Dans ce cas, le système $(*)$ devient

$$\begin{cases} \tilde{\theta} = \theta \\ \tilde{\theta_0} = \theta_0 - \bar{Y} \end{cases}$$

On sait que dans un modèle où Y a été centrée, il n'y a pas d'intérêt d'inclure d'intercept, donc $\tilde{\theta}_0 = 0$. Cela implique que $\theta_0 = \bar{Y}$, i.e., l'intercept du modèle original peut être estimé en calculant la moyenne des observations. Comme avant, les coefficients autres que l'intercept ne sont pas modifiés par le centrage de Y.

Cas 3 : X et Y centrées. Dans ce cas, le système $(*)$ devient

$$\begin{cases} \tilde{\theta} = \theta \\ \tilde{\theta}_0 = \theta_0 - \left(\bar{Y} - \bar{X}\tilde{\theta}\right) \end{cases}$$

On voit donc que quand on centre X seulement, on ajoute $\bar{X}\tilde{\theta}$ au coefficient de l'intercept et que quand on centre Y seulement, on soustrait $\bar{Y}$. Finalement, quand on centre X et Y, on voit que l'on fait l'un puis l'autre. De même, on voit que θ n'est pas modifié.

Cas 4 : X et Y centrées réduites. Cela correspond au système $(*)$.

Ces différents cas particuliers seront vérifiées numériquement sur nos données dans la partie 3.

1.3 Régression ridge lorsque le coefficient de pénalisation tend vers 0

On cherche maintenant à déterminer l'estimateur ridge lorsque $\tau \to 0$. Dans l'appendice, on a montré la décomposition en valeurs singulières de X. Cette décomposition est un outil très important dans plusieurs analyses statistiques. On peut trouver une expression simple pour $(X'X + \lambda\mathbb{I})^{-1}$ en utilisant cette décomposition. D'abord, on rappelle que dans l'appendice on a utilisé la décomposition spectrale de $X'X$, qui s'écrit $X'X = V\Lambda V'$ avec $\Lambda = \text{diag}(\sigma_1^2, \ldots, \sigma_p^2)$ les valeurs propres de $X'X$ (mais seulement les r premières sont non-nulles). Cela peut être écrit comme $X'X = \sum_{j=1}^{r} \sigma_j^2 v_j v_j'$. Alternativement, si l'on part de la DVS, on peut déduire cette expression par un calcul simple :

$$X = \sum_{j=1}^{r} \sigma_j u_j v_j \quad \text{et} \quad X' = \sum_{i=1}^{r} \sigma_i u_i v_i$$

$$\begin{aligned} X'X &= \left(\sum_{i=1}^{r} \sigma_i u_i v_i\right)\left(\sum_{j=1}^{r} \sigma_j u_j v_j\right) \\ &= \sum_{i=1}^{r}\sum_{j=1}^{r} \sigma_i \sigma_j v_i \underbrace{u_i' u_j}_{=\delta_{ij}} v_j' \\ &= \sum_{i=1}^{r} \sigma_i^2 v_i v_i' = V\Lambda V' \end{aligned}$$

On somme $\lambda\mathbb{I}$:

$$(X'X + \lambda\mathbb{I}) = V\Lambda V' + \lambda\mathbb{I} = V\Lambda V' + \lambda V V' = V(\Lambda + \lambda\mathbb{I})V'$$

et donc l'inversion est très simple, car dans la base de vecteurs propres la matrice est diagonale :

$$(X'X + \lambda\mathbb{I})^{-1} = V(\Lambda + \lambda\mathbb{I})^{-1}V' = \sum_{i=1}^{r} \frac{1}{\sigma_i^2 + \lambda} v_i v_i'$$

En particulier on trouve facilement la limite quand $\lambda \to 0$

$$\lim_{\lambda \to 0}(X'X + \lambda\mathbb{I})^{-1} = \sum_{i=1}^{r} \frac{1}{\sigma_i^2} v_i v_i'$$

Si l'on note $A_\lambda = (X'X + \lambda\mathbb{I})^{-1}X'$ et $A_0 = \lim_{\lambda\to 0} A_\lambda$, alors on a

$$A_0 = \sum_{i=1}^{r} \frac{1}{\sigma_i^2} v_i v_i' X' = \sum_{i=1}^{r} \frac{1}{\sigma_i^2} v_i (Xv_i)'$$

Finalement, on peut étudier le comportement de l'estimateur ridge $\tilde{\theta} = \left(X'X + \tilde{\lambda}\mathbb{I}\right)^{-1} X'Y$ lorsque $\lambda \to 0$. On a

$$\lim_{\lambda \to 0} \left(X'X + \tilde{\lambda}\mathbb{I}\right)^{-1} X'Y = A_0 Y$$

On trouve ainsi une formule pour l'estimateur ridge à la limite $\lim_{\lambda\to 0} \tilde{\theta}_\lambda = \sum_{i=1}^{r} \frac{1}{\sigma_i^2} v_i \, (Xv_i)' \, Y$. On observera en pratique cette convergence dans la partie 3.2.

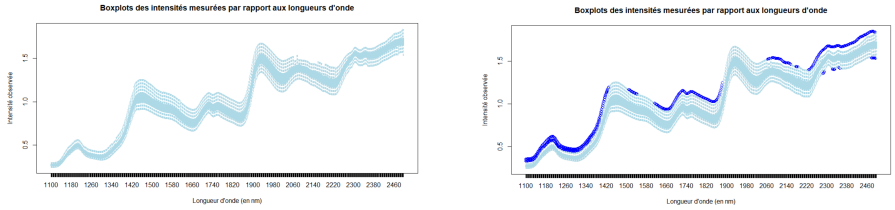
2 Analyse exploratoire

On procède à une analyse exploratoire de nos données. On réalise une étude graphique puis une analyse par composantes principales (ACP). On étudie également la reconstruction des données grâce à l'ACP.

2.1 Analyse descriptive visuelle des spectres

On appelle respectivement `xtrain` et `xtest` les matrices des variables explicatives du jeu de données d'apprentissage et de test. De même, `ytrain` et `ytest` sont les vecteurs réponses (teneur en sucre) respectivement pour les jeux de données d'apprentissage et de test.

On souhaite étudier visuellement les spectres associés aux différentes observations. On trace donc un graphique de boîtes à moustaches et les "courbes" des spectres pour chaque observation du jeu d'apprentissage `xtrain`. On transforme les labels des variables (axe des abscisses) pour qu'ils correspondent aux longueurs d'onde. Ainsi la lecture graphique est plus aisée.

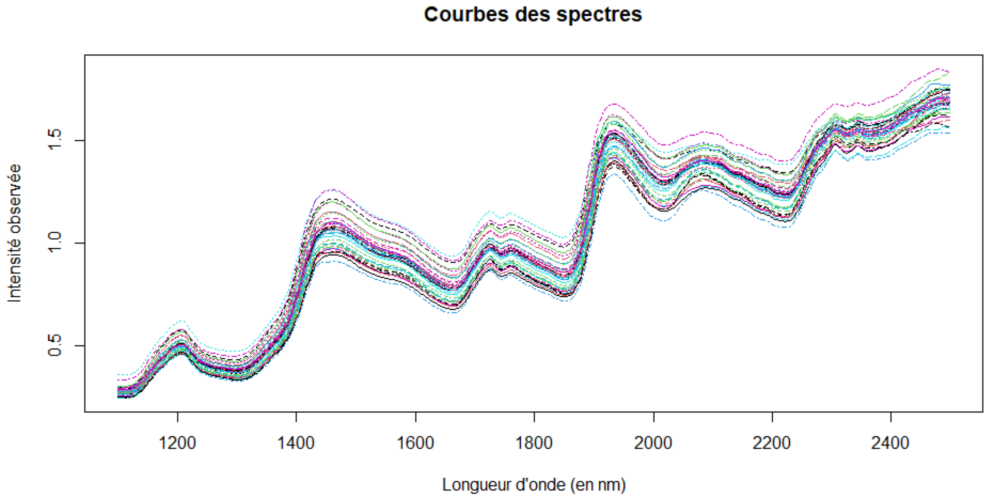


(a) Sans outliers (b) Avec outliers

FIGURE 1 – Boxplots des intensités mesurées par rapport aux longueurs d'onde

On remarque que les intensités observées sont comprises entre 0 et 2, plus précisément, la valeur minimale est 0.242927 et la valeur maximale est 1.85297. Sur le graphique de gauche, où les outliers ne sont pas affichés, on observe l'allure générale des boxplots. La tendance globale est croissante avec des fluctuations locales. Il est difficile d'étudier la position de la médiane par rapport à la moyenne ainsi que les quartiles car les variables sont très nombreuses. On peut noter que la dispersion des données autour de la médiane est plus importante à partir de la 180ème variable (1460 nm) par rapport aux premières variables. Lorsqu'on ajoute les outliers (en bleu foncé) sur la figure de droite, on remarque qu'ils sont maximaux pour l'écrasante majorité d'entre eux. Seule la fin du jeu de données, à partir de 2300 nm, présente des outliers minimaux.

On continue l'étude graphique avec les courbes des spectres.



On remarque que les différentes courbes des spectres ont le même motif avec seulement une translation verticale. En effet, les courbes ne s'entremêlent que très peu. Chaque spectre semble donc posséder le même schéma avec une translation verticale provoquée par les différences entre les individus, notamment la teneur en sucre.

Comme pour les boxplots, on retrouve une corrélation globale positive : lorsque les longueurs d'onde augmentent, l'intensité observée augmente.

On peut aussi considérer la corrélation des variables explicatives entre elles. On déduit graphi-

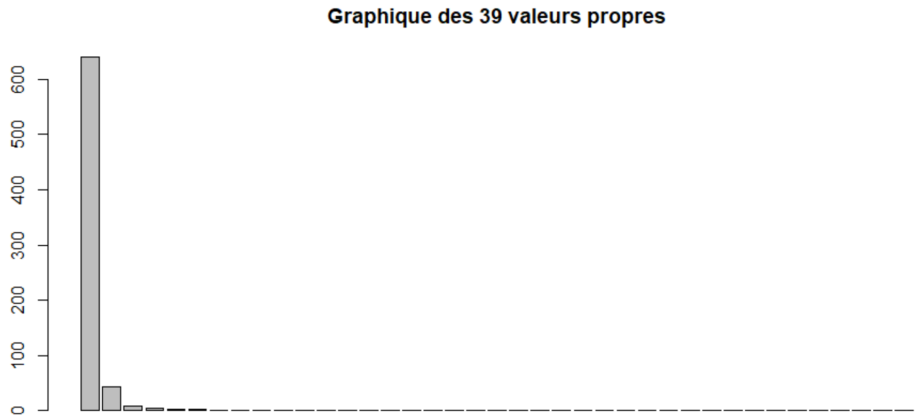
quement que les variables sont fortement corrélées entre elles. En effet, les courbes ne se chevauchent pas. Un individu avec une grande intensité observée pour une longueur d'onde aura une grande intensité observée pour les autres longueurs d'onde (par rapport à un autre individu).

