INSTITUT DES ACTUAIRES

ISUP

Mémoire présenté le :
pour l'obtention du diplôme
de Statisticien Mention Actuariat
et l'admission à l'Institut des Actuaires

Par : Damien FABRE RUDELLE

Titre du mémoire : Apport des méthodes d'apprentissage statistique pour le provisionnement individuel en assurance non-vie

Confidentialité : ☒ NON ☐OUI (Durée : ☐ 1 an ☐2 ans)

Les signataires s'engagent à respecter la confidentialité indiquée ci-dessus

Membres présents du jury de l'Institut des Actuaires signature

Membres présents du jury de la filière

Entreprise :

Nom : KPMG

Signature :

Directeur de mémoire en entreprise :

Nom : Salma JAMAL

Signature :

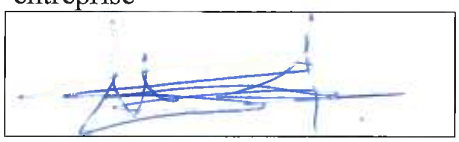
Invité :

Nom : Fabrice OGER

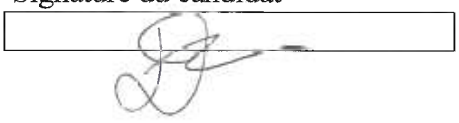
Signature :

Autorisation de publication et de mise en ligne sur un site de diffusion de documents actuariels *(après expiration de l'éventuel délai de confidentialité)*

Signature du responsable entreprise



Signature du candidat



Résumé

Mots clés : Provisionnement, Apprentissage statistique, GLM, Elastic Net, Forêts aléatoires, Réseaux de neurones, Bagging, Chain-Ladder.

Dans un contexte où l'information est plus que jamais omniprésente et disponible, les assureurs non-vie cherchent à développer de nouvelles méthodes de provisionnement utilisant des données jusqu'alors peu exploitées. Ces méthodes ont plusieurs objectifs : d'une part obtenir des provisions plus fines et précises que celles obtenues par des méthodes agrégées comme Chain-Ladder, et d'autre part mieux évaluer leurs risques et mieux connaître leurs portefeuilles de sinistres.

Les enjeux de la bonne prédiction de l'évolution du coût des sinistres sont partie intégrante de la maîtrise des risques, notamment pour les branches à duration longue comme la Responsabilité Civile Corporelle. Ces enjeux peuvent être aussi bien réglementaires, stratégiques que financiers, et incitent les assureurs non-vie à se questionner sur l'utilisation de nouvelles approches de provisionnement.

Dans ce mémoire, nous nous intéressons à l'élaboration d'un modèle de provisionnement individuel, c'est à dire sinistre par sinistre et non pas en vision agrégé comme une méthode Chain-Ladder. Notre objectif est de mieux tirer partie d'informations individuelles. Nous utiliserons notamment une base de données réelles, fournie par un assureur français, comportant de nombreuses variables descriptives sur les sinistres : le caractère grave ou non du sinistre, les caractérisques des blessures, ou bien encore le taux d'AIPP.

Nous commencerons par un état de l'art sur les méthodes de provisionnement en assurance non-vie, aussi bien sur celles classiques agrégées que sur celles individuelles.

Nous formaliserons par la suite notre problème de provisionnement et la modélisation utilisée. Nous utiliserons des méthodes classiques en assurance non-vie, comme les GLM pénalisés, ainsi que des méthodes d'apprentissage statistique permettant de capter des interactions non-linéaires : les forêts aléatoires et les réseaux de neurones.

Nous comparerons notamment les résultats obtenus par nos modèles aux données réelles et aux prédictions obtenues par une méthode Chain-Ladder. Nous proposerons aussi des pistes de réflexion sur l'utilisation des prédictions individuelles.

La synthèse de ce mémoire viendra finalement décrire les apports et les limites de nos travaux, tout en proposant une ouverture sur de potentielles études futures.

Abstract

Keywords : Reserving, Machine Learning, GLM, Elastic Net, Random forest, Neural networks, Bagging, Chain-Ladder.

In times where data is ever more omnipresent and available, non-life insurers are searching for new reserving methods capable of using data that is not commonly used. These methods have several purposes : estimate reverses more accurately than by using traditional methods such as Chain-Ladder, but also enhance their knowledge on the risks to which they are exposed.

A precise evaluation of is a main part in a good risk management process, especially for long duration segments, such as the risk of corporal injury. Insurers try to find new reserving methods to tackles everchanging challenges such as reglementary, strategic or financial conundrums.

In this study, we will focus on the implementation of an individual reserving method. We aim at using the full potential of individual data, and we'll base our work on a dataset from a french insurer. It includes detailed claim-level data with many explanatory variables, such as the whether the claim is considered large or not, information on the injuries and the disability rate.

In the first part, we will begin by describing traditional reserving methods used in non-life insurance, and we will distinguish aggregate-level and individual-level methods.

In the following part, we shall formalise our reserving methodology. We will begin by commonly used methods such as penalised GLM, and will also use machine learning methods capable of detecting non-linear interactions, such as random forests and neural networks.

We will then compare all the obtained results with real data and with predictions resulting from the use of a tradition reserving method, Chain-Ladder. We will also inquire on the uses of individual predictions.

Finally, this paper synthesis will describe the contribution of our study and propose improvements for future potential works.

Remerciements

Ce mémoire est le résultat d'une grande aventure, et n'aurait pas été sous sa forme actuelle sans le concours de nombreuses personnes.

Je n'oserais commencer ces remerciements sans exprimer toute la gratitude que j'ai envers celle qui m'a suivi, encadré et soutenu tout au long de cette étude. Salma Jamal, de par ta disponibilité, ton expertise aussi bien sur les méthodes d'apprentissage statistique que sur le provisionnement en assurance non-vie, sans oublier ta bonne humeur permanente, tu laisseras un souvenir impérissable. J'espère pouvoir un jour te rendre tout le temps que tu m'as consacré !

Olivier Constantin, merci de m'avoir fait rejoindre cette belle équipe qu'est l'équipe Actuariat de KPMG. Tes connaissances en assurance non-vie ont pu éclairer des coins sombres de notre problématique de provisionnement et permettre de les explorer.

Je remercie aussi la MAIF d'avoir permis cette étude en ayant autorisé l'utilisation de leurs données, et plus particulièrement mes interlocuteurs et partenaires dans ce travail, Fabrice Oger et Loubna Niny, pour les nombreux échanges et leurs remarques pleines de bon sens et de perspicacité.

J'adresse mes plus sincères remerciements au directeur de l'ISUP, Olivier Lopez, ainsi qu'à toute l'équipe pédagogique de l'UPMC pour m'avoir transmis leur savoir tout au long de ces trois années de formations.

Enfin, je tenais à remercier mes parents pour leur soutien sans faille.
À toi, Joanna, qui illumine ma vie.

Table des matières

Introduction

Afin de pouvoir couvrir leurs engagements, les assureurs doivent établir des provisions, notamment pour les sinistres survenus dont les coûts ne sont pas totalement réglés. Les techniques traditionnelles pour quantifier ces provisions reposent sur l'agrégation de données individuelles en triangles de développement agrégés.

Les provisions sont une part primordiale dans le bilan d'une société d'assurance. D'après une étude Swiss Re (2008) [40], la principale cause de non solvabilité des sociétés d'assurances américaines de 1969 à 2002 était liée à des pertes causées par des provisions mal estimées. En France, l'évolution de l'environnement réglementaire, de par les normes Solvabilité 2 et IFRS 17, impose un suivi plus précis des risques, notamment par l'estimation des provisions.

Il est important pour les assureurs d'obtenir des estimations fiables de leurs provisions, précises et peu volatiles. Nous sommes donc en droit de nous interroger sur l'adéquation des méthodes classiques de provisionnement aux attentes des assureurs, celles ci étant peu flexibles et ayant tendance à surestimer les provisions.

Les méthodes classiques présentent l'intérêt d'être simples à mettre en place et à utiliser. Toutefois, compte tenu de l'agrégation, les méthodes classiques de provisionnement ne permettent pas de profiter d'une information individuelle, pourtant détaillée et de grande valeur. L'information non utilisée peut aussi bien être des données liées au sinistre et à son développement, que des données détaillées sur l'assuré ou bien également des informations exogènes.
Récemment, de nombreux articles issus de la recherche actuarielle se sont interrogés sur des méthodes de provisionnement au niveau individuel.

De plus, de par le développement récent de méthodes d'apprentissage statistique, ainsi que l'amélioration incessante de la puissance de calcul des ordinateurs, il parait désormais possible de pouvoir utiliser pleinement les immenses bases de données détenues par les assureurs. Ces bases de données comportent généralement de nombreuses informations précieuses sur les assurés et les sinistres déclarés. Elles paraissent donc idéales pour alimenter les modèles de provisionnement actuels.

Dans ce mémoire, nous nous attarderons tout d'abord sur les techniques traditionnelles de provisionnement, puis nous dresserons un état de l'art de la recherche actuarielle sur le provisionnement au niveau individuel.

Nous présenterons par la suite la modélisation et les données utilisées pour la prédiction au niveau individuel, avant de décrire diverses méthodes d'apprentissage statistique, notamment les arbres de régression et les réseaux de neurones, et leur utilisation pour notre modèle. Nous analyserons aussi les résultats obtenus avec la même modélisation par des méthodes GLM avec pénalisation Ridge/Lasso.

Enfin, nous comparerons notre modèle de provisionnement individuel avec d'autres méthodes classiques avant de conclure.

Première partie

Contexte et état de l'art

Chapitre 1

État de l'art du provisionnement en assurance non-vie

Depuis plus d'un siècle, les actuaires ont utilisé en assurance non-vie des triangles de développement pour projeter les paiements et charges futurs liés aux sinistres. Toutefois, la plupart de ces méthodes reposent sur des triangles de données agrégées.

Dans ce chapitre, nous allons tout d'abord présenter les méthodes agrégées, avant de parler des méthodes individuelles. Nous finirons par une comparaison entre méthodes agrégées et individuelles.

1.1 Méthodes agrégées

En 1938, E. Astesan [3] a formalisé la technique des triangles de développement, engendrant ainsi la méthode très populaire dite Chain-Ladder.

En 1993, R. Mack [30] a démontré que les estimateurs de Chain-Ladder pouvaient être modélisés par un simple modèle stochastique. England & Verrall (2002) [13] ont eux fourni un aperçu global des méthodes stochastiques pouvant être connectées à la méthode Chain-Ladder, notamment des méthodes de régression.

Nous allons tout d'abord introduire la notion de triangle de développement avant d'étudier quelques méthodes parmi les plus utilisées en provisionnement non-vie.

1.1.1 Triangle de développement

Les méthodes agrégées reposent sur l'utilisation de triangles de développement (aussi appelés *run-off triangles*). Ces triangles reflètent la dynamique des sinistres et les données qui y sont incorporées peuvent être de différentes natures : paiements, charges, nombre de sinistres...

Nous fixons le cadre et les notations suivants :

— $i \in [0, n]$ l'année de survenance du sinistre,
— $j \in [0, n]$ l'année de développement du sinistre,
— $Y_{i,j,k}$ les paiements réglés durant l'année de développement j pour le k-ième sinistre survenu l'année i,
— $X_{i,j} = \sum_k Y_{i,j,k}$, le cumul des paiements réglés durant l'année de développement j pour les sinistres survenus l'année i.

On considère que l'année n désigne la dernière année d'observation. Ainsi, pour un sinistre survenu en année i, on observe le développement jusqu'en année $j = n - i$. On obtient ainsi un triangle de valeurs agrégées et cumulées (dans le cas présent, des paiements) :

$i \backslash j$	0	1	$\cdots$	j	$\cdots$	$n-1$	n
0	$X_{0,0}$	$X_{0,1}$	$\cdots$	$X_{0,j}$	$\cdots$	$X_{0,n-1}$	$X_{0,n}$
1	$X_{1,0}$	$X_{1,1}$	$\cdots$	$X_{0,0}$	$\cdots$	$X_{1,n-1}$	
$\vdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\ddots$		
i	$X_{i,0}$	$X_{i,1}$	$\cdots$	$X_{i,j}$			
$\vdots$	$\cdots$	$\cdots$	$\ddots$				
$n-1$	$X_{n-1,0}$	$X_{n-1,1}$					
n	$X_{n,0}$						

Nous allons par la suite étudier deux méthodes agrégées très populaires pour le provisionnement déterministe : la méthode Chain-Ladder et la méthode de Bornhuetter-Ferguson. Enfin, nous nous pencherons sur la méthode de Mack pour le Chain-Ladder stochastique.

1.1.2 Méthodes déterministes

1.1.2.1 Méthode Chain-Ladder

La méthode Chain-Ladder est une méthode d'estimation des réserves basée sur un triangle de développement. Nous allons ici étudier cette méthode dans le cas d'un triangle de paiements.

Nous devons tout d'abord définir les paiements cumulés $C_{i,j}$:

$$C_{i,j} = \sum_{l=0}^{j} X_{i,l}$$

En d'autres termes, nous sommons les paiements $X_{i,l} \geq 0$ pour une année de survenance i fixée. On obtient à l'ultime $C_{i,n} = S_i$, où S_i est le coût total à l'ultime des sinistres survenus l'année i.

Principe La méthode Chain-Ladder est une méthode déterministe. Elle repose sur l'idée suivante : les sinistres survenus lors d'années de survenances différentes se développent de façon similaire. De façon concrète :

Hypothèse 1. *Pour $1 \leq i, j \leq n$, soient $(C_{i,j})_{0 \leq i,j \leq n}$ les réglements cumulés. Si les ratios $\dfrac{C_{i,j}}{C_{i,j+1}}$ ne dépendent pas de l'année de survenance i, pour tout $i = 0, \cdots, n$ et $j = 0, \cdots, n-1$, alors il existe des paramètres $\lambda_1, \lambda_2, \cdots, \lambda_{n-1}, \lambda_n$ tels que $C_{i,j+1} = \lambda_j C_{i,j}$.*

En d'autes termes, on considère que les sinistres se déroulent de la même façon peu importe l'année de survenance.

Les facteurs $\lambda_1, \lambda_2, \cdots, \lambda_{n-1}, \lambda_n$ sont appelés *facteurs de développement*, ou bien encore *link ratios*. On estime alors le facteur de développement lié à l'année de développement j par :

$$\lambda_j^* = \frac{\sum_{i=0}^{n-j-1} C_{i,j+1}}{\sum_0^{n-j-1} C_{i,j}}, j = 0, \cdots, n-1$$

Ainsi, sous l'hypothèse 1, au lieu de faire une moyenne simple des facteurs de développement individuels, on estime f_j^* par une moyenne pondérée :

$$\lambda_j^* = \sum_{i=0}^{n-j-1} \frac{C_{i,j}}{\sum_0^{n-j-1} C_{i,j}} \frac{C_{i,j+1}}{C_{i,j}}, j = 0, \cdots, n-1$$

Une fois les facteurs de développement estimés, les paiements cumulés sont à leur tour déterminés par :

$$C_{i,j}^* = \prod_{k=j-1}^{n-i} \lambda_k^* \times C_{i,n-i}$$

On peut donc compléter le triangle grâce aux $C_{i,j}^*$. On obtient ainsi les réserves à l'ultime pour l'année de survenance i :

$$R_i^* = C_{i,n}^* - C_{i,n-i}$$

Ainsi la réserve à l'ultime pour l'ensemble des sinistres est donnée par :

$$R^* = \sum_{i=1}^{n} R_i^*$$

Nous obtenons au final le tableau complété des paiements cumulés suivant :

$i \backslash j$	0	1	$\cdots$	j	$\cdots$	$n-1$	n	R_i^*
0	$C_{0,0}$	$C_{0,1}$	$\cdots$	$C_{0,j}$	$\cdots$	$C_{0,n-1}$	$C_{0,n}$	0
1	$C_{1,0}$	$C_{1,1}$	$\cdots$	$C_{0,0}$	$\cdots$	$C_{1,n-1}$	$C_{1,n-1}^*$	$C_{1,n}^* - C_{1,n-1}$
$\vdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
i	$C_{i,0}$	$C_{i,1}$	$\cdots$	$C_{i,j}$	$C_{i,j+1}^*$	$\cdots$	$C_{i,n}^*$	$C_{i,n}^* - C_{i,j}$
$\vdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$n-1$	$C_{n-1,0}$	$C_{n-1,1}$	$C_{n-1,2}^*$	$\cdots$	$\cdots$	$\cdots$	$C_{n-1,n}^*$	$C_{n-1,n}^* - C_{n-1,1}$
n	$C_{n,0}$	$C_{n,1}^*$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$C_{n,n}^*$	$C_{n,n}^* - C_{n,0}$

Le modèle doit ensuite être validé, par exemple par l'intermédiaire de Q-Q plots (diagrammes Quantile-Quantile) : les $(n-j)$ couples $(C_{i,j}, C_{i,j+1})_{i=0,\cdots,n-j-1}$ doivent être "suffisamment bien" alignés sur une droite passant par l'origine.

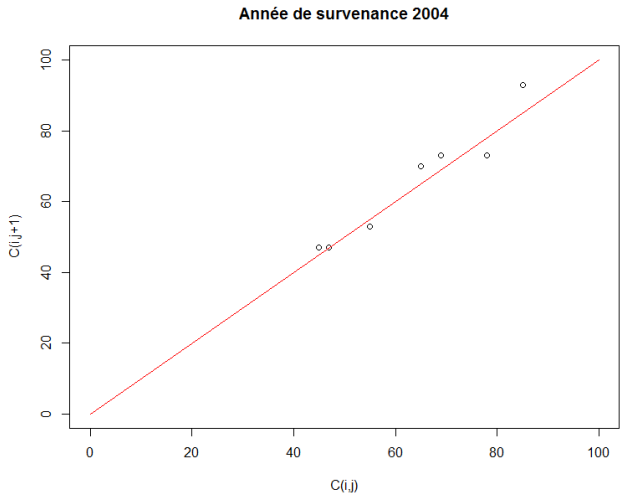


FIGURE 1.1 – Exemple de graphique QQ-plot où les hypothèses Chain-Ladder sont vérifiées

Limites de la méthode Chain-Ladder L'hypothèse d'indépendance utilisée dans la méthode Chain-Ladder est assez restrictive. En effet, les conditions suivantes doivent être réunies pour qu'elle soit valide (Partrat & al., 2007 [38]) :

- Le passé doit être suffisamment régulier. Par exemple, il ne doit pas y avoir de changements importants dans la gestion des sinistres ou dans le taux d'inflation spécifique de la branche ;
- La branche doit être peu volatile : il est compliqué de traiter les sinistres graves, en particulier s'ils sont ponctuels, par cette méthode ;
- Les données du portefeuille doivent être nombreuses et fiables.

L'utilisation très répandue des méthodes type Chain-Ladder est essentiellement due à leur simplicité de mise en œuvre, mais également à leur universalité. En effet, adaptées à des valeurs négatives de montants non cumulés (liées notamment aux recours), elles peuvent s'appliquer à des triangles de charges ou à des triangles de paiements comprenant des recours.

1.1.2.2 Méthode de Bornhuetter-Ferguson

Une autre méthode très utilisée pour le provisionnement non-vie est celle de Bornhuetter-Ferguson, introduite par Bornhuetter R.I. et Ferguson R.E. en 1972 [5]. Cette méthode repose sur une hypothèse exogène d'estimation préalable de la charge à l'ultime, à laquelle on applique un taux de liquidation croissant sur les années de développement. Ainsi, les estimations récentes dépendent moins des premiers paiements qu'avec une méthode du type Chain-Ladder. La méthode Bornhuetter-Ferguson assure donc une meilleure stabilité des estimations et est donc plus adaptée aux triangles dont les incréments sont instables.

Hypothèses La méthode Bornhuetter-Ferguson repose sur un modèle multiplicatif sur les $(C_{i,j})_{i,j=0,\cdots,n}$ qui vérifie :

$$\forall i,j \in \{0,\cdots,n\}, C_{i,j} = \alpha_i \gamma_j$$

où $\alpha_0,\cdots,\alpha_n$ représentent les charges à l'ultime prévisibles pour chaque année de survenance i, et $\gamma_0,\cdots,\gamma_n$ les cadences de paiements cumulés.