2.2 Analyse par composantes principales

On poursuit l'analyse exploratoire de notre jeu de données par une analyse par composantes principales (ACP).

A l'aide de la fonction `PCA`, on réalise une ACP sur le jeu de données d'apprentissage `xtrain`. La fonction nous fournit 39 dimensions et donc il y a 39 valeurs propres. C'est beaucoup moins que le nombre de variables (700). On obtient si peu de valeurs propres car à partir de la 40ème, elles sont trop proches de 0 et donc insignifiantes dans l'ACP. En effet, on calcule les valeurs propres de la matrice de corrélation de `xtrain` avec `eigen(cor(xtrain))$values` puis on les classe par ordre décroissant. On remarque que l'ordre des 39 premières valeurs propres va de 10^3 à 10^{-4} puis l'ordre des $700 - 39 = 661$ autres valeurs propres est inférieur à 10^{-13}. On en conclut que l'ACP ne considère plus les valeurs propres après le passage de l'ordre 10^{-4} à 10^{-13}. On retrouve donc le résultat de la partie 1.1 où nous avons dit que, le rang de $X'X$ étant inférieur à p, la matrice $X'X$ a au plus 40 valeurs propres non nulles.

On représente ces 39 valeurs propres sur un barplot :

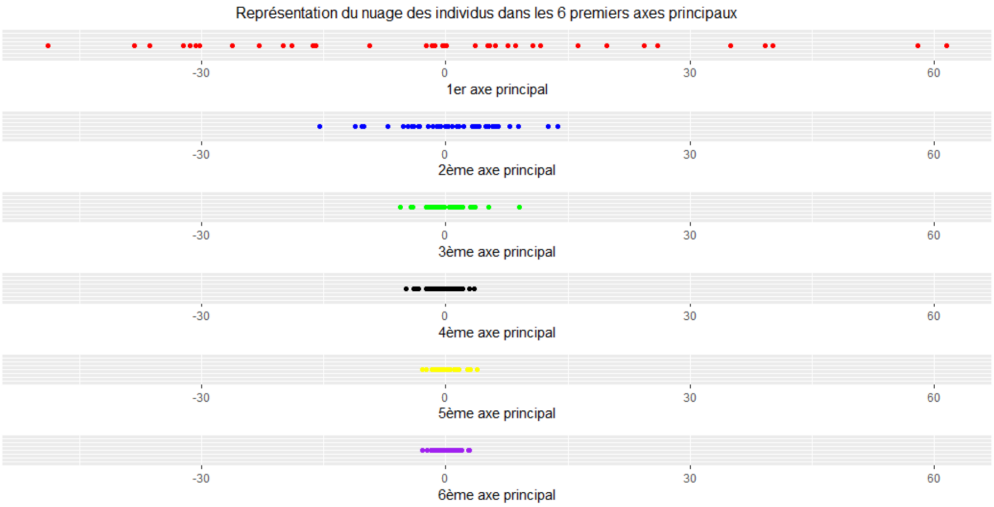


On remarque que l'inertie associée au premier axe principal du nuage des individus, c'est-à-dire la première valeur propre λ_1, est beaucoup plus grande que les suivantes. On voit que les valeurs propres sont rapidement proches de 0 et il est donc difficile d'étudier davantage le barplot. C'est pourquoi on s'intéresse aux 10 premières valeurs propres (arrondies au centième) :

λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	λ_8	λ_9	λ_1
639.82	43.61	7.91	3.81	2.17	1.65	0.39	0.22	0.09	0.06

On remarque que seules les 6 premières valeurs propres sont supérieures à 1. Or, étant donné que l'ACP est centrée-normée, le pourcentage d'inertie associé à un axe s est $\lambda_s/700$ et $\sum_j \lambda_j = 700$.

Ainsi, si $\lambda_s < 1$, alors l'axe s représente moins de données qu'une variable isolée. C'est pourquoi on sélectionne les 6 premiers axes. On représente maintenant les nuages des individus et des variables dans les 6 premiers axes principaux.



On représente ici le nuage des individus dans les 6 premiers axes principaux. Chaque ligne correspond à une composante principale et comporte 40 points (le nombre d'individus du jeu d'apprentissage `xtrain`). L'échelle est la même pour chaque axe principal afin de comparer les composantes principales des individus entre elles.

On remarque que la dispersion des points entre eux et autour de 0 est de moins en moins importante quand le rang de la composante principale augmente. De plus, on voit qu'elle est beaucoup plus importante dans la première composante principale que dans les 5 suivantes. Or, elle est mesurée par l'inertie projetée dans le cadre de l'ACP centrée. En effet, on a la définition suivante : si on note H_i la projection de l'individu M_i sur l'axe u pour $i \in \{1, ..., I\}$, alors

$$\text{Inertie}(u) = \frac{1}{I} \sum_{i=1}^{I} OH_i^2$$

De plus, étant donné que dans l'ACP centrée l'origine est l'individu moyen, on a

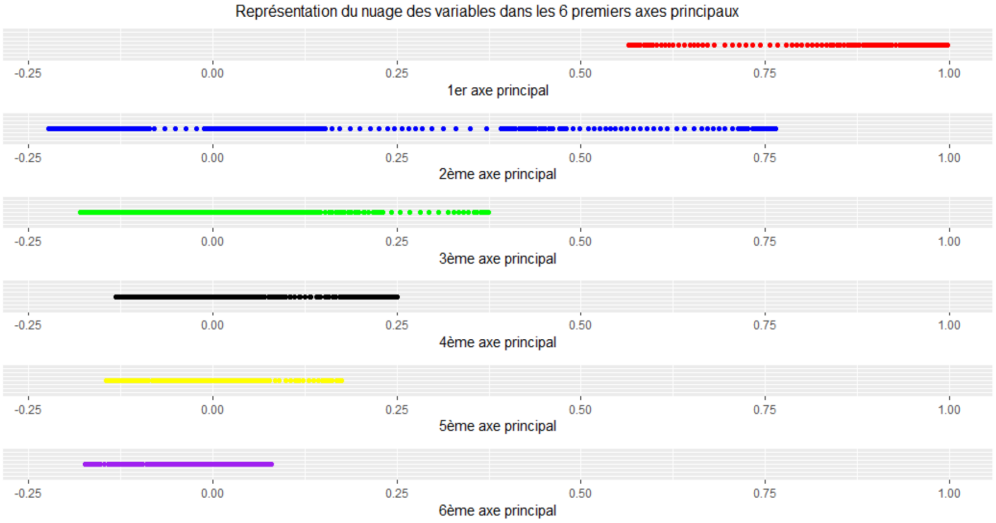
$$\text{Inertie}(u) = \frac{1}{2I^2} \sum_{i=1}^{I} \sum_{j=1}^{J} d^2(H_i, H_j)$$

Ainsi, l'inertie projetée du nuage des individus dans le cadre d'une ACP centrée mesure la dispersion des points projetés par rapport à l'origine mais aussi entre eux (distances interindividuelles). On peut donc réinterpréter l'observation précédente. L'inertie de la projection du nuage des individus sur les axes principaux diminue avec l'augmentation du rang. L'inertie de la projection du nuage des individus selon le premier axe principal est beaucoup plus importante que celle selon les 5 axes principaux suivants.

On peut interpréter mathématiquement cette observation graphique à partir de la méthode de construction l'ACP. En effet, on sait que les vecteurs propres sont rangées par ordre décroissant de leur valeur propre associée. Or, on sait également que la valeur propre associée à un vecteur propre est l'inertie du nuage projeté sur la droite générée par le vecteur propre. Donc il est normal d'observer une décroissance des inerties lorsque le rang de l'axe factoriel augmente.

Concernant la grande différence entre l'inertie projetée associée au premier axe factoriel et les inerties des 5 suivants, on peut le déduire de la grande différence entre la première valeur propre et les 5 suivantes (voir le tableau ici). On voit que λ_1 est 14 fois plus grande que λ_2, 80 fois plus grande que λ_3, 167 fois plus grande que λ_4, 294 plus grande que λ_5 et enfin 387 fois plus grande que λ_6.

On représente maintenant le nuage des variables dans les 6 premiers axes principaux.



Chaque ligne correspond à une composante principale et comporte 700 points (le nombre de variables du jeu d'apprentissage `xtrain`). L'échelle est la même pour chaque axe principal afin de comparer les composantes principales des variables entre elles.

Tout d'abord, on observe un effet taille selon le premier axe principal. C'est-à-dire que les variables contribuent toutes fortement au premier axe factoriel et dans le même sens. On le voit sur le premier graphique où les projections sont toutes regroupées entre 0.5 et 1. On peut interpréter cela grâce au graphique des spectres (ici) où on voit que les valeurs des variables sont corrélées positivement entre elles : les courbes ne se chevauchent pas. Un individu avec une grande intensité observée pour une longueur d'onde aura une grande intensité observée pour les autres longueurs d'onde.

Ensuite, on remarque que la distance des variables projetées par rapport à 0 est de moins en moins importante quand le rang de la composante principale augmente. De plus, on voit que ces distances dans la première composante principale sont beaucoup plus importantes que dans les 5 suivantes. On trouve une observation graphique similaire au nuage des individus à un détails près : dans la première composante principale des variables, les variables projetées sont éloignées de 0 mais sont toutes proches les unes des autres. Ce qui diffère avec le cas précédent où l'inertie mesure les distances interindividuelles des individus projetés. On ne peut donc pas appliquer le même

raisonnement dans le cadre des variables que dans le cadre des individus.

On va voir qu'ici on s'intéresse uniquement à l'éloignement des variables projetées par rapport à 0 et elles ne sont pas nécessairement dispersées entre elles. On va interpréter cela mathématiquement. Premièrement, étant donné que l'on se trouve dans le cadre d'une ACP réduite, l'inertie projetée du nuage des variables est

$$\text{Inertie}(u) = \sum_{i=k}^{K} (T_k)^2$$

où K est le nombre de variables et T_k la projection de la k-ème variable. L'inertie projetée du nuage des variables mesure donc bien l'éloignement des variables projetées par rapport à 0. C'est un point commun avec l'inertie projetée du nuage des individus. Deuxièmement, étant donné que l'origine n'est pas la variable moyenne, on ne peut pas relier la somme des distances intervariables avec l'inertie projetée comme précédemment. Ainsi, on n'a pas de contrainte sur la dispersion des variables projetées entre elles. C'est la différence avec la projection du nuage des individus.