Estimation Dans cette méthode, on dispose *a priori* d'estimateurs des charges à l'ultime exogènes : $\alpha_0^*,\cdots,\alpha_n^*$.

On souhaite donc obtenir des estimateurs pour les cadences de développement, de la même façon qu'avec la méthode Chain-Ladder :

$$1-\gamma_j^* = 1 - \frac{1}{\prod_{k=j}^{n} \frac{C_{i,j+1}}{C_{i,j}}}$$

Enfin, les états futurs sont estimés par :

$$C_{i,j}^* := C_{i,n-i} + (\gamma_j^* - \gamma_{n-i}^*)\alpha_i^*$$

Avantages et limites Ce modèle permet, contrairement aux autres méthodes déterministes, d'ajouter de l'information au travers de jugements d'experts pour la détermination des a_i. Toutefois ces données exogènes présentent un inconvénient : elles sont, par nature, subjectives.

Autres méthodes agrégées classiques Parmi les autres méthodes agrégées déterministes utilisées, nous retrouvons par exemple :

- La méthode **London-Chain** : on considère une relation affine entre les $C_{i,j}$ et $C_{i,j+1}$: $C_{i,j+1} = \lambda_j C_{i,j} + \alpha_j$, λ_j et α_j des paramètres réels ;
- La méthode **London-Pivot**, qui est un compromis entre la méthode Chain-Ladder et la méthode London-Chain. On considère cette fois une relation du type $C_{i,j+1} = \lambda_j(C_{i,j} + \alpha_j)$, λ_j et α_j des paramètres réels.

Nous allons par la suite étudier la plus utilisée des méthodes stochastiques : la méthode de Mack.

1.1.3 Méthode stochastique de Mack

Ce modèle introduit par Mack en 1993 [30] est un modèle non-paramétrique qui permet d'estimer une marge d'erreur sur le montant des provisions. Il est aussi appelé *distribution free Chain-Ladder*, car aucune hypothèse de distribution n'est faite sur les composantes du triangle.

Principe La méthode de Mack représente le pendant de la méthode de Chain-Ladder au niveau stochastique.
On suppose que les hypothèses suivantes sont vérifiées :

Hypothèse 2. *Indépendance des années de survenance : pour tout $i \neq i'$, les $(C_{i,j})_{j=0,\cdots,n}$ et $(C_{i',j})_{j=0,\cdots,n}$ sont indépendants*

Hypothèse 3. *Pour $j = 0, \cdots, n-1$, il existe un facteur μ_j tel que pour $i = 0, \cdots, n$ on ait conditionnellement :*

$$\mathbb{E}[C_{i,j+1}|C_{i,1}, \cdots, C_{i,n}] = \mu_j C_{i,j}$$

Hypothèse 4. *Il existe des facteurs $\sigma_1, \cdots, \sigma_n$ tels que pour $i = 0, \cdots, n$:*

$$V[C_{i,j+1}|C_{i,j}, \cdots, C_{i,0}] = V[C_{i,j+1}|C_{i,j}] = \sigma_j^2.C_{i,j}$$

Ainsi, sous ces trois hypothèses, et conditionnellement au triangle agrégé initial $T = \{C_{i,j} = i + j \leq n+1\}$, nous avons :

$$\mathbb{E}[C_{i,n}|T] = \prod_{k=j-1}^{n-i} \mu_k \times C_{i,n-i}$$

On peut estimer les facteurs μ_j de façon non biaisée par les estimateurs non corrélés :

$$\mu_j^* = \frac{\sum_{i=0}^{n-j} C_{i,j+1}}{\sum_{i=1}^{n-j} C_{i,j}}$$

Nous obtenons de façon similaire à la méthode Chain-Ladder la charge à l'ultime estimée :

$$C_{i,j}^* = \prod_{k=j-1}^{n-i} \mu_k^* \times C_{i,n-i}$$

$$R_i^* = C_{i,n}^* - C_{i,n-i}$$

Le paramètre σ_j^2 peut être estimé, pour $j = 1, \cdots, n-1$ par :

$$\sigma_{*j}^2 = \frac{1}{n-j-1} \sum_{i=0}^{n-j-1} C_{i,j} \left(\frac{C_{i,j+1}}{C_{i,j}} - \lambda_j \right)^2$$

Pour estimer σ_j^2 pour $j = n$ nous avons deux possibilités :
— si $\lambda_n^* = 1$ alors $\sigma_n^{*2} = 0$
— Sinon, $\sigma_n^{*2} = min(\frac{\sigma_{n-1}^*}{\sigma_{n-2}^*}; min(\sigma_{n-1}^*; \sigma_{n-2}^*))$

Erreurs de prédiction Il convient maintenant d'estimer l'erreur de prédiction.
On mesure classiquement l'incertitude d'estimation de $C_{i,n}^*$ par l'écart quadratique moyen conditionnel (MSEP, *Mean Square Error Prediction*) :

$$MSEP(C_{i,n}^*) = \mathbb{E}[(C_{i,n} - C_{i,n}^*)^2|T]$$

Dans cette espérance, seule $C_{i,n}$ est aléatoire. On en déduit ainsi :

$$MSEP(C^*_{i,n}) = V[C_{i,n}|T] + [\mathbb{E}[C_{i,n}|T] - C^*_{i,n}]^2$$

Cet écart mesure aussi l'incertitude présente dans les provisions :

$$MSEP(R^*_i = \mathbb{E}[(R_i - ^*_i)^2]$$

Validation du modèle Comme pour la méthode Chain-Ladder, des contrôles graphiques doivent être mis en place pour les hypothèses. Il convient donc de vérifier que les $(n-j)$ couples $(C_{i,j}, C_{i,j+1})_{i=0,\cdots,n-j-1}$ sont suffisamment bien alignés sur une droite passant par l'origine. De plus, pour vérifier l'hypothèse 4, il convient de vérifier à j fixé que le nuage de points des paiements cumulés et des résidus $\frac{C_{i,j+1} - \lambda^*_j C_{i,j}}{\sqrt{C_{i,j}}}$ ne fait apparaître aucune tendance.

1.2 Méthodes individuelles

Un développement récent de la littérature actuarielle s'interroge sur la question du provisionnement au niveau individuel. La plupart de ces travaux sont réalisés dans un cadre stochastique.
Arjas (1989) [2] et Norberg (1993) [36] (1999) [37] formulent un cadre théorique au dévelopemment des sinistres individuels. Ils utilisent notamment des idées issues de la théorie des martingales et présentent un cadre probabiliste plutôt que statistique pour le provisionnement individuel.

Norberg propose l'utilisation des processus de Poisson marqués. Il s'agit de modèles représentant la survenance de sinistres à l'aide de processus de Poisson en temps continu. Le nombre de sinistres et la gravité de ces sinistres sont modélisés de façon séparée. À la survenance d'un sinistre, on associe à celui-ci un vecteur de variables aléatoires modélisant les différents éléments du cycle de vie du sinistre. Cette idée a été par la suite reprise et développée par Haastrup & Arjas (1996) [22] et Larsen (2007) [28].

Ces modèles sont flexibles et permettent de prendre en compte des caractéristiques exhaustives du sinistre :

- date de survenance ;
- délai de déclaration ;
- date et montant des paiements réalisés ;
- date de clôture.

Ces modèles peuvent aussi prendre en compte des caractéristiques détaillées de l'assuré et du contrat d'assurance.
Il apparait donc que le provisionnement individuel peut prendre de nombreuses formes en fonction des paramètres retenus pour la modélisation, ainsi que le type de modélisation proprement dite.

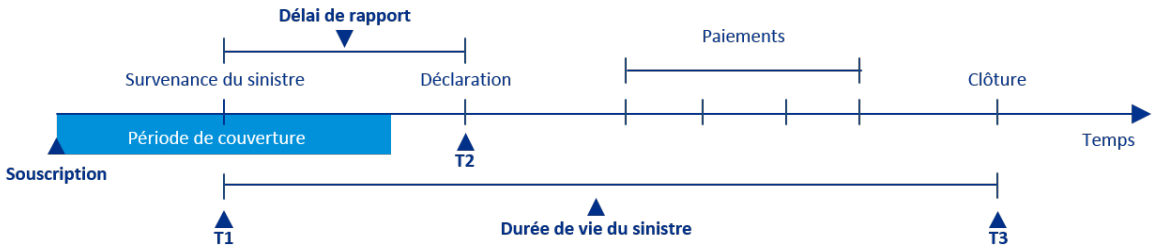


FIGURE 1.2 – Étapes de la vie d'un sinistre

Nous retrouvons ainsi des méthodes d'estimations très variées, preuve en est le nombre de recherches actuarielles récentes qui s'intéressent au sujet du provisionnement individuel.

Ainsi, Antonio & Plat (2014) [1] modélisent le développement en temps continu des sinistres individuels. Drieskens et al. (2012) [11], Rosenlund (2012) [42] et Pigeon et al. (2013) [39] travaillent en temps discret et agrègent les paiements par période de temps (par exemple par année de développement) mais conserve la ligne de temps spécifique du sinistre, comme notamment les dates de survenance et de déclaration.
Zhao et al. (2009) [52] et Zhao & Zhou (2010) [53] travaillent eux aussi en temps continu et proposent un modèle semi-paramétrique pour le développement individuel.

Par ailleurs, le développement récent des méthodes d'apprentissage statistique, et le gain croissant de puissance de calcul ont permis de développer des approches innovantes.
Wüthrich (2016) [50] propose de modéliser des provisions pour sinistres individuels à l'aide d'arbres de régression. Wüthrich (2017) [51], quant à lui, part du modèle de Mack et ajoute de l'information des sinistres individuels aux coefficients de développement grâce à des réseaux de neurones. Cela lui permet ainsi d'analyser les provisions au niveau individuel.

Lopez et al. (2016) [29] proposent quant à eux une méthode d'estimation des poids de Kaplan-Meier par arbres CART et l'appliquent à des données d'assurance, notamment pour du provisionnement individuel.

Nous allons maintenant comparer les avantages et inconvénients des méthodes agrégées et individuelles.

1.3 Comparaison des méthodes agrégées et individuelles

Il est tout d'abord intéressant de remarquer que les modèles stochastiques pour le provisionnement sont apparus à la même période pour les méthodes individuelles (Norberg, 1993) [36] que pour les méthodes agrégées (Mack, 1993).
England & Verral (2002) [13] et Taylor et al.(2008) [45] s'interrogent sur l'utilisation de données agrégées. En effet, en utilisant des données agrégées, de nombreuses informations sur les sinitres restent non utilisées. Par exemple, les méthodes agrégées ne prennent pas en compte l'information disponible sur le contrat d'assurance, son souscripteur, voire même l'historique du contrat.

Pour citer England & Verral (2007) [12] : *"[...] il faut garder à l'esprit que les techniques tradionnelles ont été développées avant l'arrivée des ordinateurs, en utilisant des méthodes qui pouvaient être évaluées avec un crayon et du papier. Avec l'accroissement permanent de la puissance de calcul des ordinateurs, il faut s'interroger s'il ne vaudrait mieux pas utiliser des données individuelles plutôt qu'agrégées".*

1.3.1 Avantages des méthodes agrégées

Le premier avantage des méthodes agrégées est que celles-ci sont très robustes, puisqu'elles reposent sur la loi des grands nombres.

Un deuxième avantage des méthodes agrégées est qu'elles sont bien établies et documentées. Utilisées par les actuaires depuis de nombreuses années, ces méthodes ont fait l'objet de nombreuses études. De plus, ces méthodes sont très pratiques pour la comptabilité et l'audit, et sont facilement compréhensibles.

Enfin, ces méthodes ont le principal avantage d'être simples à mettre en œuvre, de nécessiter peu de puissance de calcul et un niveau de granularité des données assez faible.

1.3.2 Inconvénients des méthodes agrégées et intérêt des méthodes individuelles

Le premier inconvénient est que la prévision n'est pas possible au niveau individuel. Ainsi nous avons une différence de traitement entre la tarification, qui permet d'obtenir le prix d'un contrat au niveau individuel et le provisionnement. De plus, une non-stationnarité des provisions, ainsi que de possibles déformations individuelles importantes sont difficiles à détecter. Il est, de plus, impossible de prendre en compte la réassurance non proportionnelle, qui s'applique sinistre par sinistre.

Toutefois, le principal inconvénient est la perte d'information. En agrégeant les données, nous perdons de précieuses informations détaillées au niveau individuel, qui pourraient améliorer la prédiction. En outre, à cause du faible nombre d'observations dans un triangle agrégé, ces méthodes sont très sensibles aux valeurs extrêmes et à la sur-paramétrisation.
Un autre problème résultant de l'utilisation de méthodes agrégées est la difficulté de mener des études sur les sinistres ouverts mais non clos et sur les sinistres non déclarés.
Enfin, ces méthodes peuvent être biaisées sous certaines conditions comme l'ont démontré Halliwell (2007) [23] et Schiegl (2015) [44]. Notamment, les méthodes type Chain-Ladder ont tendance à surestimer les provisions et sont biaisées pour des structures temporelles non multiplicatives.

Ces différents inconvénients ont motivé le développement de la recherche actuarielle sur le provisionnement individuel. Toutefois, les méthodes individuelles actuelles présentent elles aussi des inconvénients.

1.3.3 Défauts des modèles individuels actuels

Les méthodes individuelles actuelles présentent les inconvénients suivants :

— Ces méthodes sont très techniques, ce qui rend leurs utilisations et explications plus complexes que celles des méthodes agrégées ;
— Ces méthodes sont généralement coûteuses en temps de calcul, surtout pour les modèles stochastiques. De plus, leur temps d'implémentation est supérieur à celui des méthodes agrégées ;
— Le nombre de paramètres à estimer peut parfois être important ;
— Ces méthodes n'ont pas de caractère générique et sont spécifiques à chaque garantie.
— Demandent des données parfois absentes ou difficiles à extraires des bases de gestion.
— Peuvent demander des outils spécifiques en fonction du volume de données.

Malgré ces inconvénients, les méthodes individuelles produisent en général de meilleurs résultats que les méthodes agrégées. Huang (2015) [27] a démontré que les méthodes de provisionnement individuel ont une meilleure précision, au sens de l'erreur quadratique moyenne, ainsi qu'un biais et une variance plus faibles que les méthodes agrégées.

Il convient de préciser que le but de ce mémoire n'est pas de proposer une méthode de provisionnement individuel qui permettrait de se passer de façon définitive des méthodes agrégées. Nous adoptons ici le point de vue de Katrien Antonio, développé lors de la conférence "R in Insurance 2017", sur le sujet. Il s'agit non pas de remplacer les méthodes agrégées, qui ont les avantages conséquents d'être très robustes et simples à mettre en œuvre, mais plutôt d'apporter de l'information supplémentaire grâce aux modèles individuels.

Cette information pourrait être utilisée de nombreuses façons par l'assureur, que ce soit pour détecter d'éventuels problèmes dans la gestion des sinistres ou bien pour affiner la prévision de ses provisions techniques sur certains périmètres.

Ainsi, le provisionnement individuel pourrait apporter des indications sur d'éventuelles typologies de sinistres "à risque", ce qui permettrait notamment de suivre de façon plus attentive l'évolution de ces sinistres, qui peuvent représenter un réel enjeu pour l'assureur.

Deuxième partie

Modèle de provisionnement individuel

Dans cette partie, nous allons étudier le modèle de provisionnement individuel que nous proposons.

Notre étude aura la structure suivante :

1. Formalisation du problème de provisionnement et la démarche (étapes) de modélisation.
2. Application de la démarche de modélisation sur les données réelles. En particulier, le découpage de la base en partitions spécifiques (triangle échantillon, triangle inclus...) sera décrit dans cette partie.
3. Description et optimisation des méthodes utilisées : GLM pénalisés, forêts aléatoires et réseaux de neurones.
4. Comparaison des modèles/résultats (entre les modèles et réel/estimé) puis synthèse.

Chapitre 2

Formalisation du problème et modèle de provisionnement individuel

Dans ce chapitre, nous allons formaliser la façon dont nous allons estimer les provisions au niveau individuel, avant de décrire l'algorithme que nous proposons pour le provisionnement individuel puis de représenter graphiquement les résultats obtenus avec cet algorithme.

2.1 Formalisation du problème de provisionnement

Commençons par fixer le cadre suivant :
Soit une base de données concernant N sinistres survenus entre les années $j = 1$ et n.
Nous connaissons le développement des paiements cumulés de ces sinistres sur les années $i = 1, \cdots, n$.
Notons T le triangle de paiements connus, et $T_{\alpha,j}$ le développement des paiements en j du sinistre α.
Nous pourrions, sans perte de généralité, tout aussi bien considérer les charges à la place des paiements.

De plus, nous pourrions tout aussi bien considérer des sinistres graves ou attritionnels, et des données comportant ou non des frais, des recours ou de la réassurance.

Nous avons de plus des données **individuelles** sur ces sinistres, pouvant être utilisées comme variables explicatives. Fixons le nombre de ces variables explicatives à p.
Nous avons la représentation suivante :

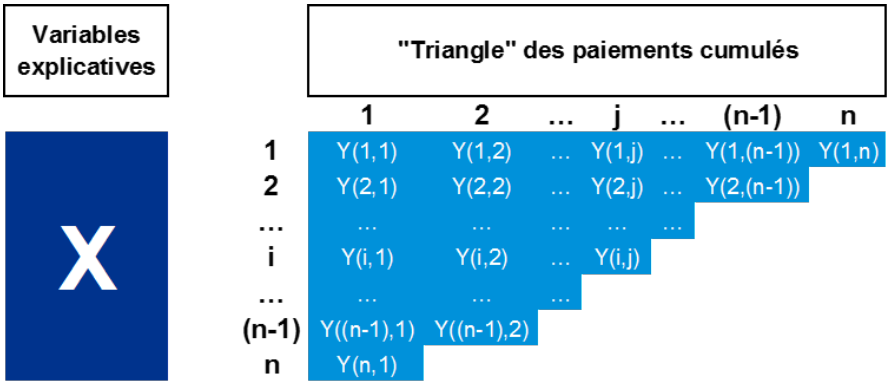


FIGURE 2.1 – Représentation des données

La matrice X, de taille $N \times p$ (N vecteurs de taille p), contient les variables explicatives individuelles de nos sinistres.
Chaque élément $Y_{i,j}$ du triangle de paiements est en fait un vecteur de taille N_i, où N_i est le nombre de sinistres survenus l'année i.
Le vecteur $Y_{i,j}$ contient donc le développement de l'année j des sinistres survenus l'année i.

Nous cherchons à estimer le coût à l'année n, donc les $Y_{i,n}$ de ces sinistres. Autrement dit, nous cherchons à compléter notre triangle de développement.
Nous pouvons voir cet objectif comme un problème de régression :
nous cherchons une fonction $f : X_\alpha \times T_\alpha \to \mathbb{R}$ qui nous permet d'estimer les paiements cumulés pour le sinistre α au développement n.
En notant X_α le vecteur contenant les variables explicatives relatives au sinistre α et $Y_\alpha(n)$ les paiements cumulés du sinistre α au développement d'année n, nous aurons ainsi :

$$f(X_\alpha, T_\alpha) \approx Y_\alpha(n)$$

Cette régression sera réalisée grâce à des algorithmes d'apprentissage statistique, que nous décrirons dans des chapitres ultérieurs.