Après avoir étudié la différence d'interprétation entre l'inertie projetée du nuage des individus et des variables, on peut étudier la représentation des composantes principales des variables. On reprend simplement l'étude des composantes principales des individus en retirant le critère des distances interindividuelles :

— La distance des points par rapport à 0 est de moins en moins importante quand le rang de la composante principale augmente. En effet, les axes principaux sont rangés dans l'ordre décroissant des valeurs propres associées. Ces valeurs propres sont les inerties projetées et sont donc mesurées par les distances des variables projetées à l'origine.

— On voit que les distances à 0 dans la première composante principale sont beaucoup plus importantes que dans les 5 suivantes. En effet, comme précédemment, on voit que λ_1 est 14 fois plus grande que λ_2, 80 fois plus grande que λ_3, 167 fois plus grande que λ_4, 294 plus grande que λ_5 et enfin 387 fois plus grande que λ_6.

Le dernier commentaire concernant toutes les composantes principales est le suivant : les individus projetés ne semblent pas être bornés alors que les variables projetées semblent être bornées par 1 en valeur absolue. En effet, on se situe dans le cadre d'une ACP réduite, c'est-à-dire que les variables sont toutes de norme 1 et leur projection sont donc forcément de norme au plus 1.

2.3 Reconstruction du nuage

On s'intéresse maintenant à la reconstruction du nuage. Ici, on considère le nuage comme étant le jeu de données `xtrain`. On peut le voir comme le nuage des individus ou le nuage des variables grâce aux relations de dualité.

Si on regarde `xtrain` sous la forme du nuage des individus, on peut voir l'ACP comme un changement de base de $\mathbb{R}^K$ où $K = 700$ est le nombre de variables. On a alors

$$X = \sum_{k=1}^{K} F_k u_k'$$

où X est la matrice du plan d'expérience centrée réduite, F_k est la k-ème composante principale des individus et u_k est le k-ème axe principal des individus. Dans cette reconstruction, on peut

négliger les termes de la somme associés aux plus faibles valeurs propres, et dans ce cas on obtient une approximation de X. On peut donc reconstruire le nuage centré réduit puis ensuite récupérer le nuage initial en "annulant" le centrage et la réduction.

On construit donc une fonction `reconstruct(res,nr,Xm,Xsd)` effectuant la reconstruction du nuage à partir de `res` l'ACP, `nr` le nombre d'axes principaux utilisés, `Xm` le vecteur des moyennes des colonnes et `Xsd` le vecteur des écarts-types des colonnes. D'après la formule de reconstruction, on a besoin de $(F_k)_{k=1,...,39}$ et $(u_k)_{k=1,...,39}$. La fonction `PCA` nous permet de récupérer les composantes principales $(F_k)_{k=1,...,39}$ mais ne nous fournit pas les axes principaux $(u_k)_{k=1,...,39}$. On obtient ces derniers par une formule de dualité :

$$u_k = \frac{1}{\sqrt{\lambda_k}} G_k$$

où G_k est la k-ème composante principale des variables. Voici le code de la fonction `reconstruct` :

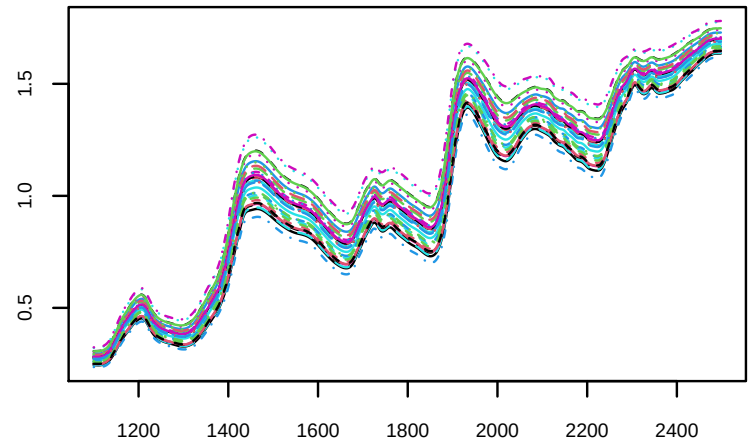
```
reconstruct ← function(res,nr,Xm,Xsd)
{
  #Les nr premiers axes principaux des individus :
  u_s ← t(t(res$var$coord[,1:nr])/sqrt(res$eig[,1])[1:nr])

  #Reconstruction du nuage centre reduit par multiplication
  #matricielle des nr premieres composantes principales des
  #individus par la transposee des nr premiers axes principaux
  #des individus
  rep ← res$ind$coord[,1:nr]%*%t(u_s)

  #On multiplie chaque colonne de la matrice centree reduite
  #reconstruite par l'ecart-type des variables explicatives.
  #Autrement dit, on annule la reduction des donnees :
  rep ← sweep(rep,2,res$call$ecart.type,FUN="*")

  #On ajoute a chaque colonne de la matrice centree
  #reconstruite la moyenne des variables explicatives.
  #Autrement dit, on annule le centrage des donnees :
  rep ← sweep(rep,2,res$call$centre,FUN="+")

  #Finalement, on renvoie la matrice reconstruite
  return(rep)
}
```

On vérifie notre fonction `reconstruct` en reconstruisant le nuage avec le plus d'axes principaux possibles (nr = 39) puis on calcule différentes erreurs. On note `rec` le nuage reconstruit. Grâce au package `Metrics`, on calcule ensuite l'erreur quadratique moyenne (`RMSE`) et l'erreur moyenne absolue (`MAE`) : `RMSE` $= 2.51e-16$ et `MAE` $= 1.83e-16$. Ces erreurs sont extrêmement faibles et on en conclut que la fonction `reconstruct` fonctionne correctement.

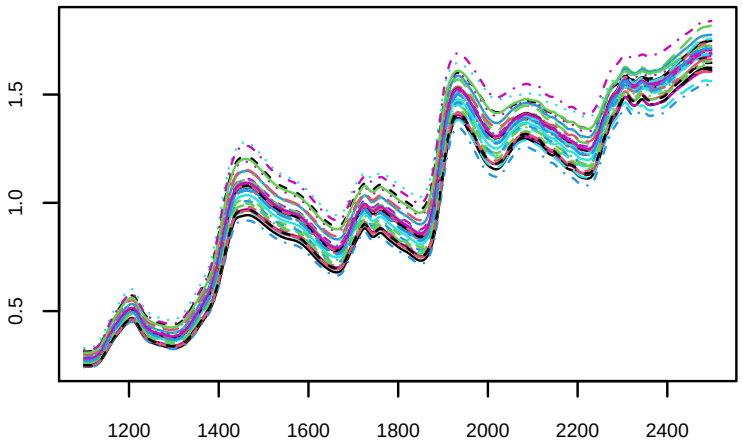
Afin d'observer les qualités de reconstruction du nuage, on va reconstruire le nuage pour $nr = 1, ...5, 39$ puis afficher les courbes des spectres. On affiche dans le titre de chacun des 6 graphiques le nombre d'axes principaux utilisés `nr`, le `RMSE` et le `MAE`.

Reconstruction du nuage pour nr = 1,...,5,39

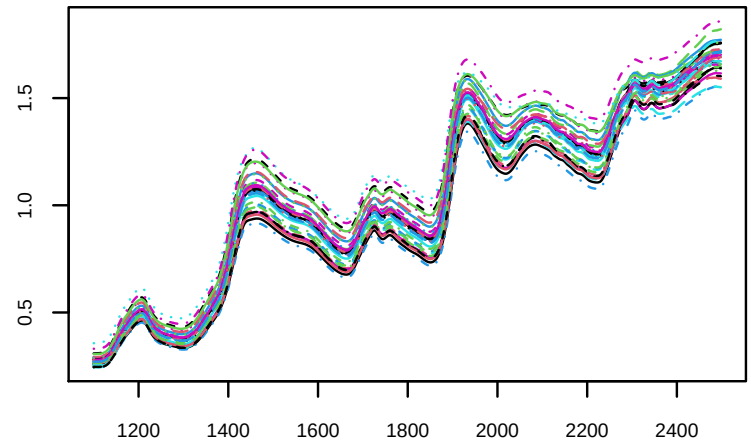
nr = 1 , RMSE = 1.686e−02 , MAE = 1.129e−02



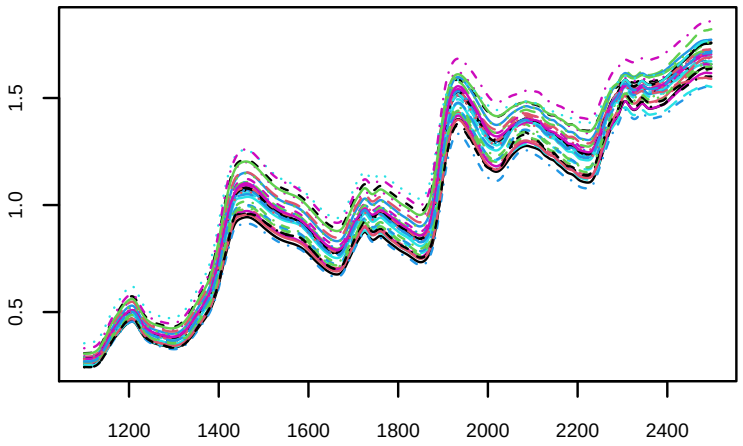
nr = 2 , RMSE = 9.128e−03 , MAE = 6.465e−03



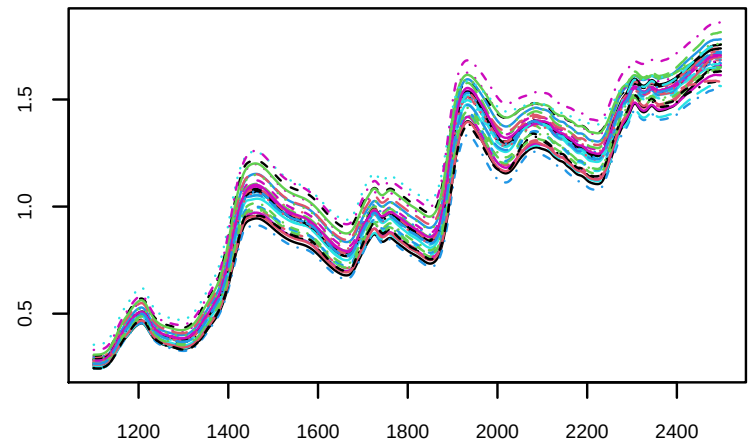
nr = 3 , RMSE = 7.389e−03 , MAE = 4.998e−03



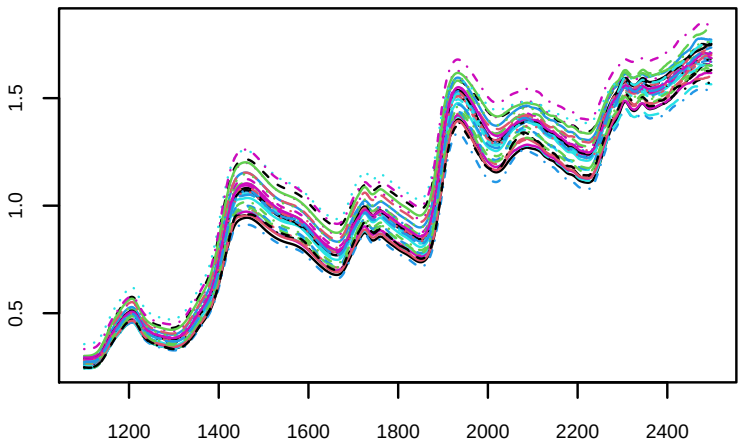
nr = 4 , RMSE = 4.864e−03 , MAE = 3.534e−03



nr = 5 , RMSE = 3.628e−03 , MAE = 2.620e−03



nr = 39 , RMSE = 2.508e−16 , MAE = 1.829e−16



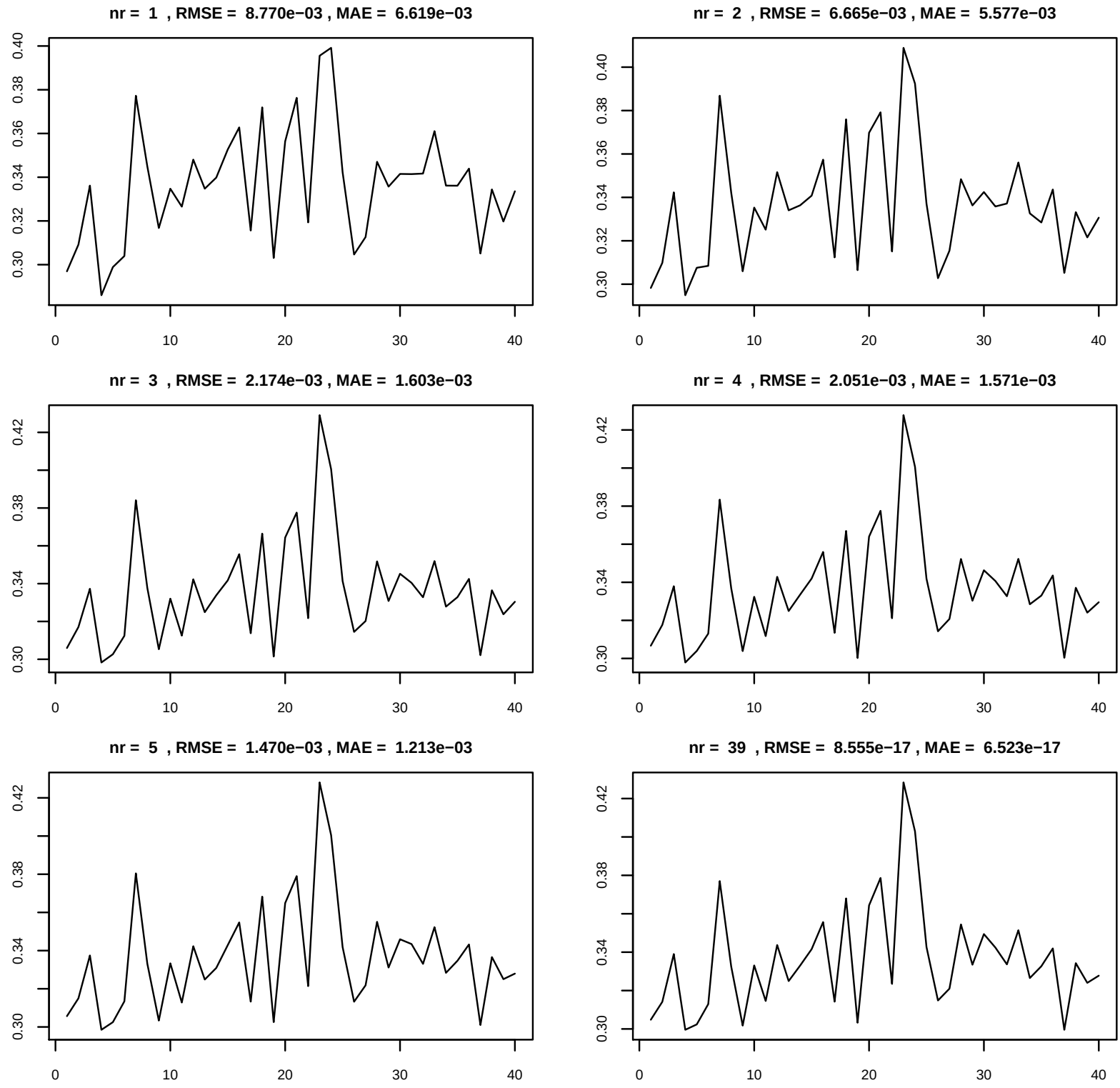
Tout d'abord, on commence par une observation graphique des spectres reconstruits que l'on compare avec le "vrai" graphique des spectres (ici). On remarque que l'aspect des spectres reconstruits est très similaire avec l'aspect des vrais spectres et cela dès le graphique où `nr` = 1. La reconstruction avec un seul axe principal est donc très satisfaisante et on l'explique par la grande valeur de $\lambda_1 = 639.82$ qui est proche de $K = 700$. Ensuite, on remarque que l'évolution des spectres reconstruits entre `nr` = 1 et les autres valeurs de `nr` sont très peu visibles à l'oeil nu. C'est pourquoi on s'intéresse maintenant aux erreurs `RMSE` et `MAE`.

En comparant les ordres de grandeur des erreurs `RMSE` et `MAE`, on voit maintenant l'amélioration de la qualité de reconstruction du nuage. En effet, on gagne un ordre de grandeur entre la reconstruction où `nr` = 1 et celles où `nr` = $2, 3, 4, 5$: on passe de 10^{-2} à 10^{-3}. On améliore de manière beaucoup plus importante la reconstruction quand on utilise les 39 axes principaux. En effet, l'ordre de grandeur des 2 erreurs est 10^{-16}. C'est une illustration de la possibilité de reconstruction du nuage à partir de tous les axes principaux possibles.

On remarque dans notre cas que la reconstruction avec un seul axe principal (sur 39) est déjà très satisfaisante avec le `RMSE` et le `MAE` d'ordre 10^{-2}. Ici, la réduction à une dimension est donc très performante et conserve bien les données.

On représente maintenant les reconstructions de la variable `X24` pour `nr` = $1, ..., 5, 39$. Les valeurs de `X24` sont représentées pour les 40 de `xtrain`.

Reconstruction de la variable X24 pour nr = 1,...,5,39



Comme pour la reconstruction du nuage, on remarque que les erreurs `RMSE` et `MAE` sont de plus en plus petites lorsqu'on augmente le nombre de composantes principales. En effet, pour `nr` entre 1 et 5, les erreurs sont d'ordre 10^{-3} et avec `nr` = 39, les erreurs sont d'ordre 10^{-17}. On doit donc que la qualité de reconstruction augmente avec le nombre de composantes principales et avec une reconstruction complète (`nr`= 39), l'erreur est insignifiante. On observe également cette convergence sur les représentations graphiques où les points de $X24$ semblent converger.

3 Régression pénalisée

Dans cette section on va utiliser différentes librairies `R` pour ajuster une régression ridge aux données. Cela nous permettra de vérifier le comportement de la régression ridge pour des différents valeurs du paramètre de régularisation τ comme discuté dans la partie 1. Aussi, on pourra vérifier numériquement les formules d'effet des centrages et réductions. Une bonne référence pour la partie logiciel `R` et régression ridge se trouve dans [6].

3.1 Variation de l'intercept en fonction de τ

Nous avons remarqué que la pénalisation de la régression ridge correspond à une contrainte sur la norme des coefficients. Ainsi, quand le paramètre τ est trop petit, on ne pénalise pas les coefficients et ils doivent s'approcher de ceux obtenus par une régression linéaire simple. Lorsque la paramètre croît vers l'infini, la norme des coefficients doit s'approcher de zéro, et il nous reste que l'intercept pour ajuster le modèle. Or, le meilleur modèle constant est celui égal à la moyenne des observations. Donc on doit observer le coefficient de l'intercept tendre vers $\bar{Y}$.

Dans notre code, on crée une grille de 100 valeurs uniformément espacées dans $[10^6, 10^{-10}]$. On utilise la librairie `glmnet` pour ajuster un modèle ridge pour chaque valeur du paramètre τ dans la grille avec les données originales, sans aucune transformation de centrage ou réduction. La figure 2 montre l'intercept θ_0 en fonction de τ.