Nous proposons de procéder de façon itérative, par année de développement. Cette démarche a notamment été proposée par Harej et al [24]. En d'autres termes, nous allons compléter les colonnes inconnues du triangle année de développement par année de développement.
Nous cherchons donc à obtenir le résultat suivant :

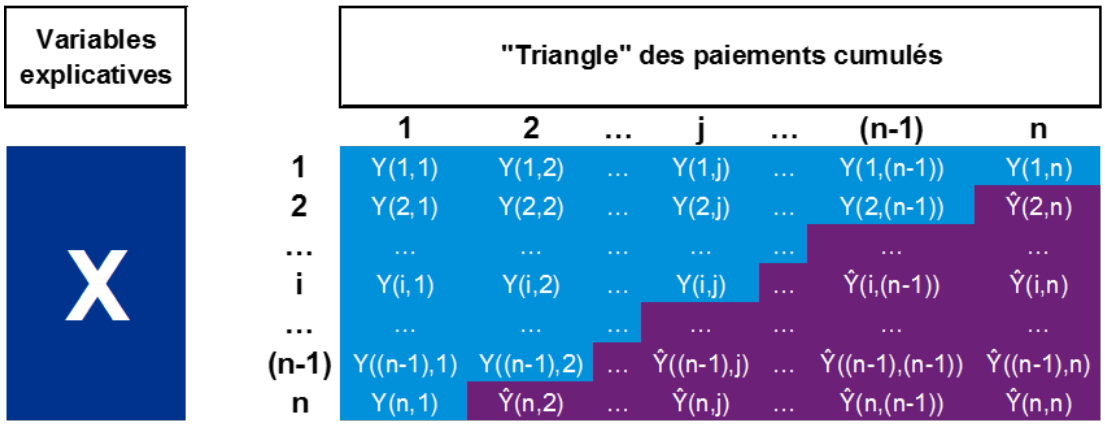


FIGURE 2.2 – Objectif du modèle : complétion du triangle de paiements cumulés

où $\hat{Y}(i,j)$ représente l'**estimation** du vecteur des paiements cumulés des sinistres survenus l'année i pour le développement j.
Nous connaissons déjà cette information pour (i,j), avec $i \leq (n-j+1)$, ce que représente le triangle supérieur bleu clair.
Nous cherchons donc à estimer les $\hat{Y}(i,j)$ pour $j = 2, \cdots, n$ et $i > (n-j+1)$.
Nous allons estimer en premier temps le vecteur $\hat{Y}(n,2)$, puis les vecteurs $(\hat{Y}((n-1),3), \hat{Y}(n,3))$, et ainsi de suite jusqu'aux vecteurs $(\hat{Y}(2,n), \cdots, \hat{Y}(n,n))$.

Nous présentons par la suite notre algorithme de complétion du triangle de paiements cumulés.

2.2 Algorithme de projection pour le provisionnement individuel

Considérons un algorithme d'apprentissage statistique. Pour chaque année de développement $j = 2, \cdots, n$:

1. Sélectionner les sinistres d'années de survenance $i \leq (n-j+1)$, dont les paiements cumulés sont connus.
2. Utiliser l'algorithme d'apprentissage pour estimer les vecteurs $Y(i,j)$, pour $i = 1, \cdots, (n-j+1)$.
   Cette estimation est réalisée à partir des vecteurs de variables explicatives X_α pour les sinistres α de survenance i, et des paiements cumulés antérieurs $Y(i,k)$ pour $k < j$ et $i = 1, \cdots, (n-k+1)$.
3. Utiliser l'algorithme ainsi entraîné sur l'échantillon de test des vecteurs $Y(i,k)$ pour $k < j$ et $i = (n-j+2), \cdots, n$, afin d'estimer les vecteurs $Y(i,j)$ pour $i = (n-j+2), \cdots, n$.
4. Compléter le triangle avec les vecteurs $\hat{Y}(i,j)$ ainsi obtenus, pour $i = (n-j+2), \cdots, n$.

Dans un but d'aide à la compréhension, nous allons ci-après représenter graphiquement notre algorithme étape par étape.

2.3 Représentation graphique

Nous commençons par le triangle de paiements cumulés individuels T.

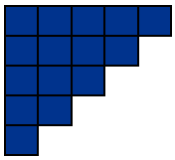


FIGURE 2.3 – Triangle T des paiements cumulés individuels

Ici, chaque carré représente l'ensemble des paiements cumulés de sinistres pour une même année de survenance et année de développement.
Commençons tout d'abord par le cas $j = 2$.

Première étape Nous ne gardons que les sinistres d'année de survenance $i \leq (n - j + 1)$ pour l'apprentissage, c'est à dire tous les sinistres sauf ceux de la dernière année de survenance.

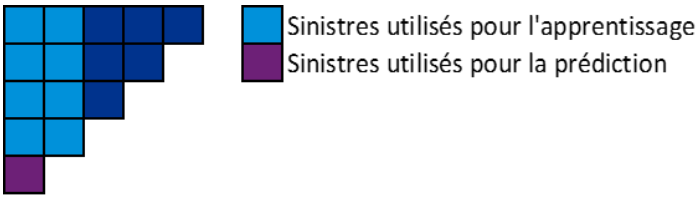


FIGURE 2.4 – Séparation par années de survenance

Deuxième étape Nous apprenons ensuite à notre algorithme d'apprentissage à estimer, pour $i = 1, \cdots, (n-1)$, $Y(i, 2)$ à partir des vecteurs de X correspondants à nos sinistres sélectionnés et des vecteurs $Y(i, 1)$.

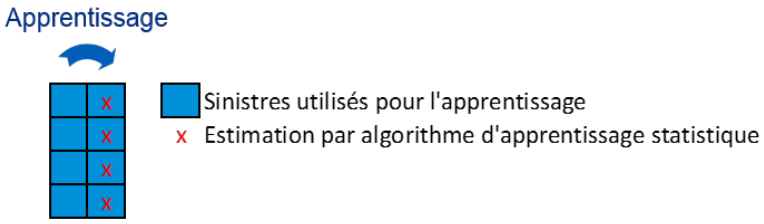


FIGURE 2.5 – Apprentissage

Troisième étape Nous utilisons l'algorithme ainsi entraîné sur les données des sinistres de survenance $i = n$, pour estimer $Y(n, 2)$.

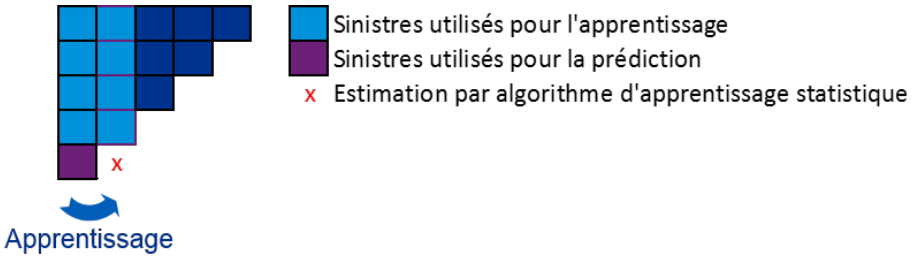


FIGURE 2.6 – Prédiction

Quatrième étape Nous complétons ainsi l'estimation $\hat{Y}(n, 2)$.

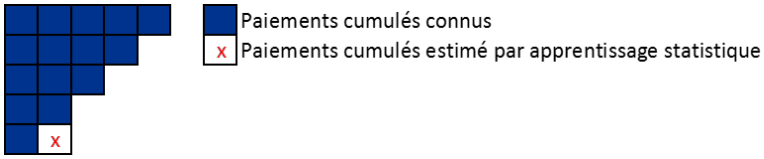


FIGURE 2.7 – Complétion du triangle avec les valeurs prédites

Il est important de noter que pour $j > 1$, nous réutilisons en phase de test des données estimées par les étapes précédentes.

Représentons maintenant l'algorithme pour j quelconque.

Première étape Nous ne gardons que les sinistres d'année de survenance $i \leq (n - j + 1)$ pour l'apprentissage.

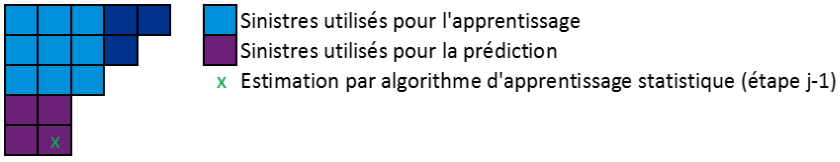


FIGURE 2.8 – Séparation par années de survenance

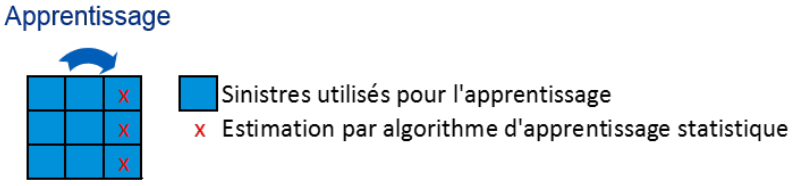


FIGURE 2.9 – Apprentissage

Deuxième étape

Troisième étape Nous utilisons l'algorithme ainsi entraîné sur la base de test, **qui contient des données issues d'estimations préalables**.

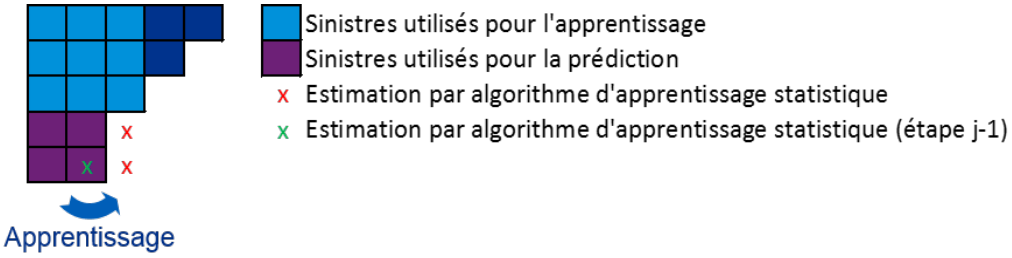


FIGURE 2.10 – Prédiction

Quatrième étape Nous complétons ainsi l'estimation.

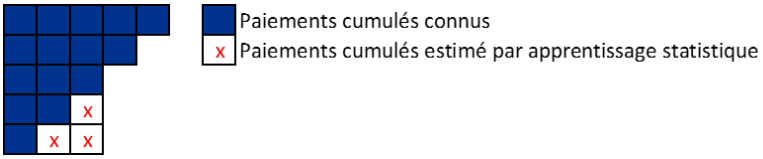


FIGURE 2.11 – Complétion du triangle avec les valeurs prédites

Ainsi, par itérations successives, nous obtiendrons le triangle de paiements cumulés complété.

Nous allons maintenant présenter succinctement la base de données et la modélisation que nous utiliserons pour notre étude.

Chapitre 3

Présentation de la base de données, traitements effectués et application du modèle

Ce chapitre présente la base de données étudiée, ainsi que la modélisation totale utilisée.

3.1 Présentation de la base de données et traitements effectués

Avant d'appliquer nos algorithmes, nous allons dans un premier temps étudier la base de données utilisée.

Nous sommes en présence de données fournies par un assureur français partenaire de cette étude. **Pour des questions de confidentialité** , certaines valeurs ou variables ont été modifiées, sans que cela n'impacte la pertinence des résultats présentés.

La base comporte **275000** sinistres, répartis sur la période **2004-2016**, pour une garantie assurance responsabilité civile **corporelle**.

Cette base comporte notamment un triangle de dépenses réelles, et un triangle de charges estimées par les gestionnaires de l'assureur. Ces triangles constituent les variables d'intérêt de notre étude, notre but est de les compléter par l'algorithme présenté au chapitre précédent.
Nous ne disposons pas de données d'exposition telles que les primes, nous utiliserons donc comme méthode agrégée de provisionnement de référence une méthode Chain-Ladder.

Nous faisons une première sélection des variables explicatives, en écartant au préalable les variables sans description et comportant trop de modalités. Nous obtenons **29** variables explicatives.

Ces variables sont uniquement catégorielles.
Les variables qui nous paraissent comme les plus importantes pour l'étude sont :

- La variable `Grave`, qui correspond au caractère grave au non du sinistre, tel qu'estimé par l'assureur.
- Les variables `IPPX_Y_surv`, qui correspond à la présence ou non de victimes avec un taux d'IPP compris entre X et Y, vision année de survenance du sinistre.
- La variable `Nbvic_surv`, qui représente le nombre de sinistrés.
- Les variables `Para,Tetra,Trauma,Poly`, qui représentent respectivement la présence ou non de victimes paralysées, tétraplégiques, traumatisées ou polytraumatisées, à cause du sinistre.
- La variable `Clos_surv`, qui représente le caractère clos ou non du sinistre à la fin de l'année de survenance.

Ces variables nous apparaissent comme étant les principales porteuses d'information des dépenses et des charges des sinistres.

Nous pouvons obtenir une matrice de "corrélation" en représentant la mesure d'association *V de Cramer* entre chaque variable :

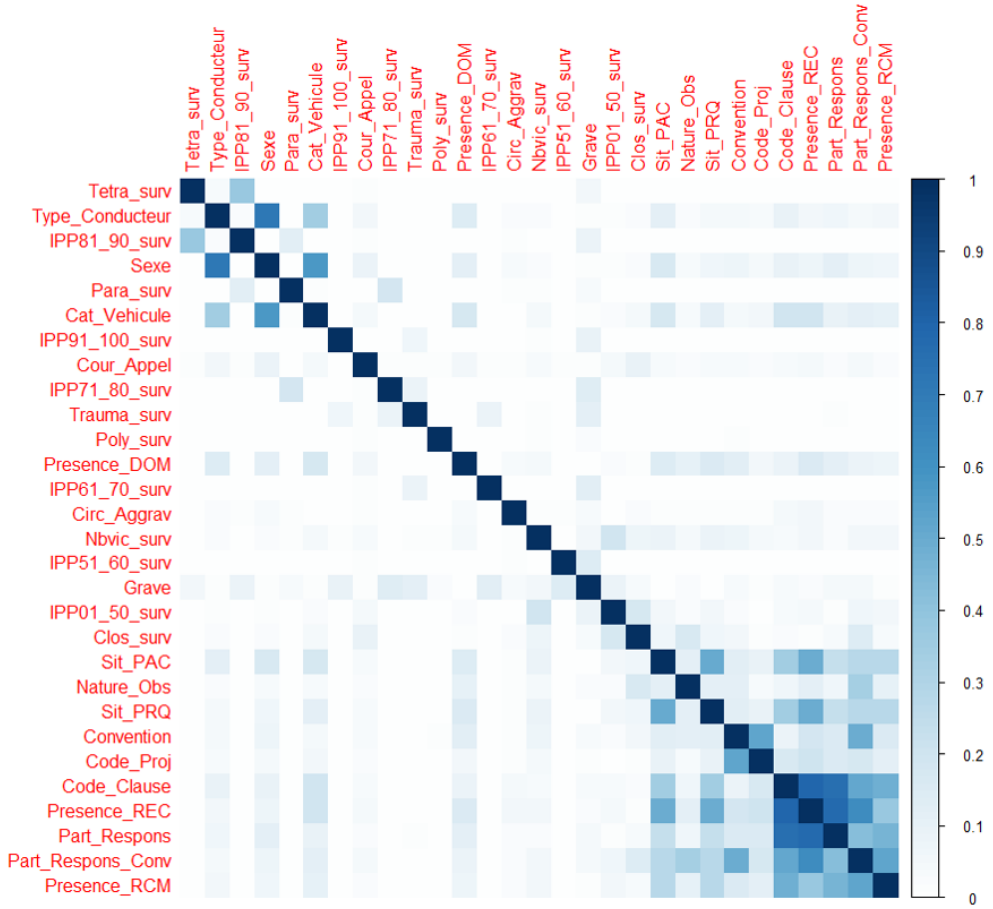


FIGURE 3.1 – Matrice de corrélation (V de Cramer) des variables explicatives.

Le V de Cramer est une mesure d'association, basée sur la statistique du χ^2 de Pearson.
Notons deux variables catégorielles A et B, ayant un nombre de modalités respectives r et k. Soit $n_{i,j}$ le nombre de fois où nous observons les valeurs (A_i, B_j).

La statistique du χ^2 s'écrit :

$$\chi^2 = \sum_{i,j} \frac{\left(n_{i,j} - \frac{n_{i.}n_{.j}}{n}\right)^2}{\frac{n_{i.}n_{.j}}{n}}$$

Le V de Cramer s'écrira alors :

$$V = \sqrt{\frac{\chi^2/n}{min(k-1, r-1)}}$$

Nous représentons par une matrice de chaleur les variables les plus corrélées entre elles selon la mesure d'association V de Cramer : plus les variables sont corrélées entre elles, plus le V de Cramer sera proche de 1, ce que nous représentons ici par une couleur sombre. Des variables totalement indépendantes entre elles auront un V de Cramer nul, ce que nous représentons par une absence de couleur.
Nous remarquons que nos variables ont généralement l'air peu corrélées entre elles. Les variables les plus corrélées entre elles sont :

- `Sexe` (le sexe de l'assuré) et `Type_Conducteur` (type de conducteur au moment de l'événement : assuré, conjoint de l'assuré...).
- `Code_Clause` (application ou non d'un malus), `Part_Respons` (code de la part de responsabilité de l'assuré) et `Presence_REC` (présence ou non de recours).

Nous avons choisi préalablement d'écarter une variable : la variable `Lieu_surv`, qui correspond au type de lieu de survenance du sinistre (campagne ou ville). En effet, cette variable est renseignée de façon différente sur les périodes 2004-2013 et 2014-2016 : bien plus de sinistres sont enregistrés en milieu urbain sur cette deuxième période. De plus, les sinistres en milieu urbain sur la première période présentent un coût moyen supérieur à l'ensemble des sinistres. Cette variable pourrait entraîner une déformation évidente dans notre projection, nous choisissons donc de l'écarter.

Nos variables cibles seront les incréments de dépenses et les variations de charges. Nous cumulerons ensuite les résultats obtenus pour comparer plus facilement à une méthode type Chain-Ladder.
Nous choisissons ici les incréments de dépenses, et non pas les dépenses cumulées, dans le but d'éviter des phénomènes possibles de décroissances dans les prédictions. En effet, prenons ici en exemple le cas de deux sinistres ayant des caractéristiques similaires, le premier ayant une dépense initiale de 100 et une dépense cumulée totale lors de sa deuxième année de développement à 110, donc un incrément de 10.
Notre second sinistre à quant à lui une dépense initiale de 115, mais toutes ses autres caractéristiques sont identiques à celles du premier sinistre.

	Variables explicatives	Dépense initiale	Dépenses cumulées au 2ème développement	Variation
Sinistre 1	X	100	110	10
Sinistre 2	X	115	110	-5

FIGURE 3.2 – Représentation d'un phénomène de décroissante dans la prédiction de dépenses cumulées. Les prédictions sont ici notées en rouge.

Si nous cherchons à prédire directement la dépense cumulée de ce sinistre pour la deuxième année de développement, nous pourrions obtenir une prédiction identique à la dépense cumulée

du premier sinistre, car ces eux sinistres sont très proches. Notre second sinistre passerait ainsi d'une dépense cumulée de 115 à 110, donc une dépense décroissante, ce qui n'est généralement pas souhaitable.
Toutefois, si nous passons par les incréments de dépenses, les prédictions seront toujours supérieurs ou égales à 0, nous n'aurons ainsi pas de phénomènes de décroissances.
Nous choisissons aussi de prédire les variations de charges, pour des soucis de cohérences entre les deux modèles.
Les dépenses et charges initiales seront respectivement nommées dépenses ou charge au développement d'année 1.

Nos incréments de dépenses réelles pour le développement N seront notées `dev_N`, tandis que les variations de charges pour le développement N seront notées `ctb_N`.
Dans notre approche, nous utilisons de plus respectivement les incréments de dépenses et les variations de charges des développements précédents comme variables explicatives pour l'estimation des incréments de dépenses et des variations de charges des développements futurs.
Certaines méthodes d'apprentissage statistique ont besoin, ou offrent de meilleurs résultats, avec des variables centrées et réduites. Nous choisissons de créer des variables `dev.n_N`, qui seront des versions centrées et réduites des variables `dev_N`.
Nous procédons de façon similaire pour créer des variables `ctb.n_N`, versions centrées et réduites des variables `ctb_N`.

Ces variables seront créées uniquement lorsque tous les sinistres de l'année N auront été estimés : par exemple, la variable `dev.n_1` peut être créée avant application de notre algorithme car toutes les dépenses à la survenance sont connues. Toutefois pour la variable `dev.n_2`, les sinistres développés à 1 an pour l'année de survenance 2016 ne sont pas connues, cette variable sera donc créée après la première itération de notre algorithme.

Nos variables autres que les dépenses et les charges sont uniquement catégorielles, nous n'aurons donc pas besoin de les normaliser. Toutefois, certains algorithmes d'apprentissage statistique ont besoin de valeurs numériques : nous créerons donc une version modifiée de notre base à l'aide de variables *dummy* :
Pour une variable catégorielle donnée à n modalités, nous créons $n-1$ variables codées binairement. La i-ème variable vaudra 1 dans le cas d'un individu présentant la modalité i et 0 sinon.

Nous allons maintenant présenter la modélisation complète qui sera utilisée, notamment pour le choix des variables pour les algorithmes d'apprentissage statistique, ainsi que pour l'optimisation des paramètres.

3.2 Modélisation

La base de données étant relativement volumineuse et le champ des paramètres que nous voulons tester pouvant s'avérer imposant, notamment pour les réseaux de neurones, nous décidons

de procéder en plusieurs étapes.

— La première étape regroupe le traitement préalable des variables explicité ci-dessus, ainsi que le traitement des valeurs manquantes ou aberrantes et la création d'un triangle échantillon.
— La seconde étape regroupe l'application des modèles à un échantillon du triangle, afin de pouvoir trouver les paramètres optimaux plus rapidement que sur l'ensemble des données, mais aussi de pouvoir tester bien plus de combinaisons de paramètres.
— La troisième étape consiste à appliquer le modèle de provisionnement aux données initiales avec les paramètres retenus à la seconde étape.

Par abus de langage, nous appellerons "triangle" l'ensemble de données individuelles regroupant les variables choisies, et en fonction soit les dépenses soit les charges associées.
La modélisation complète est représentée dans le graphique suivant :

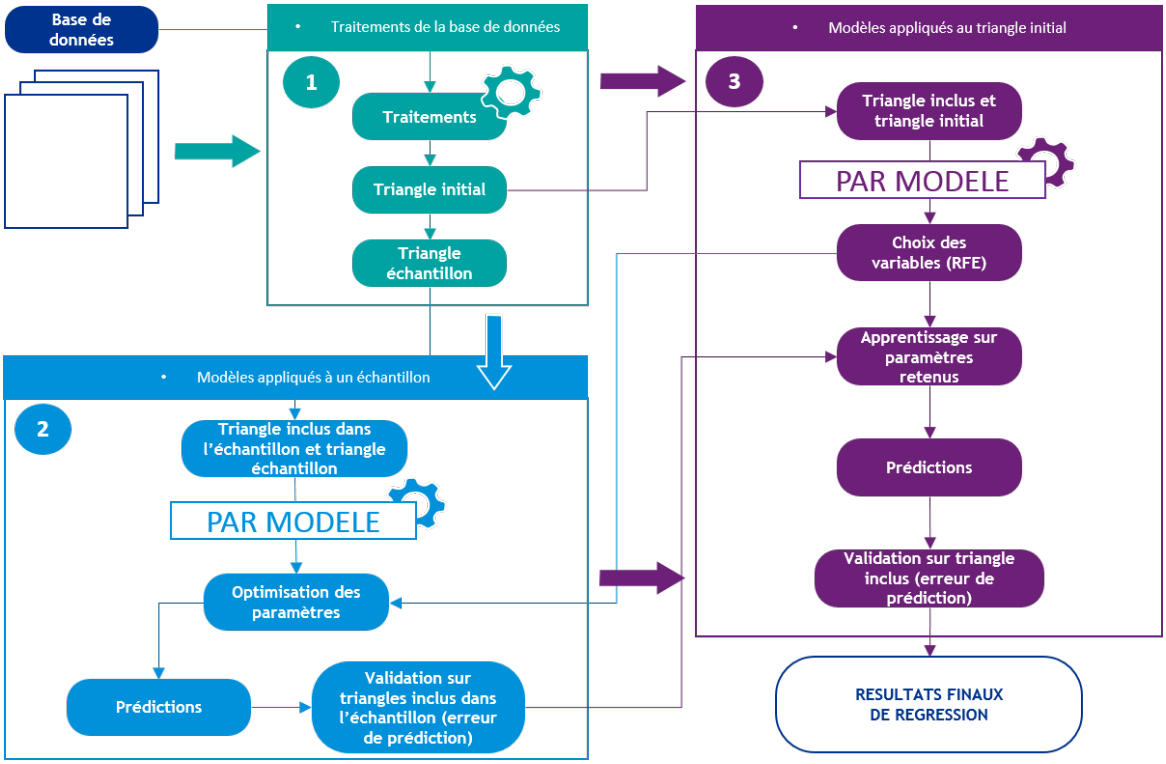


FIGURE 3.3 – Représentation de la modélisation

3.2.1 Séparation en triangles inclus et triangles échantillons

Nous allons séparer chacun de nos triangles de données (de dépenses et de charges) de la manière suivante :

— Le triangle sans aucune modification sera nommé "triangle initial",
— Le triangle obtenu par une technique d'échantillonage sera nommé "triangle échantillon",
— Le triangle regroupant les années de survenances 2004 à 2010 sera nommé "triangle inclus".

Triangle échantillon Le triangle échantillon correspond à un triangle obtenu par technique d'échantillonage stratifié sur la variable `grave`, représentant 20% des données initiales.
L'échantillonage stratifité permet de conserver la répartition de la variable `grave` du triangle initial dans le triangle échantillon, ce qui permet d'avoir un échantillon suffisamment représentatif par rapport à cette variable.
Le triangle échantillon a comme objectif de pouvoir tester plus rapidement plus de combinaisons de paramètres possibles, car il est moins volumineux que le triangle initial.

Triangle inclus Le triangle inclus correspond au triangle initial mais restreint aux années de survenances 2004 à 2010. Cela permet de pouvoir "compléter" ce triangle et le comparer à des données réelles, car tous les développements du triangle inclus complété sont connus. Nous aurons notamment un triangle inclus dans le triangle échantillon ; nous utiliserons le triangle échantillon associé au triangle inclus pour déterminer les paramètres optimaux sur ce dernier.

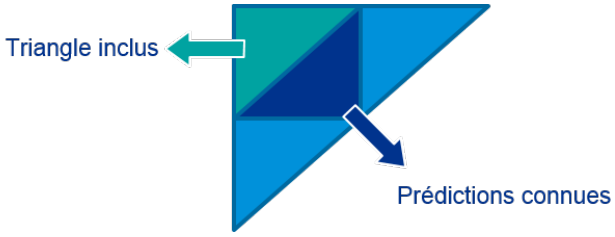


FIGURE 3.4 – Représentation schématique du triangle inclus

Nous pouvons résumer les différents triangles comme suit :

Triangle	Nombre de lignes	Périmètre	Construction	Utilité
Initial	255 900	2004 - 2016	Ensemble des données initiales	-
Echantillon (20%)	51 180	2004 - 2016	Echantillonage stratifié	Test de paramètres
Inclus	145 671	2004 - 2010	Périmètre 2004 - 2010	Validation des modèles Comparaison à des données réelles
Inclus dans l'échantillon (20%)	29 286	2004 - 2010	Périmètre 2004 - 2010 de l'échantillon stratifié	Validation des modèles Comparaison à des données réelles

FIGURE 3.5 – Synthèse des différents triangles

3.2.2 Application des modèles au triangle échantillon

Pour chaque modèle, les étapes suivantes seront réalisées sur chaque triangle échantillon (de dépenses et de charges). Elles seront d'abord réalisées sur le triangle inclus, à des fins de validation par comparaison à des données réelles, puis sur le triangle initial.
Nous cherchons à prédire les incréments de dépenses et les variations de charges. Nous ne projeterons que les sinistres connus et non-clos lors de leur premier développement.

Utilisation de la validation croisée : la validation croisée est une technique d'échantillonage qui permet de réduire le risque de surapprentissage ainsi que le risque de sur-représentation de valeurs extrêmes dans l'échantillon d'apprentissage ou de test. Nous considérons ici la *k-fold cross*

validation.
Elle consiste à créer k partitions de l'échantillon d'origine. L'algorithme d'apprentissage statistique est ensuité entrainé sur $k-1$ de ces échantillons, et le dernier est conservé comme échantillon test ; l'erreur de prédiction sera calculée sur cet échantillon. Le principe de validation croisée est ensuite répété pour que chaque k échantillon soit utilisé exactement une seule fois.
L'erreur de prédiction finale sera calculée comme la moyenne des erreurs de prédictions obtenues à chaque étape de la validation croisée.

Nous choisissons ici comme erreur de prédiction le **RMSE** (Root-mean-squared error). Pour un estimateur $\hat{\theta}$ de θ, le RMSE est l'erreur d'estimation suivante :

$$RMSE(\hat{\theta}) = \sqrt{\mathbb{E}((\theta - \hat{\theta})^2)}$$

A partir de valeurs prédites $\hat{y}_i$ par l'algorithme, $i = 1, \cdots, n$ nous comparons avec les valeurs attendues y_i :

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

Nous garderons les paramètres offrant le plus petit RMSE à notre modèle.
Dans notre étude, nous utiliserons une validation croisée à 10 sous-échantillons.

Choix des variables par *Recursive Feature Elimination* (RFE) Une première étape consiste à choisir les variables explicatives utilisées pour chaque modèle, à partir du triangle initial.

Le RFE est un algorithme de choix de variables basé sur **l'importance** que donne le modèle aux variables. L'avantage d'une telle approche est qu'elle est liée à la performance obtenue par le modèle, et peut inclure la structure des corrélations entre les prédicteurs pour le calcul des importances.

Pour calculer les importances, nous utiliserons les algorithmes suivants :
— Pour les forêts aléatoires, la réduction d'impureté (variance)
— Pour les réseaux de neurones, l'algorithme de Garson
— Pour les GLM pénalisés, nous conserverons toutes les variables ; c'est en effet l'un des avantages de la pénalisation, nous reviendrons plus en détail dessus dans le chapitre consacré aux GLM.

Nous allons appliquer l'algorithme RFE pour le passage de la valeur du premier développement à la valeur du deuxième développement.
L'algorithme va procéder de la manière suivante :

1. Avec des paramètres "par défaut", en utilisant toutes les variables, calculer un RMSE de prédiction avec une validation croisée à 10 sous-échantillons,
2. Calculer l'importance des variables avec la méthode liée au modèle, et enlever la variable la moins importante,

3. Recommencer jusqu'à ne conserver qu'une seule variable,
4. Comparer tous les RMSE obtenus, et conserver les variables liées au modèle avec le RMSE le plus faible.

Optimisation des paramètres Cette étape consiste à trouver, pour chaque modèle, les paramètres optimaux.
Nous utilisons une technique dite de **"recherche en quadrillage"**, couplée à une **validation croisée** à 10 sous-échantillons. La combinaison de paramètres ayant obtenue le plus petit RMSE sera dite "optimale", et conservée pour la prédiction.

Nous procédons de la façon suivante :

1. Choix des paramètres à optimiser
2. Subdivision en 10 échantillons pour utilisation de la validation croisée
3. Prédiction avec validation croisée pour chaque combinaison de paramètres
4. Comparaison des prédictions (RMSE) obtenues ; on conserve la combinaison avec le RMSE le plus faible

Il est à noter que certains paramètres dépendront du développement, tandis que d'autres seront communs à tous les développements.
Pour ces derniers, nous comparerons les RMSE obtenus lors du dernier développement sur le triangle inclus (donc comparaison à des données réelles), et nous réutiliserons ces même paramètres sur le triangle initial.

Prédictions et validation, puis application au triangle initial Nous réalisons maintenant les prédictions pour les incréments de dépenses et les variations de charges grâce aux paramètres optimaux obtenus, et nous comparons les résultats obtenus aux données réelles dans le cas du triangle inclus. Enfin, nous appliquons le modèle avec les paramètres optimaux sur le triangle échantillon au triangle initial pour obtenir les résultats finaux.

Analyse des résultats La visualisation de résultats est toujours un problème épineux dans le cas de problèmes de régression par algorithme d'apprentissage.
Sur le triangle inclus, nous pouvons comparer directement les résultats obtenus à des données réelles au niveau individuel, et comparer les modèles entre eux :

— par des graphiques prédis contre réels, qui sont souvent utilisés dans les analyses de résultats de régressions,
— par une comparaison de métriques d'erreurs de prédictions individuelles entre modèles,
— par une comparaison de résultats agrégés, que l'on pourra comparer à des résultats issus de méthodes type Chain-Ladder.

Sur le triangle initial, nous ne connaissons pas les prédictions réelles. Nous ferons pour cela un backtesting, en appliquant le modèle au triangle sans la dernière diagonale (correspondant à l'année comptable 2016), et nous comparerons les prédictions des modèles à cette diagonale réelle.

Avant de présenter nos algorithmes d'apprentissage statistique, nous commencerons par présenter les GLM et la méthode de régularisation Elastic Net.

Chapitre 4

GLM et pénalisation Ridge/Lasso

Le Modèle Linéaire Généralisé (noté GLM pour *Generalized Linear Model*) est une méthode classique de modélisation, souvent utilisée par les assureurs non-vie pour modéliser les risques auxquels ils sont exposés.

Développé initialement par Nelder et Wedderburn en 1972 [35], ils sont par la suite détaillés dans Nelder & Mc Cullag (1983)[32].

Dans ce chapitre, nous commencerons tout d'abord par rappeller succinctement le cadre général de la modélisation par GLM, avant d'introduire les pénalisations Ridge et Lasso et leur utilisations pour les GLM, pour finalement appliquer un GLM pénalisé à notre modélisation pour le provisionnement individuel.

4.1 Cadre général de la modélisation par GLM

Le GLM est un modèle prédictif, basé sur la combinaison linéaire de variables explicatives.

Soit un vecteur de variables explicatives $X = (X_1, \cdots, X_p)$, celles-ci pouvant être aussi bien quantitatives que qualitatives. Nous sommes dans un problème de régression et cherchons à prédire la valeur d'une variable continue Y en fonction des valeurs de X.
Le modèle GLM cherchera donc à estimer la valeur : $\mathbb{E}[Y|X = x]$.

Nous allons tout d'abord décrire les composantes classiques d'un modèle GLM.

4.1.1 Composantes d'un modèle GLM

Les modèles considérés comme des modèles GLM comportent trois composantes :

— Une *composante aléatoire*,
— Une *composante déterministe*,
— Une *relation fonctionnelle*, établissant un lien entre la composante aléatoire et la composante déterministe.

Composante aléatoire La *composante aléatoire* est caractérisée par la distribution de probabilités de la variable à expliquer.
Elle identifie la distribution de probabilité de notre variable cible Y, sous hypothèse que la loi de celle-ci soit issue de la famille exponentielle. Si cette hypothèse est vérifiée, nous pouvons mettre la densité de Y sous la forme :

$$f(y, \theta, \phi) = exp\{\frac{y\theta - v(\theta)}{u(\phi)} + w(y, \theta)\}$$

Cette formulation inclut la plupart des lois usuelles, telles que les lois gaussienne, gaussienne inverse, Poisson, gamma, binomiale...
Le paramètre θ est appelé *paramètre naturel*, le paramètre ϕ est appelé *paramètre de dispersion*. Ce dernier sert à ajuster la variance du modèle à celle observée.

Composante déterministe La *composante aléatoire* est caractérisée par la matrice des variables explicatives X et un vecteur appelé *prédicteur linéaire* β.
Il s'écrit de la façon suivante :

$$\eta = X\beta = \beta_0 + \sum_{i=1}^{p} X_i\beta_i$$

Le vecteur $\beta = (\beta_0, \beta_1, \cdots, \beta_p)$ est le vecteur des coefficients inconnus à estimer.

Relation fonctionnelle Cette troisième composante exprime une relation entre la composante aléatoire et le prédicteur linéaire, à travers une fonction de lien g, supposée monotone et différentiable au moins une fois :

$$g(\mathbb{E}[Y|X = x]) = \eta = X\beta$$

Prenons par exemple le cas d'un échantillon gaussien, de famille de lois $N(\mu, \sigma^2)$, la densité s'écrit :

$$f(y_i, \mu_i) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\{-\frac{(y_i - \mu_i)^2}{2\sigma^2}\}$$

Ou bien encore, sous la forme :

$$f(y_i, \mu_i) = exp\{-\frac{1}{2}\frac{\mu_i^2}{\sigma^2}\} exp\{-\frac{1}{2}\frac{y_i^2}{\sigma^2} - \frac{1}{2} ln(2\pi\sigma^2)\} exp\{y_i \frac{\mu_i^2}{\sigma^2}\}$$

En posant :

$$Q(\theta_i) = \frac{\theta_i}{\phi} = \frac{\mu_i}{\sigma^2}$$

$$a(\theta_i) = exp\{-\frac{1}{2}\frac{\mu_i^2}{\sigma^2}\}$$

$$b(y_i) = exp\{-\frac{1}{2}\frac{y_i^2}{\sigma^2} - \frac{1}{2}ln(2\pi\sigma^2)\}exp\{y_i\frac{\mu_i^2}{\sigma^2}\}$$

la famille gaussienne se met alors sous la forme canonique suivante :

$$f(y_i, \theta_i) = a(\theta_i)b(y_i)exp\{y_iQ(\theta_i)\},$$

ce qui en fait donc une famille exponentielle de paramètre de dispersion $\phi = \sigma^2$ et de paramètre naturel $\theta_i = \mathbb{E}(Y_i) = \mu_i$, et de fonction lien canonique la fonction identité.

4.1.2 Ajustement d'un GLM

L'estimation d'un modèle GLM se fait par la détermination des paramètres β_i, calculés à partir de la méthode dite de maximisation de la (log-)vraisemblance. La contribution à cette dernière s'écrit, pour la i-ème observation :

$$l(y_i, \theta_i, \phi) = lnf(y_i; \theta_i, \phi) = \frac{y_i\theta_i - v(\theta_i)}{u(\phi)} + w(y_i, \phi)$$

Expression des moments Pour l'étude de la log-vraisemblance, nous avons besoin des dérivées premières et secondes par rapport au paramètre θ :

$$\frac{\partial l}{\partial \theta_i} = \frac{y_i - v'(\theta_i)}{u(\phi)}$$

$$\frac{\partial^2 l}{\partial \theta_i^2} = -\frac{v"(\theta_i)}{u(\phi)}$$

Pour les distributions de la famille exponentielle, les conditions de régularité permettent de simplifier car :

$$\mathbb{E}[\frac{\partial l}{\partial \theta_i}] = 0 \quad \text{et} \quad -\mathbb{E}[\frac{\partial^2 l}{\partial \theta_i^2}] = \mathbb{E}[\frac{\partial l}{\partial \theta_i}]^2$$

Alors il vient :

$$\mathbb{E}[Y_i] = \mu_i = v'(\theta_i)$$

Et comme :

$$\mathbb{E}[v"(\theta_i)/u(\phi)] = \mathbb{E}[(Y_i - v'(\theta_i))/u\phi]^2 = \mathrm{Var}(Y_i)/u^2(\phi)$$

Nous obtenons finalement :

$$\mathrm{Var}(Y_i) = v"(\theta_i)u(\phi)$$

Equations de vraisemblance Nous cherchons notre vecteur β, qui est un vecteur de p paramètres. Le prédicteur linéaire η est quant à lui composé de n éléments, car nous avons n observations pour chacune de nos variables explicatives. Ainsi, si nous considérons nos n observations comme indépendantes, et par le fait que θ dépende de β, la vraisemblance s'écrit :

$$\mathfrak{L} = \sum_{i=1}^{n} lnf(y_i, \theta_i, \phi) = \sum_{i=1}^{n} l_i(y_i, \theta_i, \phi)$$

Nous cherchons à maximiser cette fonction de vraisemblance $\mathfrak{L}$; nous cherchons les coefficients β_i, $i = 1, \cdots, p$ tels que $\mathfrak{L}(\beta)$ s'approche le plus possible de 1.