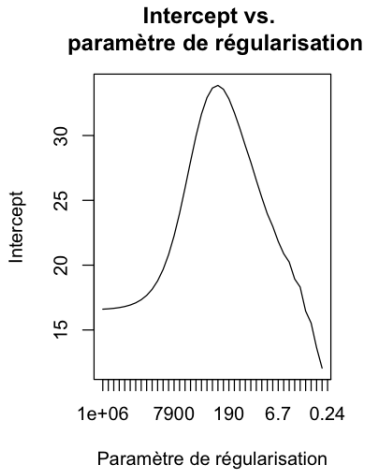
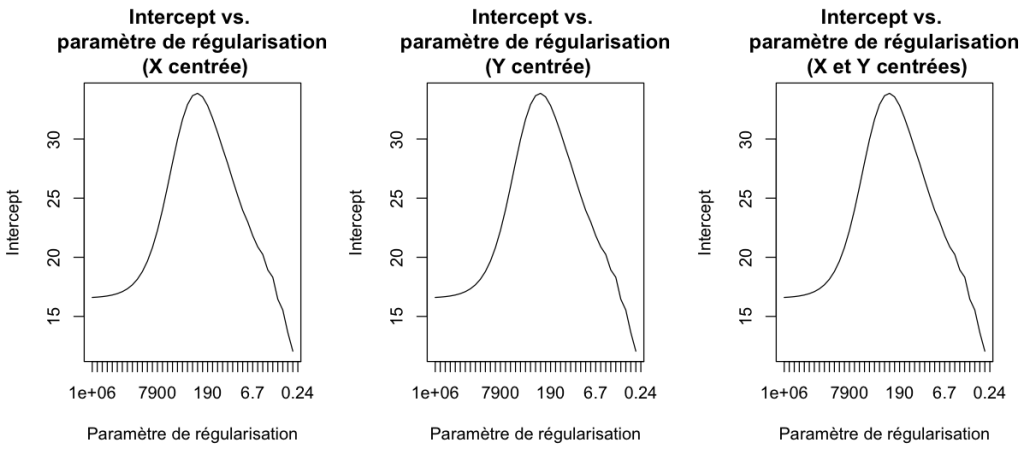


FIGURE 2

D'abord, nous observons que quand $\tau >> 1$, l'intercept vaut $\theta_0 = 16.60476$. On a $\bar{Y} = 16.54475$, ce qui démontre la validité de notre intuition. Ensuite, on note que quand $\tau << 1$, on a $\theta_0 = 12.05811$. Si l'on ajuste une régression linéaire simple, on obtient un intercept $\theta_0^{\text{reg. lin.}} = 12.16582$. Ici c'est important de dire que `glmnet` fonctionne par descente de gradient, et donc on doit ajuster le paramètre `thresh = 2*10^(-11)` pour obtenir ces résultats. C'est pour cela que dans le graphe le paramètre ne devient aussi petit que la grille nous permet (le gradient devient plus petit que `thresh` et donc l'algorithme s'arrête). La valeur par défaut est `thresh = 1e-7`, donc on est plus précis quand on fixe notre valeur `1e-11`.

Une fois que l'on a vérifié nos intuitions sur le comportement du coefficient de l'intercept, on peut aussi vérifier la validité des formules des coefficients lors de centrage et réduction. On représente dans la figure 3 les coefficients θ_0 obtenus à partir des formules. Si nos formules sont exactes, on doit retrouver la figure 2.

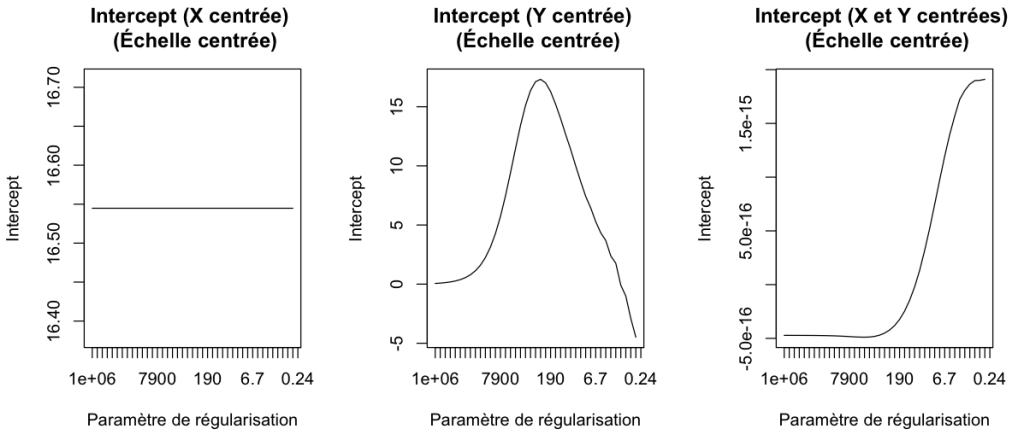


(a) (b) (c)

FIGURE 3 – Intercepts obtenus après utilisation des formules de centrage

On obtient exactement le même résultat que recherché. Cela démontre numériquement la validité de ces formules.

On peut maintenant voir le comportement du coefficient de l'intercept $\tilde{\theta}_0$ dans l'échelle centrée pour chaque cas. C'est ça que montre la figure 4.



(a) (b) (c)

FIGURE 4 – Intercepts obtenus après utilisation des formules de centrage

On vérifie que quand on centre X, le coefficient de l'intercept devient constant et égal à $\bar{Y}$. Quand on centre Y, le coefficient de l'intercept a exactement la même allure qu'avant, mais il est décalé et vaut zéro quand $\tau >> 1$. Cela correspond au fait que pour données Y centrées, $\bar{Y} = 0$. Finalement pour X et Y centrées, même si le graphe semble indiquer une variation on observe que le coefficient de l'intercept est d'ordre 10^{-15}, ce qui est numériquement zéro. C'est-à-dire, centrer X laisse l'intercept constant, et centrer Y met cette constante à zéro.

Dans la prochaine partie, on calcule de différentes manières les coefficients θ_τ des variables explicatives, notamment lorsque $\tau \to 0$ (on utilisera les résultats de la partie 1.3).

3.2 Estimation des coefficients

Regardons maintenant les coefficients autres que l'intercept, c'est-à-dire ceux des variables explicatives. On se place dans le cas où les données X et Y sont centrées et réduites. On peut les calculer manuellement de deux manières : on peut utiliser l'équation qui nous donne A_0 et puis le fait que $\theta_{\tau\to 0} = A_0 X'Y$ (d'après la partie 1.3). Une autre façon est d'essayer l'inversion de la matrice $(X'X + \tau\mathbb{I})$ directement et après la multiplier par $X'Y$ à droite. Pour l'estimation en utilisant le logiciel `R`, on peut utiliser soit `glmnet`, soit la fonction de base `lm.ridge`. On discutera ensuite les différences observées.

On note

$$\theta_{\tau\to 0} := \left(\sum_{j=1}^{r} \frac{1}{\sigma_j^2} v_j v_j'\right) X'Y$$

$$\theta_{\text{inv}} := (X'X + \tau\mathbb{I})^{-1} X'Y$$

où τ vaut la valeur minimale qui fait l'algorithme de `glmnet` converger, et $\theta_{\texttt{glmnet}}$, $\theta_{\texttt{lm.ridge}}$ les

estimations obtenues en utilisant le logiciel `R`. On obtient comme résultats

$$\begin{cases} \theta_{\tau\to 0} & = (-0.3791, -0.4845, 0.1187, 0.3331, 0.1272, \dots) \\ \theta_{\texttt{lm.ridge}} & = (-0.3649, -0.4665, 0.1143, 0.3207, 0.1225, \dots) \\ \theta_{\text{inv}} & = (-0.3617, -0.4627, 0.0939, 0.2933, 0.1051, \dots) \\ \theta_{\texttt{glmnet}} & = (-0.1891, -0.5049, 0.2008, 0.3124, 0.0889, \dots) \end{cases}$$

Sur ces premières valeurs (et les suivantes), on voit que $\theta_{\tau\to 0} \approx \theta_{\texttt{lm.ridge}} \approx \theta_{\text{inv}}$. Cependant, on remarque que $\theta_{\texttt{glmnet}}$ n'approche pas parfaitement les autres. Plus précisément, l'ordre maximal de l'écart absolue entre $\theta_{\tau\to 0}, \theta_{\texttt{lm.ridge}}, \theta_{\text{inv}}$ est de 10^{-2} alors que celui entre $\theta_{\texttt{glmnet}}$ et $\theta_{\tau\to 0}, \theta_{\texttt{lm.ridge}}, \theta_{\text{inv}}$ est de 10^{-1}. Pour comprendre le résultat de $\theta_{\texttt{glmnet}}$, il est important de regarder l'implémentation de `glmnet`.
Cette librairie résout le problème suivant

$$\min_{(\theta_0,\theta)\in\mathbb{R}^{p+1}} \frac{1}{2N} \sum_{i=1}^{N} \left(y_i - \theta_0 - x_i^T\theta\right)^2 + \tau \left[(1-\alpha)\|\theta\|_2^2/2 + \alpha\|\theta\|_1\right]$$

On est dans le cas où l'intercept n'est pas pénalisé (car variables centrées), et $\alpha = 0$ (pour la régression ridge). Donc `glmnet` résout

$$\min_{\theta\in\mathbb{R}^{700}} \frac{1}{2N} \sum_{i=1}^{N} \left(y_i - x_i^T\theta\right)^2 + \frac{\tau}{2}\|\theta\|_2^2$$

C'est-à-dire, le paramètre de régularisation de `glmnet` est le paramètre de régularisation de notre formulation originale multiplié par N. C'est pourquoi dans notre code, quand on donne un paramètre de régularisation à `glmnet`, on le divise par N. Un autre paramètre très important est la valeur de `thresh`. Si l'on essaye de le mettre à `thresh = 2*10^(-11)` comme dans les utilisations précédentes de `glmnet`, on obtient un message de non convergence. C'est-à-dire, la valeur de τ minimale qui fait la descente de gradient converger est $\tau = 0.07742/N$. Cette valeur de τ est toujours très grande et on obtient des coefficients très aux autres. Après quelques tests, on vérifie que l'on doit être capable de faire τ au moins N fois plus petit, alors on doit trouver une façon de faire l'algorithme converger pour des valeurs plus petites de τ. C'est pourquoi on augmente la valeur de `thresh`. Cependant, si l'on l'augmente trop, on perd de la précision. Après des tests, on voit que `thresh = 8*10^(-10)` est la plus petite valeur qui fait converger l'algorithme pour $\tau = 0.7742/N^2$. On observe finalement des coefficients θ_{glmnet} similaires aux autres. On note que cette estimation semble s'approcher des autres, mais pas parfaitement.

Notre conclusion est que `glmnet` n'arrive pas à converger avec précision pour notre τ petit, et cela doit être un problème avec la méthode de descente de gradient utilisée par cette librairie utilise pour trouver la solution.

3.3 Validation croisée et erreur de généralisation

On vient d'étudier le comportement du coefficient de l'intercept θ_0 quand le paramètre de régularisation τ varie, et on a aussi obtenu les coefficients θ en utilisant plusieurs méthodes lorsque $\tau \to 0$. Maintenant, on se demande quelle valeur de τ choisir selon un critère d'optimalité. Pour

répondre cela, en général on dispose de deux solutions : soit on sépare nos données en échantillon apprentissage et échantillon test pour calculer l'erreur de prédiction sur la partie de test, soit on fait validation croisée. On choisit la deuxième méthode, car nos observations ne sont pas nombreuses et on souhaite calculer une erreur de généralisation.

On sépare nos données en quatre plis. Pour chaque valeur de τ, on utilise trois de ces plis pour entraîner un modèle, et ensuite on fait une prédiction pour le pli non utilisé dans l'entraînement. On répète cela pour pour chaque pli. Ensuite, on moyenne les erreurs et cela nous donne une estimation de l'erreur de prédiction pour chaque choix de τ. On prend le τ qui minimise cette erreur. C'est donc un critère d'optimalité qui permet de choisir la valeur de τ.

Au niveau du code, on fera cette procédure manuellement en utilisant la fonction `cvsegments` du package `pls` pour créer les plis. Le choix des plis est reproductible car on fixe la graine du générateur aléatoire avec `set.seed(1)`. Ensuite, on comparera nos résultats avec ceux obtenus en utilisant la fonction `cv.glmnet`.

Voici le résultat obtenu en réalisant manuellement la procédure décrite. Le graphe de l'évolution de l'erreur moyenne est représenté dans la figure 5. La valeur optimale trouvée est $\tau = 0.07225$. Les lignes rouges forment des intervalles de confiance pour l'erreur moyenne. Pour les calculer, on utilise une variante pondérée de l'estimateur empirique de la variance avec dénominateur $N - 1$ des erreurs pour chaque pli, et ensuite on prend sa racine carrée. On somme un écart-type à l'erreur pour obtenir la courbe supérieure, et on soustrait un écart-type pour obtenir la courbe inférieure. L'erreur estimée pour ce choix de τ est 0.3303. On remarque que les intervalles de confiance sont visuellement fins le long de la grille.

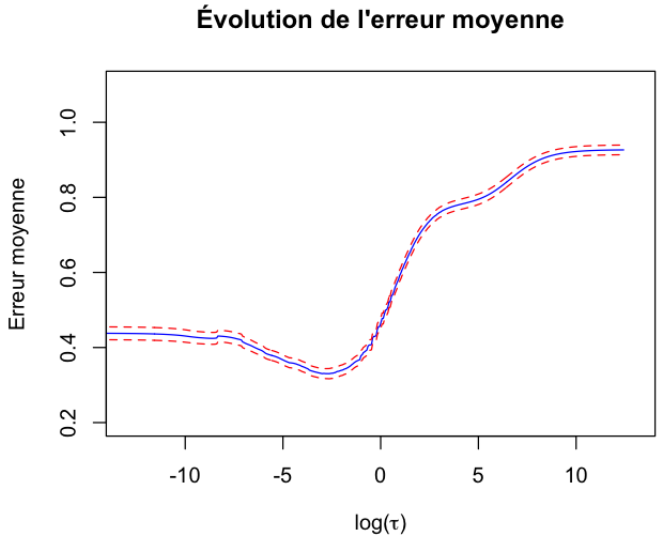


FIGURE 5

Maintenant analysons le résultat obtenu en utilisant `cv.glmnet`. Le graphe de l'évolution de l'erreur moyenne est représenté dans la figure 6. La valeur optimale trouvé est $\tau = 0.08146$. L'erreur

estimée pour ce choix de τ est 0.3666.

On observe que les résultats pour τ, l'erreur de prédiction estimée sont approximativement les mêmes. Par ailleurs, on remarque que l'allure des graphes est à peu près la même. Une différence est la taille des intervalles de confiance autour de l'erreur estimée. Nous avons regardé et essayé de reproduire le code source dans `cv.glmnet` pour les calculer, cependant nos intervalles manuels sont beaucoup plus serrés.

On peut interpréter l'allure de ces graphes. Ils nous disent que trop de pénalisation contraint trop les coefficients et cela diminue la puissance prédictive de notre modèle. Si la pénalisation est trop faible, on s'approche du modèle linéaire qui est mal conditionné et l'erreur n'est pas optimale non plus. On trouve l'erreur optimale avec une pénalisation modérée.

Une fois qu'on a choisi un τ, on ajuste un dernier modèle avec ce paramètre de régularisation, on fait une prédiction sur les données de validation et on calcule l'erreur de généralisation. On obtient une erreur de généralisation de 0.2160674, dans l'échelle des données réduites. Dans l'échelle des données originales, cette erreur est de 1.415875. On a pu calculer une erreur de généralisation car les jeux de données `xtest` et `ytest` n'ont pas servis dans la construction du modèle (modélisation et choix de τ).

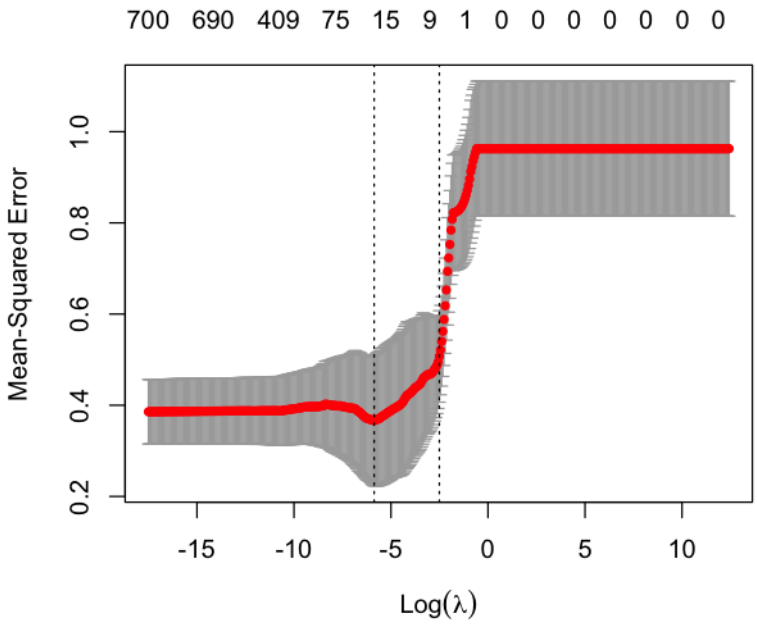


FIGURE 6

Remarque sur le choix de τ : la fonction `cv.glmnet` nous propose 2 valeurs de τ : `lambda.min` qui minimise l'erreur et `lambda.1se` qui est la plus grande valeur de τ telle que l'erreur calculée est à 1 écart-type de distance de l'erreur optimale calculée avec `lambda.min`. On a utilisé `lambda.1se` car l'erreur de généralisation calculée est meilleure. C'est d'ailleurs ce que recommande les auteurs de la

librairie. On pense que le résultat est meilleur car le modèle avec `lambda.1se` sur-apprend moins qu'avec `lambda.min` (`lambda.min` < `lambda.1se` donc la pénalisation est plus importante).

4 Régression logistique pénalisée

On s'intéresse maintenant au dépassement du seuil de 18 au lieu de la teneur en sucre. La variable réponse n'est donc plus une variable quantitative mais une variable qualitative. On transforme donc la variable Y de la teneur en sucre en une variable binaire $\tilde{Y}$ dans $\{0, 1\}$. Elle est construite par l'indicatrice de dépassement de seuil de 18 :

$$\tilde{Y}_i = \begin{cases} 1 \text{ si } Y_i > 18 \\ 0 \text{ sinon} \end{cases}$$

On va donc utiliser la régression logistique pour estimer la probabilité de dépassement de seuil. Dans un premier temps, on va définir le modèle choisit. Dans un second temps, on l'applique à nos données et enfin, on étudiera nos résultats.

4.1 Définition du modèle

On note $\tilde{Y}$ la variable réponse binaire à valeurs dans $\{0, 1\}$ et X la matrice des variables explicatives.

La régression logistique est un modèle tel que

— Y_k suit une loi binomiale $\mathcal{B}(n_k, \pi_k)$ pour $k = 1, ..., K$

— L'espérance de Y_k est une fonction non linéaire d'un régresseur linéaire. Le régresseur linéaire est donc une fonction g non linéaire de l'espérance, appelée fonction de lien :

$$g(E[Y_i]) = X_i\theta$$

Celle que l'on utilise est la fonction `logit()`.

— Les variables explicatives X_i sont indépendantes.

Lorsque les n_k sont différents de 1, on est dans le cadre de *données groupées*. Dans notre cas, on a $n_k = 1$ car les observations sont quantitatives et sans répétition. On est alors dans le cadre de *données individuelles*.

On étudie ensuite l'équilibre de nos données d'entraînement `z` et de test `ztest`. Pour ce faire, on calcule la table de confusion avec la fonction `table()`. Voici les résultats :

0	1
24	16

Table de confusion de z

0	1
19	13

Table de confusion de $ztest$

La proportion observée dans z est 0.6 et celle de $ztest$ est d'environ 0.59 donc les proportions sont quasi similaires dans les deux jeux de données. Les jeux de données z et $ztest$ ne sont pas équilibrés mais les occurrences observées de 0 et de 1 ne sont pas rares. Les jeux de données sont donc déséquilibrés mais de manière contenue.

On réfléchit maintenant au modèle que l'on va utiliser. On souhaite utiliser le modèle de régression logistique mais un problème peut nuire à notre modèle : le sur-apprentissage. En effet, dans notre cas de grande dimension, on possède beaucoup de variables par rapport au nombre d'individus. Ainsi, si on utilise la régression logistique, elle risque de sur-apprendre les données et de créer un estimateur avec beaucoup de variance.

C'est pourquoi on choisit d'utiliser la régression logistique pénalisée. La régularisation va introduire du biais dans le modèle en pénalisant les coefficients mais elle va réduire la variance d'un modèle logistique simple. L'estimateur pénalisé sera moins performant pour modéliser les données d'entraînement mais sera plus performant pour prédire de nouvelles données.

4.2 Application de la régression logistique pénalisée

Nous allons utiliser la régression logistique pénalisée en ridge et en lasso. Pour déterminer la meilleure valeur du coefficient de régularisation λ, on va utiliser la fonction `cv.glmnet`. Etant donné que l'on se situe dans le cadre d'une régression logistique pénalisée, on utilise le paramètre `family = "binomial"`.