Nous calculons les dérivées partielles :

$$\frac{\partial l_i}{\partial \beta_j} = \frac{\partial l_i}{\partial \theta_i} \frac{\partial \theta_i}{\partial \mu_i} \frac{\partial \mu_i}{\partial \eta_i} \frac{\partial \eta_i}{\partial \beta_j}$$

Comme :

$$\begin{aligned}
&\frac{\partial l_i}{\partial \theta_i} = [y_i - v'(\theta_i)]/u(\phi) = (y_i - \mu_i)/u(\phi) \\
&\frac{\partial \mu_i}{\partial \theta_i} = v"(\theta_i) = \mathrm{Var}(Y_i)/u\phi \\
&\frac{\partial \eta_i}{\partial \beta_j} = x_{i,j} \quad \text{car} \quad \eta_i = X_i'\beta \\
&\frac{\partial \mu_i}{\partial \eta_i} \quad \text{dépend de la fonction de lien} \quad \eta_i = g(\mu_i)
\end{aligned}$$

Nous obtenons finalement les équations de vraisemblance suivantes :

$$\sum_{i=1}^{n} \frac{(y_i - \mu_i)x_{i,j}}{\mathrm{Var}(Y_i)} \frac{\partial \mu_i}{\partial \eta_i} = 0 \quad \text{pour} \quad j = 1, \cdots, p$$

La résolution de ces équations permet de trouver les paramètres β_i.
Ce sont des équations non-linéaires en β, leur résolution nécessite des méthodes itératives dans lesquelles interviennent le Hessien (par exemple pour la méthode de Newton-Rapson) ou la matrice d'information de Fisher (pour la méthode des scores de Fisher).

Nous allons maintenant introduire les termes de pénalisations Ridge et Lasso, puis la méthode Elastic Net.

4.2 Pénalisation Ridge/Lasso et Elastic Net

Commençons tout d'abord par le cadre général de la régression linéaire, qui correspond à celui d'un GLM de famille gaussienne.

Nous avons une variable cible Y et un vecteur de variables explicatives $X \in \mathbb{R}^p$., et nous approximons la fonction de régression cherchée par :

$$g(\mathbb{E}[Y|X = x]) = \eta = X\beta$$

Comme démontrée dans la partie précédente, dans le cas d'une famille gausienne, la fonction g est égale à l'identitée ; nous nous ramenons donc à :

$$\mathbb{E}[Y|X = x] = \eta = X\beta$$

Dans le cas où nous ne mettons aucune contrainte sur les coefficients β, ils peuvent croître fortement en fonction des données. C'est en effet un problème classique des GLM : en ajoutant de la complexité et des variables au modèle, voire en ayant des données volatiles, la variance inhérente aux données peut croître de façon conséquente et perturber fortement la qualité des prédictions.

Afin de contrôler l'évolution des coefficients β, ou de limiter l'intervalle des valeurs prises pr ceux-ci, nous pouvons appliquer une pénalité à la norme de β. Si nous choisissons la norme L1, cette pénalité s'appellera Ridge ; si nous choisissons la norme L2, cette pénalité s'appellera LASSO. La pénalisation LASSO pour *Least Absolute Shrinkage and Selection Operator* a été introduite par Tibshirani en 1996 [46], tandis que la pénalisation Ridge, aussi appellée pénalisation de Tikhonov, a été introduite en 1963 [47]. Nous aurons donc un terme de pénalisation $\lambda||\beta||_1$ pour la pénalisation LASSO et un terme de pénalisation $\lambda||\beta||_2^2$ pour la pénalisation Ridge.
Le paramètre λ représentera l'**intensité** de la pénalisation.

Pénalisation Ridge/Lasso et influence sur le choix des coefficients Afin d'illustrer ces deux méthodes et leurs impacts sur les coefficients et variables explicatives, nous reprenons l'exemple donné par Friedman & al dans *The Elements of Statistical Learning* [17].Nous représentons sur le graphique ci-dessous l'influence des pénalisations Ridge et LASSO.

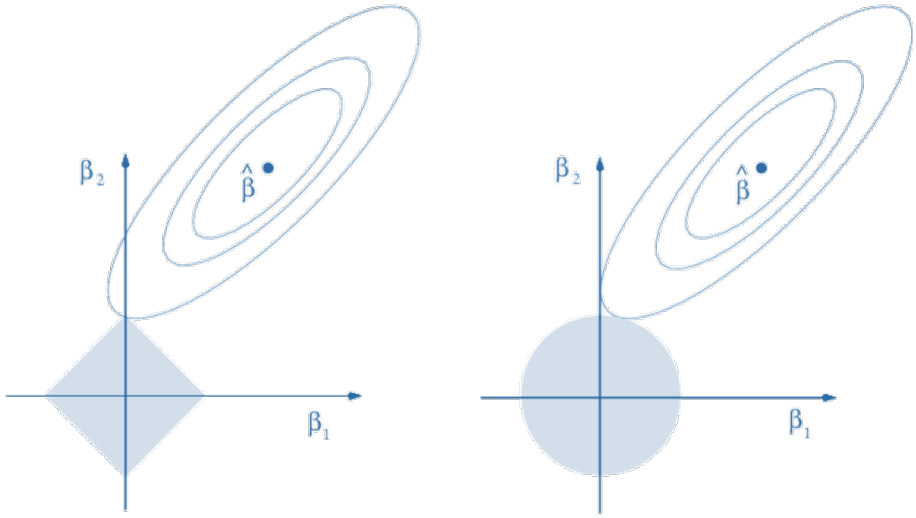


FIGURE 4.1 – Représentation des pénalisations Ridge et Lasso.

A terme d'exemple, nous nous plaçons dans le cas d'une régression linéaire simple. La solution de régression des moindres carrés est marquée $\hat{\beta}$, tandis que les aires en diamant (à gauche) et circulaire (à droite) représentent respectivement les contraintes de régression LASSO et Ridge. Les ellipses concentriques centrées en $\hat{\beta}$ représentent les régions où la somme des carrés des résidus reste constante.
Au fur et à mesure que les ellipses s'éloignent des estimations de coefficients des moindres carrés, la somme des carrés des résidus augmente.

La pénalisation Ridge a une contrainte circulaire, car de norme L2 ; l'intersection avec les ellipses aura tendance à ne pas se produire sur un axe, et donc les estimations de coefficients obtenus par la pénalisation Ridge seront exclusivement différents de zéro.
Au contraire, la pénalisation LASSO aura souvent des intersections sur un axe. De par la forme de la contrainte due à la norme L1, nous pourrons obtenir des coefficients nuls.

La méthode de régression par pénalisation Ridge présente les caractéristiques suivantes :
— Possède une stabilité numérique plus importante que celle de la régression par pénalisation LASSO, et est plus rapide à calculer.
— Garde toutes les variables explicatives dans le modèle, et permet de garder certains effets d'interactions. Notamment, la pénalisation Ridge a tendance à répartir les coefficients parmi les variables fortement corrélées. Dans le cas de n variables parfaitement corrélées, nous aurons un coefficient $\frac{\beta}{n}$ associé à chaque variable, où β représente le coefficient associé si une seule de ces variables était considérée comme variable explicative.
— Réduit les coefficients de façon simultanée avec l'augmentation de la pénalisation λ, sans

toutefois les réduire à 0.

La méthode de régression par pénalisation LASSO présente les caractéristiques suivantes :

— Pour une pénalité λ importante, la méthode LASSO associera une valeur nulle à certains coefficients. Les variables associées seront donc considérées comme retirées du modèle, ce qui permet de faire une sélection de variables.

— Dans le cas de variables explicatives fortement corrélées, la pénalisatin LASSO aura tendance à ne garder qu'une variable explicative, et à donner une valeur égale à 0 pour les coefficients respectifs des autres variables.

— Le fait d'utiliser la norme L1 entraîne l'apparition d'une solution non-linéaire et l'absence d'une formule de résolution fermée, contrairement à la pénalisation Ridge.

Méthode Elastic Net La méthode Elastic Net combine les pénalisations Ridge et LASSO. Elle a été proposée par Zou & Hastie (2005) [54]. Le problème de maximisation de la vraisemblance peut s'écrire :

$$max_{\beta,\beta_0} \sum_{i}^{n} lnf(y_i; \beta, \beta_0) - \lambda \left(\alpha ||\beta||_1 + \frac{1}{2}(1-\alpha)||\beta||_2^2 \right)$$

Le paramètre α est ici appelé "paramètre de mélange", et contrôle la part distribuée entre la pénalité L1 (LASSO, pour $\alpha = 1$) et la pénalité L2 (Ridge, pour $\alpha = 0$).

En combinant de la sorte les pénalisations Ridge et LASSO, la méthode Elastic Net permet à la fois de réaliser une sélection de variables incluse dans le modèle, par la pénalisation LASSO, tout en conservant une capacité d'interactions entre variables corrélées de par la pénalisation Ridge.

Nous allons maintenant analyser la façon dont la méthode Elastic-Net sera appliquée à notre modèle.

4.3 GLM pénalisés - Choix des paramètres optimaux sur le triangle inclus

Pour mettre en place la méthode Elastic Net, nous allons utiliser le package R `glmnet`, développé par Friedman & al. [16]
Nous rappellons la méthode que nous allons suivre pour obtenir nos paramètres optimaux sur le triangle inclus.
Nous procédons sur le triangle inclus de notre triangle échantillon, décrit dans le chapitre III.

Nous utilisons une méthode de recherche en grille. Les deux paramètres principaux à optimiser sans les paramètres α et λ. Notre problème est un problème de rrégression avec une unique variable cible à valeurs réelles continue, l'auteur du package nous recommande d'utiliser le paramètre `family = "gaussian"` après normalisation de nos données.

Nous rappellons que nos variables explicatives sont normalisées, et que nous avons créé des variables *dummy* afin de pouvoir coder chacune de nos variables explicatives catégorielles en plusieurs variables numériques, prenant commes valeurs 0 ou 1 en fonction des valeurs prises par la variable catégorielle.
Nous rappellons aussi que nos variables cibles respectives sont les incréments de dépenses et les variations de charges, ces dernières pouvant être négatives. Pour des soucis de comparaisons, nous choisissons de fixer le paramètre `family = "gaussian"` pour nos deux problèmes de régression.

Le paramètre α sera testé pour des valeurs de 0 à 1, par pas de 0.05, et le paramètre λ sera testé de 10^{-5} à 1, pour 500 valeurs. Ces deux paramètres seront testés de manière conjuguée ; nous aurons donc 5500 combinaisons possibles par développement.
Nous utilisons une validation croisée à 10 sous-échantillons, et nous conserverons pour chaque développement le modèle obtenant la plus faible erreur de prévision du point de vue du RMSE.

Nous obtenons les paramètres optimaux suivants :

Dépenses						
Développement	2	3	4	5	6	7
Alpha	0.05	0	0	0.05	0	0.05
Lambda	0.00401	0.02405	0.05612	0.01604	0.04810	0.19038

Charges						
Développement	2	3	4	5	6	7
Alpha	0.05	0	0	0.05	0.05	0.05
Lambda	0.00201	0.00001	0.08016	0.12225	1	1

FIGURE 4.2 – Paramètres λ et α optimaux obtenus sur le triangle inclus.

Nous remarquons tout d'abord que la méthode Elastic Net semble ici privilégier la pénalisation Ridge (paramètre α égal à 0 ou 0.05) sur la plupart des développements de dépenses et de charges. Toutefois, la méthode Elastic Net inclut une part de pénalisation LASSO (paramètre α strictement supérieur à 0) ; notamment, sans cette part, les modèles obtiendraient des résultats moins précis du point de vue du RMSE.

Nous pouvons remarquer aussi une faible pénalisation pour les premières années de développement, qui croit au fur et à mesure des développements, notamment très rapidement sur les charges : nous passons d'un paramètre λ égal à 0.002 pour la prédiction du second développement à un λ égal à 1 pour les deux derniers développements.

Prenons par exemple le cas de la prédiction du 7ème développement pour les dépenses, où notre modèle retient un α égal à 0.05 et un λ égal à 0.19038. Si nous regardons le RMSE en phase d'apprentissage :

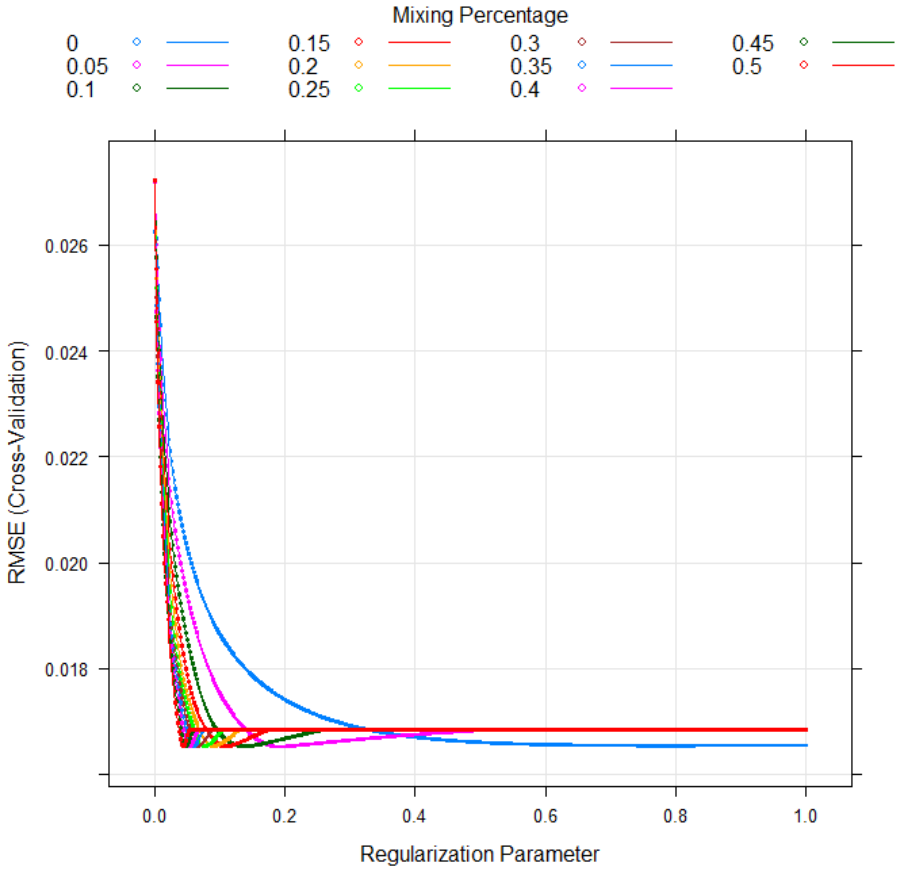


FIGURE 4.3 – Erreur d'apprentissage RMSE en fonction des paramètres λ (*Regularization Parameter)* et α (*Mixing Percentage*) sur le triangle inclus, pour la prédiction du 7ème développement des dépenses.

Nous remarquons une inflexion du RMSE se produisant pour toutes les valeurs possible du paramètre α entre des valeurs de λ comprises entre 0 et 0.2 : pour toutes les valeurs testées pour le paramètre α, nous accordons une part plus importante à la pénalisation Ridge qu'à la pénalisation LASSO.

Nous pouvons aussi regarder les trajectoires obtenues pour les coefficients β retenus en fonction de la valeur de λ, par exemple pour la prédiction de la deuxième année de développement des dépenses et de charges :

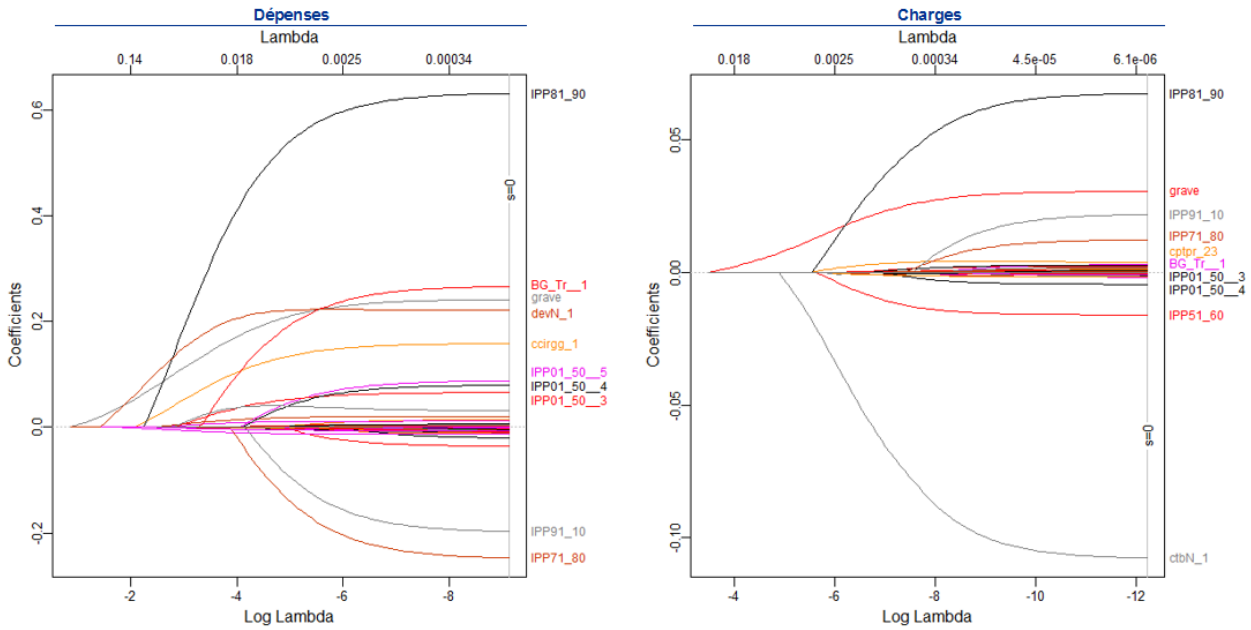


FIGURE 4.4 – Trajectoires des coefficients en fonction du paramètre λ.

Nous voyons notamment que les variables ayant le plus d'influence sont les variables indiquant la présence de victimes ayant un taux d'IPP élevé, la gravité du sinistre et la dépense ou la charge initiale lors du premier développement.
En effet, ces variables ont les paramètres β les plus fort en valeur absolue. Par exemple sur le graphique de droite pour les charges, pour des valeurs faibles de λ (à droite), les variables `ctbN_1` et `IPP81_90` auront des paramètres β associés plus élevés en valeurs absolues que ceux des autres variables. Ce seront ces variables qui auront des paramètres β non nuls pour des valeurs plus élevées de λ.

Une fois ces paramètres choisis sur le triangle échantillon, nous comparerons les résultats obtenus sur le triangle inclus initial et triangle total dans le chapitre III.

Les modèles GLM présentent de nombreux avantages, comme leur rapidité d'éxécution et leur interprétabilité.
Néanmoins, les modèles GLM souffrent de ne pas pouvoir représenter et incorporer des interactions non-linéaires. C'est notamment dans ce but que nous allons appliquer des méthodes d'apprentissage statistique non-linéaires.

Nous allons maintenant présenter le premier algorithme que nous allons utiliser : les arbres de régression.

Chapitre 5

Arbres de régression et forêts aléatoires

L'apprentissage par arbre de décision est une méthode classique d'apprentissage statistique. On appelle **arbre de régression** un arbre de décision dont la variable cible (ou d'intérêt) est **quantitative**. Dans le cas d'une variable d'intérêt qualitative, cet arbre est dit "de classification".

Dans ce chapitre, nous allons tout d'abord présenter succinctement les arbres de régression, avant de nous intéresser au principe de critère de segmentation.

Nous parlerons ensuite de l'élagage de l'arbre pour finir par les avantages et inconvénients de cette méthode d'apprentissage statistique.

5.1 Présentation des arbres de régression

un des buts de l'algorithme d'apprentissage par arbre de décision est la prédiction d'une variable cible à partir de valeurs de variables explicatives.

Il existe plusieurs types d'algorithme par arbre de décision, comme le C5.0, le CHAID, le MARS ou bien encore le CART.

Dans la suite, nous nous intéresserons à la méthode la plus classique, celle des arbres **CART** (*Classification and Rregression Tree*), créée par Breimann et al. en 1984 [8]. Nous étudierons cette méthode dans le cadre d'un problème de régression.

La méthode CART se concentre sur la création de partitions binaires récursives. L'espace est d'abord séparé en deux régions, en fonction de critères simples portant sur les variables explicatives, dans notre cas un seuil de valeurs. Ces deux régions sont appelées **nœuds**. L'algorithme cherche à construire deux populations à la fois les plus différentes possibles, en terme de valeurs de la variable à expliquer et qui présentent une homogénéité maximale au sein de chaque population : il cherche

à minimiser la variance intra-groupes tout en maximisant la variance intergroupe.
Les régions ainsi définies sont à leur tour séparées en deux nouvelles sous-régions, jusqu'à ce qu'un critère d'arrêt mette fin au développement d'une branche de l'arbre, où que les régions restantes ne contiennent plus qu'un seul individu. Ainsi, l'espace des variables explicatives sera divisé en sous-régions, et à tous les individus présents dans une sous-région sera affecté une valeur unique, qui sera en général la moyenne des valeurs cibles respectives des individus de cette sous-région.
Les nœuds terminaux sont appelés **feuilles** et représentent les sous-régions finales obtenues. La **profondeur** de l'arbre désigne le nombre maximum d'étapes de séparations au sein de l'arbre.