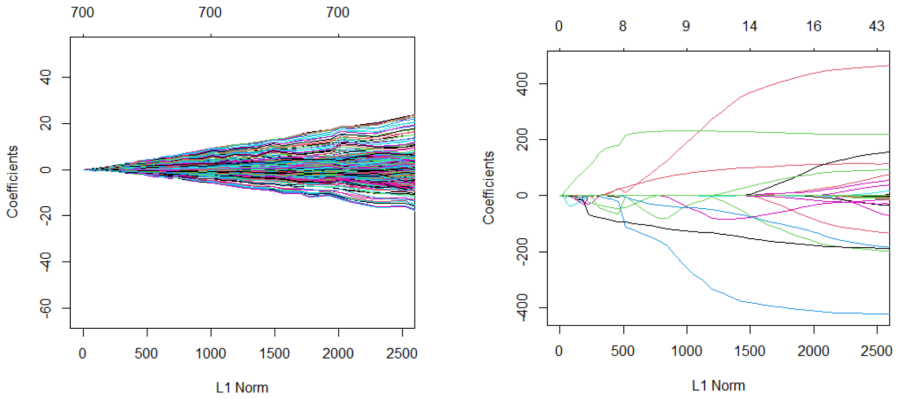
Cette fonction détermine λ tel qu'il minimise une erreur en validation croisée. L'erreur que l'on utilise est la déviance d'un modèle logistique avec le paramètre `type.measure="deviance"` car on se situe dans un modèle logistique pénalisé. On pose $k = 5$ le nombre de plis de la validation croisée et la grille de recherche de λ est composée de 100 valeurs uniformément réparties entre 10^{-6} et 10^{3}.

On note λ_{ridge} et λ_{lasso} les coefficients de régularisation optimaux dans le cas ridge et le cas lasso. On obtient $\lambda_{ridge} = 5.33e-03$ et $\lambda_{lasso} = 6.58e-035$. On peut donc calculer les estimateurs de régression logistique pénalisée ridge et lasso avec les coefficients λ optimaux.

4.3 Etude des résultats

On va comparer les régression ridge et lasso puis on compare les 2 classifieurs selon la régression utilisée.

Tout d'abord, on compare les régressions ridge et lasso. Les régressions ridge et lasso ont pour objectif de pénaliser les coefficients. La particularité qui les différencie est la norme utilisée dans la contrainte $\|\theta\| \leq \delta$. La régression lasso utilise la norme ℓ_1 qui favorise les cas de nullité de composantes, contrairement à la régression ridge qui utilise la norme ℓ_2. Pour illustrer cette différence, on représente les coefficients en fonction de leur norme ℓ_1 :



(a) Cas ridge (b) Cas lasso

FIGURE 7 – Coefficients en fonction de leur norme L1

A gauche, on voit que la régression ridge augmente progressivement la valeur des coefficients sans les sélectionner. En effet, si on regarde les valeurs de l'axe en haut du graphique, qui représente le nombre de coefficients non nuls, on voit qu'aucun sont nuls (valeur de 700 le long de l'axe). On peut l'observer plus précisément grâce au paramètre `nzero` de l'objet `cv.glmnet()`. On voit que `nzero` = 700 pour tous les λ de la grille.

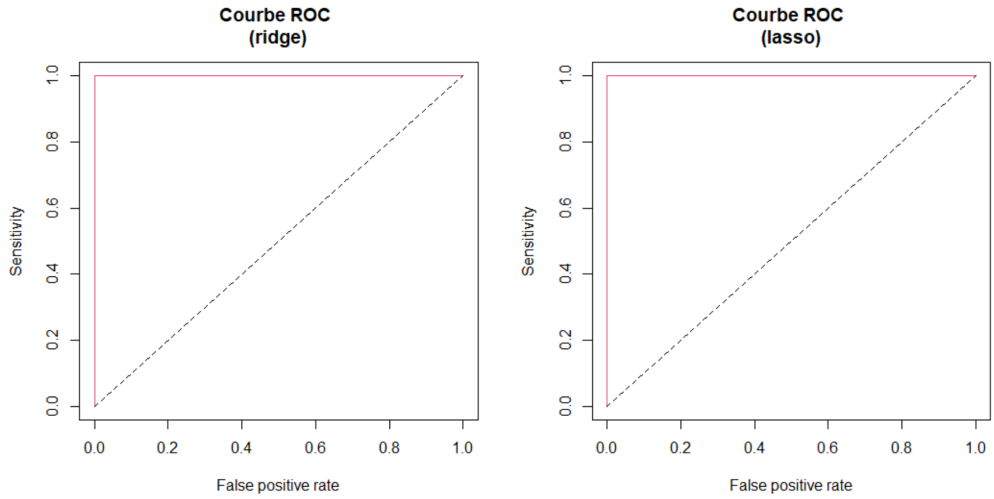
A droite, on voit que la régression lasso augmente elle aussi progressivement les coefficients mais elle les sélectionne et/ou déselectionne. On voit sur l'axe en haut du graphique que le nombre de coefficients non nuls diminuent lorsque la norme ℓ_1 des coefficients diminue. Autrement dit, plus la pénalisation est forte et plus la régression lasso est sélective. Comme précédemment, on peut observer cela grâce au tableau `nzero`. Pour le cas particulier optimal $\lambda = \lambda_{lasso}$, on a `nzero` = 6, c'est-à-dire que seuls 6 variables sur 700 interviennent dans le modèle. Voici les coefficients non nuls :

X2	X317	X318	X440	X488	X640
5.69	-0.56	-56.94	-80.10	167.21	-38.48

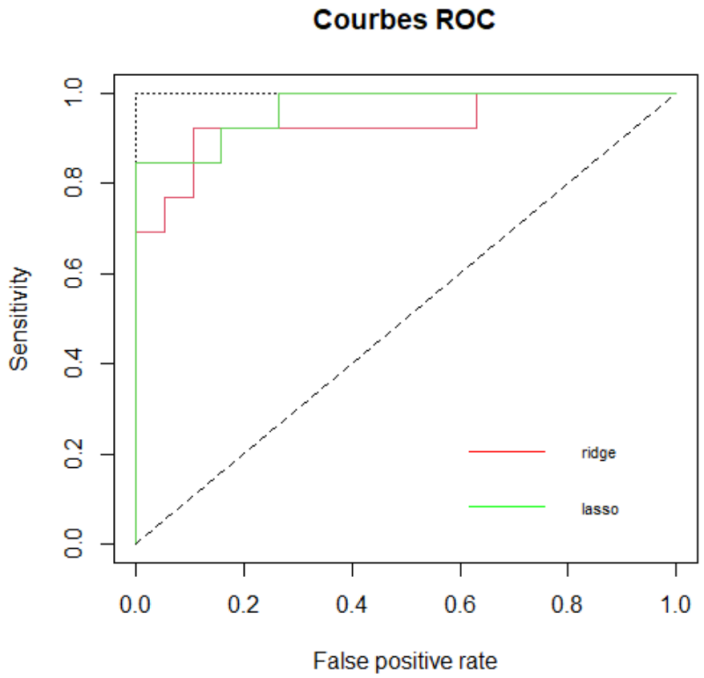
Cette comparaison des résultats des régressions ridge et lasso est intéressante dans notre cas de grande dimension. En effet, nous pensons que la régression lasso va être plus performante que la régression ridge car nous avons beaucoup de variables par rapport au nombre d'individus. La régression lasso permet de réduire drastiquement le nombre de variables dans le modèle (6 au lieu de 700) et donc va réduire le risque de surapprentissage et baisser la variance. De plus, le modèle trouvé est plus simple et plus interprétable car il y a peu de variables utilisées.

Maintenant, on compare les 2 classifieurs retenus pour déterminer le meilleur des 2. On obtient une règle de classification $h(x) = \mathbb{1}_{\tilde{\eta}(x)>s}$ où $s \in [0,1]$ est le seuil et $\tilde{\eta}$ est l'estimateur de la régression ridge pénalisée (ridge ou lasso). On note $\tilde{\eta}_{ridge}$ et $\tilde{\eta}_{lasso}$ pour les estimateurs de chaque régression.

Pour comparer ces 2 règles de classification et déterminer la meilleure, on utilise la courbe ROC. Elle représente le `True Positive Rate` (proportion de positifs détectés pour un seuil s) en fonction du `False Positive Rate` (proportion de scorés positifs parmi les négatifs). Premièrement, on représente les courbes ROC calculées sur les données d'entraînement :



On remarque que dans les 2 cas, la courbe ROC épouse parfaitement la courbe ROC de l'estimateur parfait (la courbe composée des segments $(0;0) - (0;1)$ et $(0;1) - (1;1)$). Cela veut dire que les 2 classifieurs modélisent parfaitement le dépassement de seuil de 18. Cependant, ce qui nous intéresse, c'est d'étudier le score de prévision des 2 classifieurs. C'est pourquoi on représente cette fois les courbes ROC calculées sur les données de test :



Une règle de classification est d'autant meilleure que sa courbe ROC est proche de celle de la

règle parfaite. Pour comparer les 2 courbes ROC, on peut effectuer :

— une comparaison locale. Ici les 2 courbes s'entremêlent donc on ne peut pas désigner une règle globalement meilleure. On peut le faire uniquement localement pour des seuils appartenant à des sous-intervalles de $[0, 1]$.
On remarque cependant que c'est la courbe ROC associée à $\tilde{\eta}_{lasso}$ qui est souvent au-dessus de celle associée à $\tilde{\eta}_{ridge}$. On conclut donc que l'estimateur de la régression lasso est meilleur que celui de la régression ridge.

— une comparaison globale. On peut aussi calculer l'aire sous la courbe ROC pour déterminer un score sur la performance globale d'une règle de classification et pour en comparer plusieurs, notamment si l'étude locale précédente ne permet pas de les départager à l'oeil nu. Dans notre cas, on a $\texttt{auc}_{ridge} = 0.931$ et $\texttt{auc}_{lasso} = 0.968$. Cela confirme le fait que l'estimateur associé à $\tilde{\eta}_{lasso}$ est meilleur que celui associé à $\tilde{\eta}_{ridge}$.

Enfin, si on compare les estimateurs $\tilde{\eta}_{ridge}$ et $\tilde{\eta}_{lasso}$ avec l'estimateur parfait, on remarque qu'ils sont tous les deux très bons. En effet, leur courbe ROC sont très proches de la courbe ROC de l'estimateur parfait et leur `auc` sont très proches de 1. On pense que la régression lasso est plus performante grâce à son comportement de sélection de variables. Etant donné qu'on est en grande dimension, elle permet d'empêcher le surapprentissage et de réduire la variance de l'estimateur.

La détermination des coefficients optimaux λ_{ridge} et λ_{lasso} dépend du choix des folds de la validation croisée. Cette dernière étant aléatoire, on peut obtenir des résultats sensiblement différents. La régression lasso donne tout de même de meilleur résultat sur nos différents essais.

Finalement, étant donné que l'on effectue des régressions logistiques en données individuelles, on ne connaît pas la loi asymptotique de la déviance et on ne peut donc pas tester l'adéquation des modèles.

5 Appendice : décomposition en valeurs singulières

On montre dans cette section l'existence de la décomposition en valeurs singulières de X, une matrice $n \times p$ de rang r.

5.1 Décomposition en somme de produits de Kronecker

On montre d'abord qu'il existe une famille orthonormée de vecteurs $\{u_j\}$ et des scalaires $\lambda_j > 0$ tels que $XX' = \sum_{j=1}^{r} u_j v_j'$. Observons que la matrice XX' est symétrique, car une composante quelconque s'écrit

$$(XX')_{ij} = \sum_{k=1}^{p} x_{ik} x'_{kj} = \sum_{k=1}^{p} x_{ik} x_{jk}$$

Si l'on échange i et j on voit que

$$(XX')_{ji} = \sum_{k=1}^{p} x_{jk} x_{ik} = \sum_{k=1}^{p} x_{ik} x_{jk} = (XX')_{ij}$$

d'où la symétrie.

Lemma 1 (Théorème spectral pour les matrices). *Soit A une matrice symétrique réelle, alors il existe une matrice U orthogonale et une matrice Λ diagonale dont tous les coefficients sont réels, telles que*

$$XX' = U\Lambda U'$$

Plus spécifiquement, on sait que les colonnes de U sont les vecteurs propres orthonormés de XX', i.e., $U = (u_1 \dots u_n)$, et $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$ sont ses valeurs propres. On vérifie facilement que ces valeurs propres sont non-négatives. Si l'on prend v un vecteur $n \times 1$, on a

$$v'XX'v = (X'v)'X'v = \|X'v\|_2^2 \geq 0$$

donc XX' est positive sémi-définie et ses valeurs propres sont non-négatives.

On réécrira l'équation $XX' = U\Lambda U'$ dans une autre forme. À droite, si l'on fait directement le calcul d'une composante on voit que

$$(XX')_{ij} = \sum_{k=1}^{n} (u_k)_i \lambda_k (u_k)_j$$

en particulier on peut sommer seulement sur les $\lambda_k > 0$. Rappelons que le nombre de valeurs propres non-nulles de XX' est égal à $\text{rang}(XX')$, car le rang d'une matrice est préservé par changement de base. Ainsi, si l'on change vers la base de valeurs propres, XX' "est" la matrice Λ, qui est évidemment de rang égal au nombre de valeurs propres non-nulles.

$$(XX')_{ij} = \sum_{k=1}^{\text{rang}(XX')} (u_k)_i \lambda_k (u_k)_j$$

D'autre part, on note qu'on peut écrire le produit des composantes comme une composante du produit de Kronecker :

$$(u_k)_i \, (u_k)_j = \left(u_k u_k'\right)_{ij}$$

Donc si l'on introduit $v_k := \lambda_k u_k$, la somme précédente devient

$$XX' = \sum_{k=1}^{\text{rang}(XX')} u_k v_k'$$

La dernière simplification est une conséquence de la proposition 1.1., qui nous permet de sommer effectivement jusqu'au rang de X.

$$\boxed{XX' = \sum_{k=1}^{r=\text{rang}(X)} u_k v_k'}$$

5.2 Considérations sur les projections

Voici deux propositions importantes.

Proposition 5.1. *La matrice $\sum_{i=1}^{r} u_i u_i'$ est une matrice de projection de* $\mathrm{Im}(XX')$

Démonstration. On applique cette matrice sur un vecteur v :

$$\begin{aligned}\sum_{i=1}^{r} u_i u_i' v &= \sum_{i=1}^{r} u_i (u_i' v) \\ &= \sum_{i=1}^{r} u_i \langle u_i, v \rangle \\ &= \sum_{i=1}^{r} u_i \frac{\langle u_i, v \rangle}{\langle u_i, u_i \rangle}\end{aligned}$$

où le produit scalaire est usuel et la dernière équation suit du fait que les u_i sont normés. Comme v est arbitraire, on conclut que l'application $\sum_{i=1}^{r} u_i u_i'$ est la projection de v dans le sous-espace $[u_1, \ldots, u_r]$ engendré par les vecteurs propres de valeurs propres non-nulles. Cet espace est bien l'image de XX', car la décomposition spectrale sépare l'espace en noyau de XX' (engendré par les vecteurs propres de valeurs propres nulles) et son image (engendrée par les vecteurs propres des valeurs propres non-nulles). □

On aura besoin du lemma suivant.

Lemma 2. *Le noyau d'une matrice est le complément orthogonal de l'image de sa transposée :*

$$\ker(X) = \mathrm{Im}(X')^{\perp}.$$

Démonstration. Pour le sens gauche-droite, on doit montrer que $\forall v \in \mathrm{Im}(X'), \forall w \in \ker(X)$ alors $\langle v, w \rangle = 0$. Les vecteurs dans $\mathrm{Im}(X')$ sont de la forme $X'v$ pour tout v. Soit $w \in \ker(X)$. On doit montrer que $\langle X'v, w \rangle = 0$. C'est simple, car $\langle X'v, w \rangle = \langle v, Tw \rangle = 0$.

Pour l'autre sens, i.e., $(\ker(X))^{\perp} \subset \mathrm{Im}(X')$, c'est équivalent montrer que $\mathrm{Im}(X')^{\perp} \subset \ker(X)$. Donc, soit $x \in \mathrm{Im}(X')^{\perp}$ et y quelconque. Or,

$$0 = \langle x, X'y \rangle = \langle Xx, y \rangle$$

Comme y est arbitraire et le produit scalaire n'est pas dégénéré, cela implique $x \in \ker(T)$. □

Proposition 5.2. *Les projections sur Im(X) et Im(XX') sont identiques.*

Démonstration. On montre que les deux images sont le même sous-espace. On sait par le lemme que $\ker(X') = \mathrm{Im}(X)^{\perp}$, alors $\ker(X')^{\perp} = \mathrm{Im}(X)$ et $\mathrm{Im}(XX') = \ker(XX')^{\perp}$. Or, on a déjà montré que $\ker(X') = \ker(XX')$, d'où l'égalité des images de X et XX'. □

5.3 La décomposition en valeurs singulières

On aura besoin du lemma suivant.

Proposition 5.3. *Les vecteurs définis par* $v_j := \lambda_j^{-1/2} X' u_j$ *pour* $j = 1, \dots, r$ *forment une famille orthonormée de vecteurs propres de* $X'X$.

Démonstration. Ces vecteurs sont normés :

$$\begin{aligned} v_j' v_j &= \left(\lambda_j^{-1/2} u_j' X\right) \cdot \left(\lambda_j^{-1/2} X' u_j\right) \\ &= \lambda_j^{-1} \left(u_j' X X' u_j\right) \\ &= \lambda_j^{-1} \lambda_j u_j' u_j = \lambda_j^{-1} \lambda_j = 1 \end{aligned}$$

Ce même calcul nous montre qu'ils sont aussi orthogonaux, car les u_i le sont, i.e., car si $i \neq j$ alors $u_i' u_j = 0$.

Ces vecteurs sont bien des vecteurs propres de $X'X$:

$$\begin{aligned} X'X v_j &= X'X \left(\lambda_j^{-1/2} X' u_j\right) \\ &= \lambda_j^{-1/2} X' \lambda_j u_j \\ &= \lambda_j v_j \end{aligned}$$

□

On arrive au résultat principal.

Theorem 5.4 (Décomposition en valeurs singulières)**.** *Soit* X *une matrice de dimension* $n \times p$ *et rang* r, $\{u_j\}_{1,\dots,r}$ *les vecteurs propres de* XX' *avec valeur propre strictement positive, et* $v_j := \lambda_j^{-1/2} X' u_j$. *Alors* X *se décompose comme*

$$X = \sum_{j=1}^{r} \sqrt{\lambda_j} u_j v_j'$$

Cette équation peut être écrite en forme matricielle comme

$$X = U \Lambda^{\frac{1}{2}} V'$$

Les v_j *peuvent être calculés comme les vecteurs propres de rang positif de la matrice* $X'X$.

Démonstration. Or, on peut écrire

$$u_j v_j' = \lambda_j^{-1/2} u_j u_j' X$$

alors si l'on multiplie par $\lambda^{1/2}$ et somme sur j, on obtient

$$\sum_{j=1}^{r} \lambda_j^{1/2} u_j v_j' = \left(\sum_{j=1}^{r} u_j u_j'\right) X = PX$$

où P est la matrice de projection sur $\mathrm{Im}(X)$, comme on a vu dans la proposition 2.2. et proposition 2.3. Mais $P|_{\mathrm{Im}(X)} = \mathrm{Id}$, c'est-à-dire, $PX = X$, d'où le résultat. □

Conclusion

Dans ce projet, nous avons étudié le jeu de données `cookies`, qui est un exemple de données en grande dimension. Cela nous a permis d'appréhender et d'étudier ce genre de données et de comprendre l'intérêt de la pénalisation ridge et lasso sur les données en grande dimension.

Dans les première et cinquième parties, nous avons développés des notions théoriques que nous avons retrouvées et vérifiées en pratique dans la partie 3. Plus précisément, nous avons étudié la régression ridge, les coefficients calculés grâce à elle et les effets des transformations des données. Dans la partie 3, nous avons également appliqué une recherche concrète de modèle ridge optimal sur nos données par validation croisée. Une autre application concrète a été faite dans la partie 4 où nous avons transformé la variable réponse sous forme qualitative. Cela nous a permis d'étudier et d'appliquer la régression logistique en ridge et en lasso. Finalement, dans la partie 2, nous avons étudié nos données par différentes analyses.

Références

[1] Brian G. Osborne et al. “Application of near infrared reflectance spectroscopy to the compositional analysis of biscuits and biscuit doughs”. In : *Journal of the Science of Food and Agriculture* 35.1 (1984), p. 99-105. doi : https://doi.org/10.1002/jsfa.2740350116. url : https://onlinelibrary.wiley.com/doi/abs/10.1002/jsfa.2740350116.

[2] Cornillon Pierre-André. *Régression : théorie et applications*. fre. Statistique et probabilités appliquées. Paris : Springer, DL 2006, cop. 2007. isbn : 2-287-39692-6.

[3] Keribin Christine. *Apprentissage statistique supervisé et non supervisé*. Paris : ENSTA.

[4] A. E. Hoerl et R. W. Kennard. “Ridge Regression : Biased Estimation for Nonorthogonal Problems”. In : *Technometrics* 12 (1970), p. 55-67.

[5] Trevor Hastie, Robert Tibshirani et Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA : Springer New York Inc., 2001.

[6] *Fiche WikiStat : Scénario : Calibration (1-cookies) de spectres NIR*. https://www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-scenar-app-cookie.pdf.