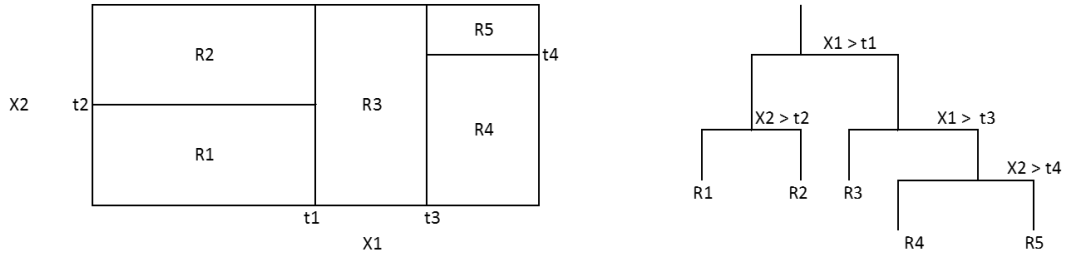


FIGURE 5.1 – Exemple de partitionnement d'un espace bidimensionnel et arbre associé. Nous avons ici un arbre à 4 nœuds, 5 feuilles, de profondeur 4.

Pour décider de la variable de séparation et du seuil de la valeur de cette variable à appliquer, on fixe un critère de segmentation.

5.2 Critère de segmentation

L'arbre réalise des choix successifs des variables explicatives. Cette segmentation se poursuit jusqu'à ce qu'il ne reste qu'un seul individu par branche, ou qu'un critère d'arrêt soit atteint, comme par exemple une homogénéité jugée optimale soit obtenue.

Dans le cadre de la régression, nous disposons de p variables explicatives $(X_1, \cdots, X_p) \subset X$, où X est un espace d'états, et d'une variable cible Y. L'algorithme CART doit décider automatiquement comment séparer l'espace des variables explicatives, ainsi que la topologie (ou la forme) de l'arbre.

Commençons tout d'abord par définir la segmentation d'un arbre. Soit A un nœud composé de N_A observations.
On appelle **segmentation** de A un couple (j, s) tel que :

1. $j \in \{1, \cdots, p\}$ est la variable explicative sur laquelle se fait la segmentation
2. s est la valeur de cette variable qui réalise le critère de segmentation : on sépare l'espace des individus entre ceux qui ont une valeur de j supérieure à s et ceux qui ont une valeur de j inférieur ou égale à s.

Supposons tout d'abord que nous ayons une partition de notre espace de variables explicatives en M régions $R_1, \cdots, R_M$ et que nous modélisons la réponse de l'algorithme comme une constante c_m (i.e. si un individu appartient à R_m, on lui affecte la valeur c_m).
Nous avons ainsi :

$$f(x) = \sum_{m=1}^{M} c_m \mathbb{1}(x \in R_m)$$

Si l'on prend comme critère d'erreur l'**erreur quadratique** :

$$\sum_i (y_i - f(x_i))^2$$

Il est aisément démontré que la valeur optimale de c_m n'est autre que la moyenne des valeurs de y_i sur la région R_m :

$$c_{moptimale} = moy(y_i | x_i \in R_m)$$

Toutefois, trouver la meilleure séparation binaire du point de vue de l'erreur quadratique n'est en général pas réalisable dans un temps de calcul raisonnable. L'algorithme suivant est en général utilisé :

1. On considère une segmentation (j, s)

2. On définit les demi-plans $R_1(j, s) = \{X | X_j \leq s\}$ et $R_2(j, s) = \{X | X_j > s\}$

3. On cherche la segmentation (j, s) qui résoud :

$$min_{j,s}(min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2)$$

Pour tout choix de (j, s), on peut minimiser 3. par

$$c_{1optimale} = moy(y_i | x_i \in R_1)$$

et

$$c_{2optimale} = moy(y_i | x_i \in R_2)$$

Ainsi, pour chaque variable de séparation j, trouver la valeur de séparation s peut se faire très rapidement. En analysant toutes les variables de séparation on peut donc trouver la segmentation (j, s) optimale.
Le processus de segmentation est répété ensuite sur les deux régions ainsi obtenues, jusqu'à ce qu'un critère d'arrêt soit atteint.

Un problème se pose néanmoins : jusqu'à quel niveau devrions-nous faire grandir l'arbre ? Avoir un arbre de trop petite taille pourrait empêcher de détecter la structure inhérente à nos données et nous donner des résultats de mauvaise qualité.

Toutefois, un arbre trop profond pourrait entraîner un phénomène de surapprentissage, qui survient quand notre algorithme d'apprentissage perd de sa puissance de généralisation en s'imprégnant trop des données d'apprentissage. Pour résoudre ce dilemne, nous introduisons par la suite le principe d'élagage.

5.3 Élagage

La profondeur de l'arbre est un paramètre gouvernant la complexité du modèle et doit être choisi en fonction des données.
Une approche pourrait être de créer un nouveau noeud seulement si la décroissance de l'erreur quadratique liée à la création de ce noeud dépasse un certain seuil. Toutefois, un problème notable lié à cette approche serait qu'une segmentation de premier abord sans valeur, et donc non conservée, aurait pu mener à une deuxième segmentation de bonne qualité.

La stratégie généralement utilisée est la suivante : faire croître un arbre T_0 le plus profond possible, en utilisant comme critère d'arrêt le nombre d'individus dans les feuilles. Cet arbre T_0 est ensuite **élagué**.
L'élagage consiste à enlever les noeuds de l'arbre qui n'apporteraient rien à l'analyse. Il faut toutefois conserver les qualités prédictives de l'arbre, et donc ne pas supprimer trop de noeuds.

L'élagage peut se résumer comme suit :
Supposons un sous-arbre T inclus dans T_0.
On définit le critère de **coût de complexité** :

$$C_\alpha(t) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha|T|$$

avec :

- $|T|$ le nombre de feuilles de T
- $N_m = card x_i \in R_m$
- $c_m = \frac{1}{N_m} \sum_{x_i \in R_m} y_i$
- $Q_m(t) = \frac{1}{N_m} \sum_{x_i \in R_m} (y_i - c_m)^2$

L'idée est simple : trouver, pour tout α, le sous-arbre T_α qui minimise C_α.
Le paramètre $\alpha \geq 0$ représente le compromis entre la taille de l'arbre et l'adéquation aux données. De grandes valeurs de α résultent dans des arbres T_α de faible profondeur, tandis que pour $\alpha = 0$ nous obtenons l'arbre entier T_0.
Nous allons enfin voir les avantages et inconvénients des arbres de régression.

5.4 Avantages et inconvénients

5.4.1 Avantages

Les arbres de régression présentent plusieurs avantages. Tout d'abord, et principalement, l'interprétabilité des résultats. Ils fournissent des résultats graphiques, par découpage de l'espace, et permettent de voir facilement comment les individus sont affectés à ces séparations.
De plus ils offrent un temps de calcul raisonnable et sont adaptés à des données mixtes et manquantes, et leur maniement est aisé de par le faible nombre de paramètres à optimiser, notamment comparé

à d'autres méthodes d'apprentissage statistique, et la présence de nombreux packages leurs étant dédiés.
Toutefois les inconvénients liés à cet algorithme sont nombreux.

5.4.2 Inconvénients

Le premier inconvénient, déjà énoncé, est le risque de surapprentissage. Il survient lorsque l'algorithme s'imprégne trop des caractéristiques de notre échantillon d'apprentissage et perd de son potentiel de généralisation. Le modèle fera des prédictions de moins bonne qualité sur un nouvel échantillon. Cet inconvénient n'est toutefois pas limité aux arbres de régression mais bien à l'ensemble des méthodes d'apprentissage statistique non-linéaires, et il doit être généralement monitoré en analysant les données et en appliquant des méthods d'agrégation et d'échantillonnages. Dans le cas des arbres de régression, ce risque augmente avec la taille de l'arbre. Il est ensuite réduit par l'élagage. Une autre méthode permet aussi de réduire ce risque : l'agrégation, un chapitre lui est consacré ultérieurement.

De plus, les arbres de régression sont sensibles aux *optima* locaux. Cela est dû au fait que l'algorithme parcourt les différentes variables explicatives de façon successive, et un critère de segmentation n'est plus réétudié par la suite. Ainsi, la construction de l'arbre en amont peut déterminer la construction de tout le reste de l'arbre. Cela peut notamment entraîner un manque de robustesse et une grande variabilité de résultats. L'agrégation permet aussi de réduire ce risque.

Avant d'appliquer les arbres de régression à notre modèle, nous allons présenter comment améliorer la robustesse des arbres de régression grâce à l'agrégation, en particulier les forêts aléatoires.
Ce sont ces dernières que nous utiliserons pour projeter les dépenses et les charges.

5.5 Agrégation de modèles et forêts aléatoires

Les algorithmes d'agrégation de modèles, basés sur des stratégies adaptatives ou aléatoires, permettent d'améliorer l'ajustement et de réduire le surapprentissage par une combinaison d'un grand nombre de modèles.
Nous allons présenter les deux méthodes les plus populaires, le *bagging* et le *boosting*.

5.5.1 Bagging

Développé par Breiman [6], le **bagging** (pour *Bootstrap Aggregation*) est un algorithme d'agrégation de modèle basé sur une stratégie aléatoire.
Son principe est simple : on cherche à moyenner les prévisions de plusieurs modèles indépendants, ce qui permettrait de réduire la variance de prévision.

5.5.1.1 Principe

Revenons à notre problème de régression.
Nous avons un échantillon $z = \{(X_1, y_1), \cdots, (X_n, y_n)\}$ avec X le vecteur des variables explicatives, et y la variable cible. Considérons un estimateur f^* de y en fonction de X.
Considérons B échantillons indépendant z_b, $b = 1, \cdots, B$. Nous construisons l'agrégation comme :

$$f^*_B(.) = \frac{1}{B} \sum_{b=1}^{B} f^*_{z_b}(.)$$

Toutefois, dans la pratique, il ne serait pas réaliste de considérer un nombre important d'échantillons indépendants, car cela nécessiterait trop de données. Ils sont donc remplacés par des échantillons *bootstrap* obtenus par des tirages aléatoires avec remise de n individus parmi n.
L'erreur **out-of-bag** de l'erreur de prévision permet de contrôler le nombre de ces échantillons. Celle-ci décroit en fonction du nombre de modèles avant de se stabiliser en indiquant le nombre de modèles ou échantillons bootstrap nécessaires à l'agrégation.

Le *bagging* offre notamment de bons résultats pour l'agrégation d'algorithmes avec forte variance et faible biais, comme les arbres de régression.

5.5.1.2 Cas des forêts aléatoires

On définit une forêt aléatoire comme une agrégation par *bagging* d'arbres de décision. Cet algorithme d'agrégation, proposé par Breiman [7], consiste en l'amélioration du bagging par ajout d'une composante aléatoire.
L'objectif est de rendre les arbres décorrelés les uns des autres lors de l'agrégation en ajoutant du hasard dans le choix des variables intervenant dans les modèles.

L'algorithme peut se résumer comme suit :

1. Pour $b = 1$ jusqu'à B :
   a. Tirer un échantillon *bootstrap* Z de taille N sur l'échantillon d'apprentissage
   b. Produire un arbre T_b de la forêt aléatoire en répétant de façon récursive les étapes suivantes pour chaque feuille de l'arbre, jusqu'à ce que la taille minimum n_{min} de feuille soit atteinte :
   i. Sélectionner m variables au hasard parmi les p variables explicatives
   ii. Choisir la segmentation (j, s) optimale
   iii. Réaliser la séparation du nœud
2. On obtient alors un ensemble d'arbre $\{T_b\}_1^B$

Pour faire une prédiction sur un nouveau point x :

$$\hat{f}^B_{rf}(x) = \frac{1}{B} \sum_{b=1}^{B} T_b(x)$$

Les arbres de régression sont des candidats idéaux pour le *bagging*, car ils peuvent capturer des interactions complexes dans les données et ont un biais relativement faible s'ils sont assez profonds.

Comme chaque arbre généré dans le *bagging* est identiquement distribué, la valeur espérée moyenne de B arbres est la même que la valeur espérée de chacun de ces arbres. Ainsi, le biais de chaque arbre est identique au biais des arbres obtenus par *bagging*. L'axe d'amélioration de l'algorithme de forêt aléatoire par rapport à un arbre de régression seul porte donc sur la réduction de variance.

5.5.2 Boosting

Le **boosting** est un processus adaptatif (qui se modifie à chaque itération) d'agrégation de modèles, initialement créé par Schapire [43] puis développé par Freund et Shapire [15].

Initialement conçu pour la classification, il peut toutefois être étendu aux problèmes de régression. La motivation première du *boosting* est de combiner les prédictions de plusieurs "faibles" prédicteurs. En ce point le *boosting* est similaire au *bagging*, toutefois il en diffère par plusieurs points.
La principale différence se situe dans l'utilisation d'un système de poids.
Á chaque itération, l'algorithme associe une importance plus grande aux individus mal classés qu'à l'étape précédente. Le poids d'un individu ne change pas s'il est bien classé.
Divers algorithmes sont possibles en fonction de comment sont calculés les poids. Par exemple, l'algorithme AdaBoost.M2, présenté par Freund et Shapire [15], s'écrit de la façon suivante :

1. Input : Soit une séquence de m exemples $(x_1, y_1), \cdots, (x_m, y_m)$, où $x_i \subset X$ sont les variables explicatives et y_i la variable à prédire.
   Soit f la fonction de régression obtenue par un prédicteur "faible" M sur l'échantillon d'apprentissage, et T le nombre d'itérations de boosting.
2. Initialisation des poids : soient $B = \{(i, y) : i \in \{1, \cdots, m\}, y \neq y_i\}$ et $D_1(i, y) = \dfrac{1}{|B|}$ pour $(i, y) \in B$.
3. Pour $t = 1$ jusqu'à T :
   — a. Entraîner M sur D_t
   — b. Obtenir la fonction de régression f_t
   — c. Calculer la **pseudo-perte** de h_t : $\epsilon_t = \dfrac{1}{2} \sum_{(i,y)\in B} D_t(i, y)(1 - h_t(x_i, y_i) + h_t(x_i, y))$
   — d. Poser $\beta_t = \dfrac{\epsilon_t}{(1 - \epsilon_t)}$
   — e. Actualiser $D_{t+1}(i, y) = \dfrac{D_t(i, y)}{Z_t} . \beta_t^{\frac{1}{2}(1+h_t(x_i,y_i)-h_t(x_i,y))}$, où Z_t est une constante de normalisation
4. On obtient $h_{fin}(x) = arg_{y\in Y} max \sum_{t=1}^{T} (log \dfrac{1}{\beta_t}) h_t(x, y)$

Contrairement au *bagging*, le *boosting* permet de réduire le biais en plus de la variance. Toutefois, il est sensible au bruit ainsi qu'aux valeurs extrêmes, ce qui est notamment dû à la concentration de l'algorithme sur les populations mal classées.

Nous allons maintenant appliquer les forêts aléatoires à notre modèle de provisionnement individuel.

5.6 Forêts aléatoires - Choix des paramètres optimaux sur le triangle inclus dans le triangle échantillon

Nous présentons ici les résultats obtenus par les forêts aléatoires sur le triangle inclus dans le triangle échantillon.
Nous utilisons le package R `ranger`.

Pour chacun de nos triangles de dépenses et de charges, nous optimiserons les paramètres suivants :

— `mtry` : le nombre de variables testées à chaque segmentation,
— `splitrule` : la règle de segmentation,
— `min.node.size` : le nombre d'individus dans les feuilles terminales.

Le premier paramètre dépendra de l'année de développement, et variera donc en fonction du temps. Les deux autres paramètres seront fixes sur tous les développements, et seront testés de manière conjointe.

Nous allons commencer par regarder les résultats du choix de variables par l'algorithme RFE.

5.6.1 Importance des variables et choix des variables explicatives par RFE

Nous rappelons que nous utilisons l'algorithme RFE, pour la prédiction du deuxième développement sur le triangle initial, dans le but de retenir les variables les plus pertinentes.
Nous rappelons aussi que nous cherchons à prédire les incréments de dépenses et les variations de charges, que la variable `dev_N` correspond à l'incrément de dépenses au développement d'année N, tandis que la variable `ctb_N` correspond à la variation de charge estimée par les gestionnaires au développement d'année N.

Nous calculons ici l'importance des variables pour les forêts aléatoires par le critère dit "réduction d'impureté" : à chaque segmentation de chaque arbre présent dans la forêt aléatoire, la réduction de l'erreur quadratique attribuée à la variable (ou les variables) associée(s) avec cette segmentation constitue la mesure d'importance. Nous aggrégeons ensuite pour chaque variable les importances correspondantes sur chaque sous-arbre de notre forêt.
Nous représenterons nos valeurs d'importances sur une échelle relative de 0 à 100, 0 représentant une importance nulle tandis que la variable la plus importante aura une importance fixée à 100.

Nous obtenons les résultats suivants pour les dépenses, pour les forêts aléatoires :

Variable	Importance relative
Grave	100,00
devN_1	55,29
Cour_Appel	39,50
Cat_Vehicule	31,40
IPP01_50_surv	23,18
Type_Conducteur	22,33
Nbvic_surv	17,49
Sit_PRQ	13,45
Part_Respons	12,60
Convention	10,40
Code_Clause	9,64
Sit_PAC	9,23
Sexe	9,07
Presence_REC	8,19
Part_Respons_Conv	8,10
Presence_RCM	7,71
Presence_DOM	7,18
Trauma_surv	6,50
IPP91_100_surv	4,13
Clos_surv	3,83
Circ_Aggrav	2,96
IPP61_70_surv	2,46
Nature_Obs	1,47
IPP51_60_surv	1,31
IPP81_90_surv	1,03
Tetra_surv	0,53
Code_Proj	0,18
Para_surv	0,16

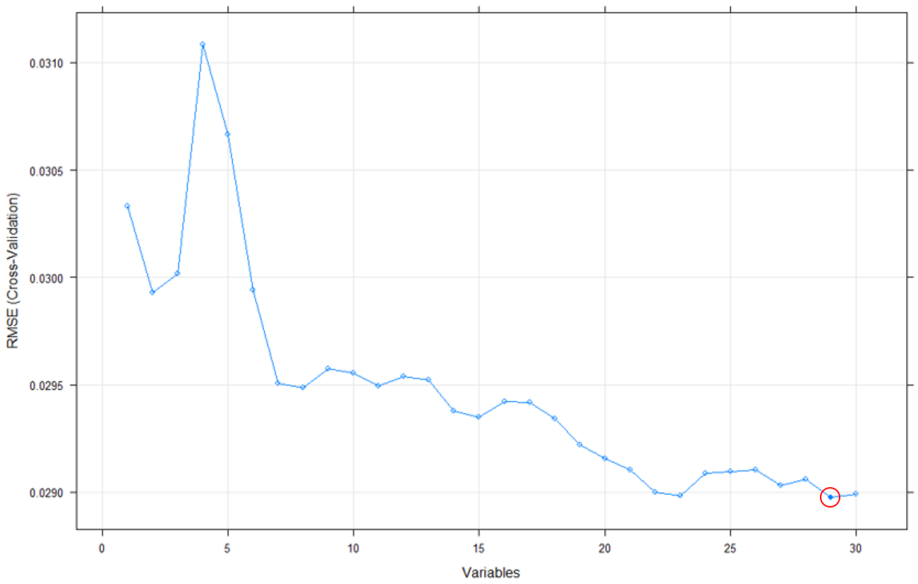


FIGURE 5.2 – Résultat du RFE pour les dépenses, pour les forêts aléatoires. A gauche, l'importance des variables calculée selon la décroissance d'impureté dans les nœuds.

Pour les dépenses, nous retenons presque toutes les variables. Nous notons que la variable considérée comme la plus importante est la variable `grave`, suivie de la dépense au premier développement (variable `devN_1`). Nous notons aussi que les variables `Cour_Appel` (Cour d'appel) et `Cat_Vehicule` (catégorie de véhicule) sont considérées comme importantes ; toutefois ces deux variables présentent un nombre important de modalités, et le calcul d'importance des variables choisi pour les forêts aléatoires peut être biaisé par le nombre de modalités.
Il semble intéressant de remarquer l'importance relative de la variable `Grave`, tandis que les variables indiquant la présence d'une victime avec un fort taux d'IPP, comme la variable `IPP_81_90_surv` est considérée comme très peu importante dans notre modèle.

Nous obtenons les résultats suivants pour les charges :

Variable	Importance relative
ctbN_1	100,00
Grave	43,62
Cat_Vehicule	40,96
Cour_Appel	40,43
Part_Respons	18,62
Type_Conducteur	17,55
Nbvic_surv	14,89
Sit_PRQ	14,36
Convention	13,83
IPP01_50_surv	12,50
Code_Clause	12,39
Part_Respons_Conv	11,65

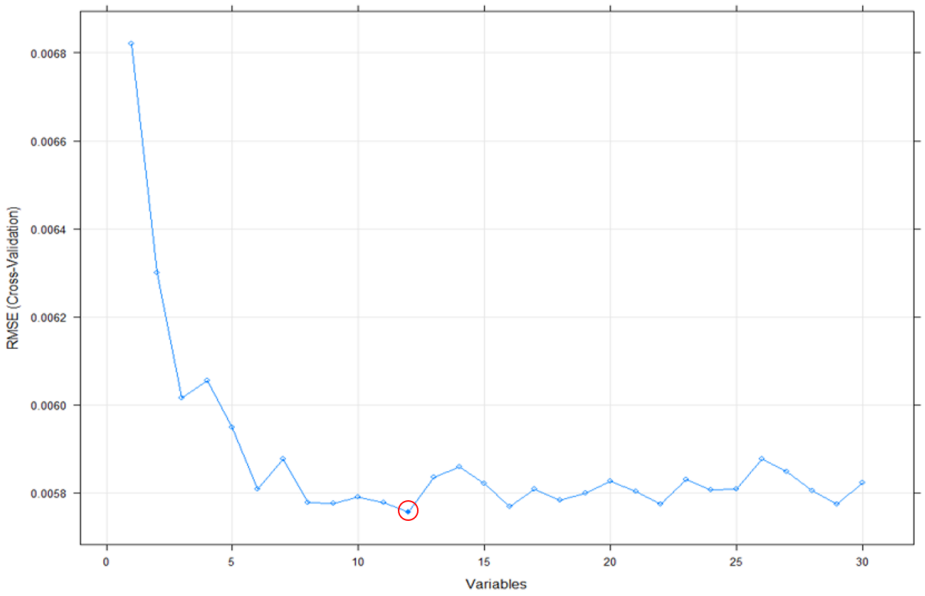


FIGURE 5.3 – Résultat du RFE pour les charges, pour les forêts aléatoires. A gauche, l'importance des variables calculée selon la décroissance d'impureté dans les nœuds.

Pour les charges, nous retenons nettement moins de variables que pour les dépenses, seulement 12. Nous notons que la variable considérée comme la plus importante est cette fois-ci la variable `ctbN_1`, qui est la charge au premier développement, suivie de la variable grave. Nous retrouvons encore une fois les variables `CA` et `CATVEH` comme relativement importantes.

Nous avons utilisé comme paramètres *a priori* :
— Nombre d'arbres par forêt : 500
— `mtry` $= \sqrt{nombredevariables} = 5$
— `min.node.size` $= 1$
— `splitrule` = variance

Le paramètre `mtry` a ici été fixé à la valeur $\sqrt{p}$ où p est le nombre de variables explicatives. C'est la valeur généralement optimale que nous pouvons trouver dans la littérature. Ici nous avons 30 variables explicatives, en comptant la variable `dev.n_N` ou `ctb.n_N`.
Le nombre d'arbres sera fixé à 500, après des tests préalables, car il est suffisament important pour assurer une stabilité des résultats.

5.6.2 Paramètre `mtry` : nombre de variables testées à chaque division

Nous testons le paramètre `mtry` pour des forêts de 500 arbres.

Cette variable sera dynamique temporellement, et évoluera en fonction du développement.

Nous présentons les résultats avec un gradient de couleur, tendant vers le vert pour les RMSE les plus faibles et vers le rouge pour les RMSE les plus forts. Nous obtenons les résultats suivants, pour le RMSE obtenu en phase d'apprentissage après validation croisée à 10 échantillons :

Dépenses

mtry\dev	2	3	4	5	6	7
1	0,04106	0,01692	**0,00911**	0,00466	**0,00442**	**0,01253**
2	0,03800	0,01605	0,00920	**0,00465**	0,00442	0,01314
3	0,03674	**0,01593**	0,00931	0,00470	0,00450	0,01402
4	0,03650	0,01603	0,00939	0,00480	0,00461	0,01423
5	**0,03645**	0,01609	0,00957	0,00484	0,00471	0,01472
6	0,03667	0,01626	0,00957	0,00490	0,00477	0,01496
7	0,03680	0,01634	0,00988	0,00493	0,00486	0,01511
8	0,03695	0,01643	0,00969	0,00498	0,00490	0,01561
9	0,03705	0,01655	0,00964	0,00510	0,00494	0,01528
10	0,03702	0,01658	0,00964	0,00514	0,00505	0,01507
mtry retenu	5	3	1	2	1	1

Charges

mtry\dev	2	3	4	5	6	7
1	**0,00497**	0,00618	**0,00433**	**0,00297**	**0,00427**	**0,00113**
2	0,00498	0,00619	0,00454	0,00305	0,00447	0,00124
3	0,00505	0,00613	0,00468	0,00310	0,00465	0,00129
4	0,00505	**0,00604**	0,00480	0,00326	0,00464	0,00134
5	0,00515	0,00613	0,00496	0,00331	0,00466	0,00135
6	0,00517	0,00619	0,00502	0,00335	0,00462	0,00137
7	0,00519	0,00621	0,00509	0,00345	0,00456	0,00134
8	0,00521	0,00622	0,00518	0,00347	0,00470	0,00135
9	0,00525	0,00627	0,00525	0,00359	0,00467	0,00134
10	0,00528	0,00632	0,00528	0,00355	0,00463	0,00135
mtry retenu	1	4	1	1	1	1

FIGURE 5.4 – Choix du paramètre mtry optimal, par développement.

Nous remarquons notamment que pour les dépenses, nous obtenons un paramètre `mtry` de plus en plus bas au fil des développements, tandis que pour les charges, nous observons un paramètre `mtry` constant à 1 sauf pour la prédiction de l'année de développement 3.

Une fois le paramètre `mtry` fixé pour chaque année de développement, nous allons étudier la sensibilité des prédictions à d'autres paramètres : le nombre d'individus dans les feuilles terminales et la règle de segmentation.

5.6.3 Optimisation simultanée : paramètre `min.node.size` et paramètre `splitrule`

Nous allons maintenant optimiser deux paramètres en même temps :
- le paramètre `min.node.size` représente le nombre d'individus minimum dans les feuilles terminales, et est assimilable à un critère d'arrêt. Il peut aussi permettre de réduire le risque de surapprentissage, toutefois un paramètre `min.node.size` trop important pourrait "mélanger" des valeurs extrêmes à des valeurs plus normales, ce qui ne serait pas souhaitable.
- le paramètre `splitrule` représente la règle de segmentation.

Le paramètre `min.node.size` a 5 comme valeur par défaut, pour la régression. Nous testerons les valeurs 1, 5, 10 et 20.
Le paramètre `splitrule` peut prendre comme valeurs :
- `variance` : la règle de segmentation basée sur l'erreur quadratique, comme décrit plus haut,
- `extratrees` : règle de décision basée sur l'algorithme Extratrees développé par Geurts et al. [20],
- `maxstat` : règle de décision basée sur des statistiques de rang maximum, d'après Wright et al. [49]

Nous obtenons les résultats suivants pour les dépenses et les charges :

Dépenses				
Règle\mns	1	5	10	20
Variance	22 809	22 926	**22 701**	23 094
Extratrees	23 733	23 710	23 758	23 736
Maxstat	25 426	25 417	25 424	25 408

FIGURE 5.5 – Choix du paramètre mtry optimal, par développement.

Charges				
Règle\mns	1	5	10	20
Variance	**52 227**	52 523	52 277	52 964
Extratrees	54 823	54 342	54 627	54 888
Maxstat	60 334	60 352	60 297	60 304

FIGURE 5.6 – Choix du paramètre mtry optimal, par développement.

Nous retenons le critère de segmentation `variance` pour les dépenses comme pour les charges. Pour le paramètre `min.node.size`, nous retenons une valeur égale à 10 pour les dépenses et 1 pour les charges : les forêts aléatoires semblent considérer devoir séparer plus précisément les charges, qui contiennent plus de valeurs extrêmes que les dépenses.

5.6.4 Vérification de la pertinence du choix de variables par la méthode RFE

Une fois tous nos paramètres fixés, il convient de vérifier la pertinence du choix de variables par la méthode RFE. En effet, par choix méthodologique nous avons appliqué cette méthode avec des paramètres par défauts, pour le passage de la première année de développement à la seconde. Or l'importance des variables variera au fur et à mesure des développements ; il convient donc de vérifier que l'application de la méthode RFE était justifiée en comparant les erreurs de prédictions obtenues par le modèle n'utilisant que les variables retenues par la méthode RFE, et le modèle utilisant toutes les variables initiales.

En utilisant nos paramètres retenus lors des étapes précédentes, nous obtenons les RMSE suivants pour le dernier développement :

Dépenses	
Choix de variables	RMSE du dernier développement
Avec RFE	25 204
Avec toutes les variables initiales	25 376

Charges	
Choix de variables	RMSE du dernier développement
Avec RFE	59 715
Avec toutes les variables initiales	61 900

FIGURE 5.7 – Comparaison des erreurs de prédictions pour le dernier développement, entre le modèle utilisant toutes les variables initiales et le modèle utilisant les variables conservées par l'algorithme RFE.

Nous observons donc qu'en utilisant seulement les variables obtenues par l'application de l'algorithme RFE à la place de toutes les variables, notre modèle commet moins d'erreurs.
Bien que la réduction d'erreur semble relativement faible, respectivement de 0,7% sur les dépenses et 3,5% sur les charges, l'application de la méthode RFE a permis à la fois de gagner en précision et de réduire le temps de calcul pour optimiser les paramètres de nos forêts aléatoires.

Une fois ces paramètres choisis à partir du triangle échantillon, et la vérification de la pertinence de la méthode RFE, nous pouvons utiliser notre modèle de prédiction sur le triangle initial inclus et initial total. Les résultats obtenus sont présentés dans le chapitre III avec les paramètres retenus en amont.

Nous allons maintenant présenter un autre algorithme d'apprentissage statistique : les réseaux de neurones.

Chapitre 6

Réseaux de neurones

Une autre méthode non-paramétrique est celle des réseaux de neurones artificiels. Ces réseaux peuvent être notamment vus comme des modèles de régression non-linéaire avec architecture.

Dans cette partie, nous introduirons tout d'abord les réseaux de neurones, puis leurs principes généraux et leur architecture générale. Nous présenterons ensuite le mode d'apprentissage de ces réseaux. Nous conclurons par la présentation de certains modèles classiques.

6.1 Brève introduction des réseaux de neurones

Le terme "réseau de neurones" trouve ses origines dans les tentatives de créer des modèles mathématiques pour représenter le traitement de l'information par les systèmes biologiques.
Développés en 1943 par McCulloch & Pitts [33], un neuro-physiologiste, et Pitts, un logicien, les neurones formels étaient censés imiter les neurones biologiques humains et être capables de mémoriser des fonctions booléennes simples. Inspirés par le système nerveux, les réseaux de neurones artificiels doivent être capables d'apprendre, mémoriser, généraliser l'information dans les connexions entre neurones et traiter l'information incomplète.

Les réseaux neuronaux fournissent des outils et algorithmes performants pour la prédiction, l'interpolation, la classification, la segmentation, le clustering et la reconnaissance de formes. Toutefois, il existe diverses difficultés liées à l'utilisation de ces réseaux : réglage des paramètres délicat, puissance informatique requise, risques de sur-apprentissage et de convergence vers un minimum local, ainsi que le côté "boîte noire", ...

6.2 Principes généraux et architecture

Un réseau neuronal peut être vu comme un graphique orienté pondéré. Il peut être représenté comme un ensemble de nœuds, appelés *neurones*. Ces neurones sont connectés entre eux et chaque unité est activée en recevant une donnée et en en renvoyant une autre. Les neurones sont organisés

en plusieurs niveaux appelés couches. Chaque couche envoie ses données à la couche suivante. Un réseau peut comporter plusieurs neurones dans la couche de sortie, dans le cas de prédiction d'une variable catégorielle.
Par la suite, nous considérons un réseau neuronal comportant n neurones.

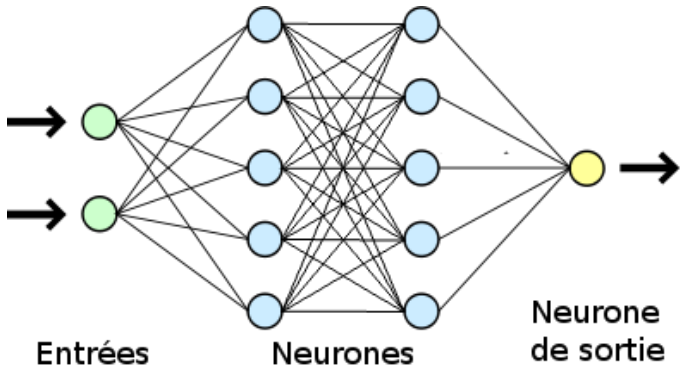


FIGURE 6.1 – Représentation simple d'un réseau de neurones

Nous allons présenter les principaux concepts définissant un réseau de neurones :

6.2.1 Activation

Chaque neurone formel est doté d'un état interne appelé l'*activation*. Notons $a_1, \cdots, a_n$ les activations respectives des neurones de notre réseau. On appelle ainsi **vecteur d'activation** le vecteur $a = (a_1, \cdots, a_n)$.

6.2.2 Poids synaptiques

On appelle **lien synaptique** l'arc entre deux neurones formels, et **poids synaptique** la pondération associée à ce lien. Le poids synaptique du neurone j vers le neurone i sera noté $w_{i,j}$, $(i, j \in [1, n])$, avec $w_{i,j} = 0$ s'il n'existe pas de lien entre les neurones i et j. La matrice des poids synaptiques s'écrit :

$$W_{i,j} = (w_{i,j})_{1 \leq i,j \leq n}$$

Elle mesure ainsi la connectivité du réseau.

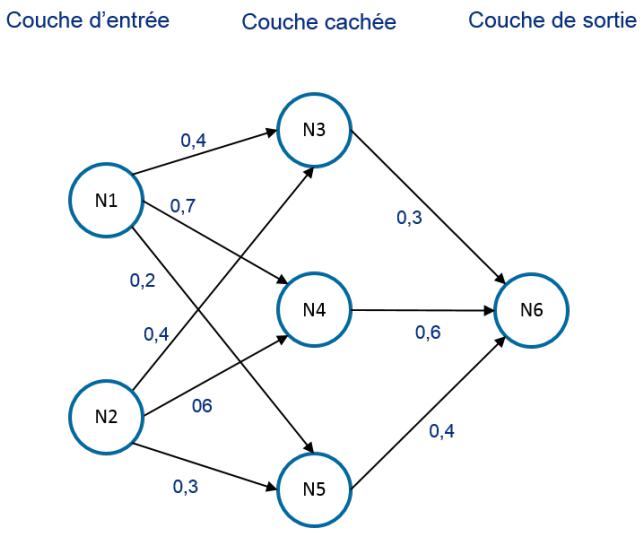


FIGURE 6.2 – Représentation d'un réseau de neurones à six neurones (2 dans la couche d'entrée, 3 dans la couche cachée, 1 dans la couche de sortie et les poids associés.

Si nous écrivons la matrice des poids associée au réseau représenté dans la figure 4.2 :

$$W = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0,4 & 0,4 & 0 & 0 & 0 & 0 \\ 0,7 & 0,6 & 0 & 0 & 0 & 0 \\ 0,2 & 0,3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0,3 & 0,6 & 0,4 & 0 \end{pmatrix}$$

6.2.3 Seuil et fonction d'entrée

Une fonction h_i du neurone i dite **fonction d'entrée** est donnée. Elle sera en général commune à tous les neurones d'une même couche et sera dans ce cas notée simplement h. Cette fonction dépend du vecteur d'activation a et de la matrice des poids W.

Lorsque le vecteur d'activation a est binaire, la fonction d'entrée h sera booléenne et pourra dépendre d'un biais ϕ. Dans le cas où a est numérique, la fonction h est généralement soit linéaire (cas $\phi_i = 0$)) :

$$h(i) := h(i, a, W) := W_i.a = \sum_{j=i}^{n} w_{i,j} a_j$$

soit affine :

$$h(i) := \sum_{j=1}^{N} w_{i,j} a_j - \phi i = \sum_{j=0}^{n} w_{i,j} a_j$$

en posant $w_{i,0} = \phi_i$ et $a_0 = -1$.
De plus, on notera $e_i = h(i, a(t), W)$ l'activation pondérée, par la matrice des poids W, du neurone i à l'instant $t \geq 0$.

6.2.4 Fonction d'activation

Chaque neurone possède une règle d'activation appelée **fonction d'activation**, notée f_i. Elle s'applique à l'activation pondérée e_i. Á l'instant $t \geq 0$, nous avons :

$$a_i(t) = f_i(e_i(t-1)) = f_i\left(\sum_{j=1}^{n} w_{i,j} a_j(t-1)\right)$$

Cette fonction d'activation a plusieurs caractéristiques :
- monotone ;
- à seuils ;
- dérivable (pour l'apprentissage).

Dans les fonctions d'activation classiques, nous retrouvons :
- les fonctions linéaires : $f(x) = \lambda x$;
- les fonctions de type sigmoïde exponentielle : $f(x) = \dfrac{1}{1+e^{-x}}$;
- les fonctions de type sigmoïde tangentielle : $f(x) = \dfrac{1-e^{-2x}}{1+e^{-2x}}$, qui converge plus vite que la sigmoïde exponentielle.

Si l'on trace la fonction $f(wx) = \dfrac{1}{1+e^{-wx}}$ pour $w = 1/3$, 1 ou 3, nous obtenons le graphique suivant :

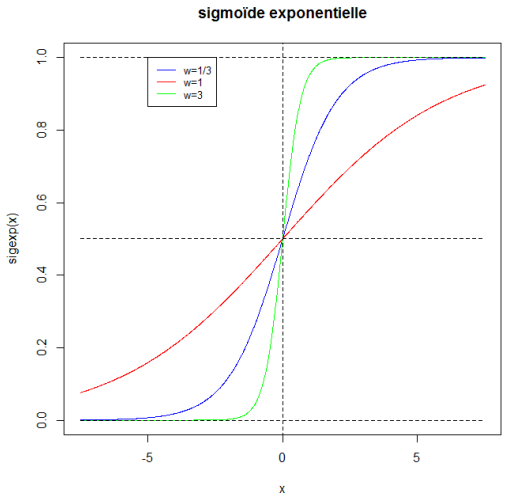


FIGURE 6.3 – Fonction sigmoïde exponentielle

6.2.5 Architecture

L'architecture du réseau de neurones dépend de la connectivité du réseau (et donc de la matrice de poids $W_{i,j}$), du nombre et du type de neurones. Nous pouvons décrire différentes caractéristiques possibles :
- Le réseau est dit **récurrent** s'il comporte des boucles (i.e. si il existe i tel que $w_{i,i} \neq 0$).

— Le réseau peut être organisé en plusieurs couches (une couche d'entrée, une ou plusieurs couches cachées et une couche de sortie) ou en une seule couche. Si le réseau comporte plusieurs couches cachées, on parle de **réseau profond**. Dans ce cas, on peut aussi spécifier les connectivités inter et intra-couches.
— Le réseau peut comporter des symétries de connexions (si la matrice des poids W est symétrique, ce qui est le cas dans le réseau de Hopfield par exemple).

En fonction du problème à résoudre, on doit donc se fixer les paramètres suivants :
— une architecture (nombre de neurones, de couches et connectivité) ;
— une matrice de poids W ;
— les fonctions d'entrée et d'activation qui seront en général identiques pour tous les neurones d'une même couche.

Ces choix sont guidés par le type d'application à traiter, ainsi que par les données considérées et les outils de programmation.
Nous allons par la suite étudier certains modèles parmi les plus classiques, connus pour répondre à des types d'applications particulières.

6.3 Réseaux de neurones et apprentissage

Un réseau de neurones peut fonctionner selon deux modes : en reconnaissance, dans lequel le réseau est utilisé pour calculer, ou bien en apprentissage.
En mode apprentissage, le réseau s'adapte à l'application à laquelle il est destiné à l'aide d'exemples. Pour cela, il s'appuie sur plusieurs paramètres : la matrice de poids W, le nombre de neurones et de couches, un "pas d'apprentissage" ainsi que des fonctions d'activation. C'est ce dernier mode qui nous intéresse dans cette étude.

Plus précisément, les valeurs données au préalable à la matrice de poids W sont purement arbitraires et ne sont habituellement pas adaptées au problème donné. Elles doivent donc être modifiées pour les adapter à l'application.
Cette correction est effectuée grâce au choix d'une **règle d'apprentissage**.

Il existe différents types d'apprentissage, notamment l'apprentissage supervisé et l'apprentissage non-supervisé. Nous considérons dans la suite un échantillon d'apprentissage C.

6.3.1 Apprentissage supervisé

Dans ce type d'apprentissage, l'échantillon d'apprentissage C est un ensembles de couples (x, d), où x est le vecteur des entrées présentées au réseau et d le vecteur des sorties attendu que nous connaissons à priori. Ainsi, l'entrée x est présentée au réseau et nous obtenons un vecteur s de valeurs en sortie. Le but est de modifier la matrice des poids W afin de minimiser une fonction d'erreur E comparant le vecteur de sorties attendu d et celui obtenue s. La matrice finale W^* ainsi obtenue doit toutefois posséder de bonnes propriétés de généralisation.

6.3.2 Apprentissage non-supervisé

Dans l'apprentissage non-supervisé, le vecteur des sorties d n'est pas connu. Il convient donc de modifier la matrice de poids W afin que les entrées $x_1, \cdots, x_n$ soient suffisamment regroupés, et de s'assurer que la généralisation est correcte.
L'apprentissage non-supervisé est souvent utilisé pour des tâches de classification et de *clustering*, où l'on cherche à créer des classes d'individus homogènes.
Parmi les réseaux de neurones opérant en apprentissage non-supervisé, nous pouvons citer les SOM (*Self-Organizing Map*), ainsi que les *Generative adversarial networks* (GANs), qui consistent en deux réseaux de neurones œuvrant en compétition : l'un génère les candidats au problème et l'autre les évalue (voir Ian Goodfellow et al. [21]).

6.3.3 Fonction d'erreur

Considérons le vecteur de sorties attendu $d = (d_1, \cdots, d_m)$ et celui obtenu $s = (s_1, \cdots, s_m)$. Les fonctions d'erreurs les plus usitées sont :

— L'erreur quadratique : $E = \frac{1}{2} \sum_{i=1}^{m} (d_i - s_i)^2$;
— L'erreur de corrélation croisée : $E = -\sum_{i=1}^{m} (d_i ln s_i + (1 - d_i) ln(1 - s_i))$

Cette dernière a comme avantage de converger plus rapidement que l'erreur quadratique.

Nous reviendrons plus en profondeur sur la fonction d'erreur et l'apprentissage supervisé après avoir introduit des notions de convergence.

6.3.4 Test et validation

Une fois l'apprentissage réalisé, il convient de s'assurer que le réseau réagit correctement à la généralisation. On utilise un nouvel échantillon de données, dit échantillon test, ainsi qu'un échantillon de validation.

Pour la méthode de validation :

— On réalise l'apprentissage sur l'échantillon d'apprentissage ;
— On suspend régulièrement l'apprentissage pour vérifier la généralisation grâce à l'échantillon test ;
— On valide *a posteriori* grâce à l'échantillon de validation, qui n'a donc pas été utilisé pour l'apprentissage.

Ainsi, l'échantillon test est utilisé pendant la phase d'apprentissage pour vérifier que le réseau de neurones garde une bonne généralisation. Il est parfois considéré comme une partie de l'échantillon d'apprentissage. L'échantillon de validation est, lui, utilisé après la période d'apprentissage. Il permet de vérifier le bon niveau de généralisation du réseau *a posteriori*.
Généralement, on répartit les données de la façon suivante :

— 70% pour l'apprentissage
— 20% pour l'échantillon test
— 10% pour l'échantillon de validation

Toutefois, dans cette étude, nous n'avons pas besoin de cette séparation, car nous utilisons directement une méthode de validation croisée à 10 sous-échantillons.

6.3.5 Convergence vers un point fixe

Système dynamique On s'intéresse ici à la convergence du réseau de neurones.
Le réseau de neurones peut être vu comme un système dynamique. En effet, pour tenir compte de tous les types de fonctions d'activation, nous pouvons écrire l'équation générale de l'activation $a_i(t+1)$ du neurone i à l'instant $t+1$:

$$a_i(t+1) = a_i(t) + f(h(i)(t), \Theta_i, I(t))$$

avec f la fonction d'activation du neurone i, $h(i)(t)$ la fonction d'entrée à l'instant t, Θ_i le seuil et $I(t)$ une valeur d'entrée à l'instant t. On peut le réecrire sous la forme :

$$\frac{dx}{dt} = F(x, W, \Theta, I)$$

ce qui permet ainsi d'étudier l'évolution du réseau comme un système dynamique.

Classes de systèmes dynamiques Nous introduisons ici quelques notions de convergence des systèmes dynamiques.

On appelle **point fixe attracteur** un point fixe entouré d'une région où toutes les trajectoires convergent vers ce point. Les systèmes dynamiques peuvent être classés en trois catégories :
- convergent, si chaque trajectoire approche un état d'équilibre ;
- oscillatoire, si chaque ensemble limite (i.e. de fin de trajctoire) est une orbite périodique (éventuellement stationnaire ;
- chaotique, s'il est difficile de faire une prédiction fiable à long terme.

A part certains cas très spécifiques, les réseaux de neurones sont convergents.

Convergence vers un point fixe Le critère de convergence des réseaux de neurones repose sur l'existence de la **fonction de Lyapunov**. Une fonction de Lyapunov est une fonction à valeurs réelles sur l'espace des états du système qui décroit strictement en temps le long de chaque trajectoire non stationnaire et qui possède une borne inférieure.

En général, la fonction de Lyapunov est supposée négative. La fonction d'erreur utilisée ci-après dans la rétropropagation est notamment une fonction de Lyapunov.

Un état a sera un point d'équilibre du système si c'est un point critique d'une fonction de Lyapunov E du système, c'est à dire que le gradient de la fonction de Lyapunov E s'annule en a. E est en général supposée de classe C^2. Les systèmes les plus naturels possédant une fonction de Lyapunov sont les systèmes de gradients, engendrés par des équations de la forme $\frac{dx_i}{dt} = -\frac{\delta E}{\delta x_i}$.

Ainsi, pour un réseau neuronal, les conditions suivantes sont connues pour être suffisantes à l'existence d'un point fixe :

— La matrice des poids W est triangulaire avec une diagonale nulle (le réseau est nécessairement non récurrent) ;
— La matrice des poids W est symétrique à diagonale positive ;
— $\sum_{i,j} w_{i,j} \leq \frac{1}{max_{i\in\mathbb{N}}|f_i'|}$; il est donc judicieux de choisir une fonction d'activation bornée, comme dans le cas de la sigmoïde exponentielle.

6.3.6 Erreur et règle d'apprentissage

Une **règle d'apprentissage**, ou algorithme d'apprentissage, est une règle permettant d'adapter un réseau à l'application concernée en modifiant la matrice des poids W. Nous nous plaçons ici dans le cadre de l'apprentissage supervisé, avec un réseau comportant N neurones en entrée et m en sortie.

On suppose disposer d'un vecteur d'entrée $x = (x_1, \cdots, x_k)$, d'un vecteur de sortie $s = (s_1, \cdots, s_m)$ et d'un vecteur de sortie supposée $d = (d_1, \cdots, d_m)$ auquel on va comparer le vecteur de sortie réellement obtenu s grâce à une fonction d'erreur E. Nous utilisons l'erreur quadratique :

$$E(s, d) = \frac{1}{2}\sum_{i=1}^{m}(s_i - d_i)^2$$

L'apprentissage consiste en l'initialisation de la matrice des poids $W = W(0)$ et en l'exécution de plusieurs cycles d'apprentissage jusqu'à ce que l'erreur $E(s, d)$ atteigne un minimum.
Soit $W = W(t)$ la matrice des poids à l'instant t. Un cycle (ou époque) d'apprentissage consiste à :

1. présenter le vecteur d'entrée x au réseau et calculer s en propageant l'activation ;
2. calculer $E(s, d)$;
3. déduire $W(t + 1)$.

La formule établissant $W(t + 1)$ en fonction de W et E est la **règle d'apprentissage** des poids du réseau. En considérant que E est une fonction de Lyapunov, et en réutilisant l'équation de système de gradient ci-dessous :

$$\frac{\delta w}{\delta t} = -\frac{\delta E}{\delta w}$$

on obtient la formule de correction des poids :

$$\Delta W := W(t + 1) - W(t) = -\lambda\frac{\delta E}{\delta W}$$

Le paramètre λ est le **pas d'apprentissage**. Il permet d'adapter une équation d'un système dynamique à temps continu à un cas à temps discret.

Règles d'apprentissage classiques Il existe plusieurs règles d'apprentissage classiques :

Principe de Hebb L'idée est la suivante : si deux neurones connectés entre eux sont activés de la même manière au même moment, la connexion qui les relie doit être renforcée. Sinon elle doit être inchangée ou affaiblie. On utilise pour cela la corrélation entre a_i et a_j, les activations respectives des neurones i et j :

$$w_{i,j} = \lambda corr(a_i, a_j)$$

QuickProp Dûe à Fahlman (1988) [14], cette règle repose sur l'idée d'augmenter le pas d'apprentissage lorsque les dérivées de la fonction d'erreur E gardent le même signe entre deux cycles d'apprentissage :

$$\Delta w_{i,j}^t = \frac{S_{i,j}^t}{S_{i,j}^{t-1} - S_{i,j}^t} \Delta w_{i,j}^{t-1}$$

avec $S_{i,j}^t = \dfrac{\delta E^t}{\delta w_{i,j}^t}$.

Nous allons maintenant étudier quelques modèles classiques pour l'apprentissage supervisé.

6.4 Modèles classiques pour l'apprentissage supervisé

6.4.1 Le perceptron

Développé par F. Rosenblatt en 1958 [41], dans le but de modéliser le mécanisme élémentaire du cerveau humain, ce réseau se compose de deux couches : une couche d'entrée (neurones 1 à k) et une couche de sortie (neurones $k-1$ à N. La matrice des poids est donc de la forme :

$$W = \begin{pmatrix} 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ w_{k+1,1} & \cdots & w_{k+1,k} & 0 & 0 & \cdots & 0 \\ w_{k+2,1} & \cdots & w_{k+2,k} & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{N,1} & \cdots & w_{N,k} & 0 & 0 & \cdots & 0 \end{pmatrix}$$

La fonction d'activation est la fonction identité, tandis que la fonction d'entrée des neurones de la couche de sortie est donnée par les $N-k$ dernières lignes du vecteur $(W.a)$.

Le perceptron est utilisé pour la classification. De fait, il permet de séparer linéairement l'espace à l'aide d'hyperplans. Toutefois, de nombreux problèmes ne sont pas linéairement séparables, c'est à dire séparables à l'aide d'hyperplans, ce qui a notamment motivé l'amélioration de ce type de réseaux, en ajoutant une couche cachée dotée d'une fonction d'activation non linéaire.

6.4.2 Le Perceptron Multi-Couches (PMC)

Le Perceptron Multi-Couches (PMC, ou encore MLP pour *Multi-Layer Perceptron*) comporte une couche d'entrée, une couche cachée (ou plus) avec une fonction d'activation **non linéaire** et une couche de sortie. Ce type de réseau est décrit comme un "approximateur universel de fonction", dans le sens où il peut approcher toute fonction continue à support borné, grâce à cette fonction d'activation non-linéaire. Il est utilisé dans de très nombreuses applications, comme la reconnaissance de formes, la classification et la régression, le contrôle automatique de processus...

L'activation se propage de la couche d'entrée vers les couches cachées successives et enfin vers la couche de sortie.

La fonction d'entrée du neurone i est linéaire pour la couche cachée et la couche de sortie : $h(i) = \sum_{j=1}^{N} w_{i,j} a_j$, tandis que la fonction d'activation f_i est non-linéaire (en général sigmoïde) et dérivable. Usuellement, la fonction d'activation de la couche de sortie est la fonction identité.

Un des règles d'apprentissage possible est une méthode de propagation des corrections des poids, appelée **rétropropagation**. Pour la correction des poids lors de l'apprentissage, on commencera par corriger les poids des neurones de la couche de sortie, puis par ceux de la dernière couche cachée, et ainsi de suite jusqu'à corriger les poids des neurones de la couche d'entrée. La section 6.5 est dédiée à la description de cette méthode.

6.4.3 Le réseau Radial Basis Function (RBF)

Les réseaux RBF (*Radial Basis Function*) sont des réseaux non bouclés, utilisés dans le cadre d'un apprentissage supervisé et développés par Powell en 1985 et améliorés par Broomhead & Lowe en 1988 [9]. Ces réseaux sont composés de trois couches :

- une couche d'entrée formée de d neurones avec, comme pour le PMC, la fonction d'activation identité ;
- une unique couche cachée formée de m neurones RBF opérant avec des **fonctions d'activation radiales de base** (en général des fonctions gaussiennes) ;
- une couche de sortie linéaire, ou bien affine, formée de s neurones.

Definition 6.4.1. Une fonction ψ de $\mathbb{R}^d$ dans $\mathbb{R}$ est dite **radiale de base** (**radial basis function**) si elle est symétrique par rapport à un point de $\mathbb{R}^d$ pour la norme euclidienne, c'est à dire s'il existe $c \in \mathbb{R}^d$ tel que :

$$\forall (x, y) \in \mathbb{R}^d \times \mathbb{R}^d, ||x - c|| = ||y - c|| \Rightarrow \psi(x) = \psi(y)$$

Le point c est appelé le *centre* ou le *noyau* de ψ.

Ce type de réseaux est connu pour être plus rapide et plus efficace que le PMC dans certaines situations, notamment pour des analyses en faible dimension. Il présente aussi l'avantage, par rapport au PMC, d'avoir des paramètres plus aisément ajustables et moins de risque de convergence vers un minimum local de la fonction d'erreur.
Toutefois, les réseaux RBF perdent de leur efficacité lorsque les données vivent dans des espaces

en grande dimension ou sur des données très bruitées. Ils ont également de moins bonnes capacités de généralisation que les PMC.

6.5 Rétro-propagation de l'erreur

Considérons un réseau de neurones PMC. Nous allons tout d'abord nous intéresser à l'erreur des neurones de la couche de sortie, que nous noterons S.

Erreur sur la couche de sortie En reprenant les notations précédentes, nous avons l'erreur quadratique :

$$E = \frac{1}{2}\sum_{i \in S}(a_i - d_i)^2$$

Comme E est une somme de carrés, son minimum sera unique. La correction des poids $w_{i,j}$ est donnée par l'équation :

$$\Delta w_{i,j} = -\lambda \frac{\delta E}{\delta w_{i,j}}$$

avec λ le poids d'apprentissage. Nous cherchons à calculer $\frac{\delta E}{\delta w_{i,j}}$ avec $i \in S$ un neurone de sortie. Nous avons :

$$\frac{\delta E}{\delta w_{i,j}} = \frac{\delta E}{\delta h(i)}.\frac{\delta h(i)}{\delta w_{i,j}}$$

Ainsi, avec $a_i = f(h(i))$, nous obtenons pour la première fraction :

$$\frac{\delta E}{\delta h(i)} = \frac{1}{2}\frac{\delta (a_i - d_i)^2}{\delta h(i)} = \frac{1}{2}\frac{\delta (a_i - d_i)^2}{d\delta a_i}\frac{df(h(i))}{dh(i)}$$

On obtient donc :

$$\frac{\delta E}{\delta h(i)} = (a_i - d_i).f'(h(i))$$

Pour la deuxième fraction, comme $h(i) = \sum_k w_{i,k}a_k$, alors :

$$\frac{\delta h(i)}{\delta w_{i,j}} = a_j$$

De sorte que, pour chaque neurone de la couche de sortie $i \in S$, nous avons la règle d'apprentissage dite *delta généralisée* :

$$\Delta w_{i,j} = \lambda.a_j.(d_i - a_i).f'(h(i))$$

Nous devons donc maintenant calculer les dérivées $\frac{\delta E}{\delta h(i)}$ de l'erreur sur les neurones i de la couche cachée C.

Erreur sur la couche cachée Considérons que nous avons une seule couche cachée C. Soit $i \in C$ un neurone de la couche cachée. Nous avons :

$$\frac{\delta E}{\delta h(i)} = \frac{1}{2} \sum_{k \in S} \frac{\delta (a_k - d_k)^2}{\delta h(i)} = \sum_{k \in S} \frac{\delta E}{\delta h(k)} . \frac{\delta h(k)}{\delta a_i} . \frac{df(h(i))}{dh(i)}$$

Comme nous avons $\frac{\delta E}{\delta h(k)} = (a_k - d_k) f'(h(k))$ pour tout $k \in S$, et que de plus $h(k) = \sum_{j \in C} w_{k,j} a_j$ nous obtenons donc :

$$\frac{\delta E}{\delta h(i)} = f'(h(i)) \sum_{k \in S} (a_k - d_k) f'(h(k)) w_{k,i}$$

Ainsi, pour tout neurone i de la couche cachée C, nous avons :

$$\frac{\delta E}{\delta w_{i,j}} = \frac{\delta E}{\delta h(i)} . \frac{\delta h(i)}{\delta w_{i,j}} = a_j f'(h(i)) \sum_{k \in S} w_{k,i} (a_k - d_k) f'(h(k))$$

Par conséquent :

$$\Delta w_{i,j} = \lambda . a_j . f'(h(i)) \sum_{k \in S} w_{k,i} (d_k - a_k) . f'(h(k))$$

Cette démonstration peut être généralisée à plusieurs couches cachées. Pour tout couple (i, j), où le neurone j appartient à la couche précédant de celle à laquelle appartient le neurone i :

$$\Delta w_{i,j} = \lambda . a_j . err_i$$

où :

- $err_i = (d_i - a_i) f'(h(i))$ si i est un neurone de sortie ;
- $err_i = f'(h(i)) \sum_k w_{k,i} err_k$ si i est un neurone d'une couche cachée et k parcourt les neurones de la couche suivant celle du neurone i.

L'erreur de corrélation croisée La fonction d'erreur la plus utilisée est la fonction d'erreur quadratique. Elle possède toutefois un inconvénient majeur : sa dérivée s'annule en plusieurs autres valeurs de a_i que celle désirée. Par exemple, si nous considérons la fonction sigmoïde exponentielle comme fonction d'activation : $f(x) = \frac{1}{1 + e^{-x}}$, et nous avons :

$$\Delta w_{i,j} = \lambda . a_j . (d_i - a_i) f'(h(i))$$

pour un neurone i de la couche de sortie S.
Nous avons $f'(x) = f(x)(1 - f(x))$, nous obtenons donc :

$$\Delta w_{i,j} = \lambda . a_j . (d_i - a_i) . a_i . (1 - a_i)$$

qui s'annule donc en trois valeurs de a_i : 0, 1 et d_i.
C'est pour cette raison qu'en pratique, la fonction de corrélation croisée l'emporte souvent sur l'erreur quadratique.
En effet, la fonction de corrélation croisée sur la sortie du réseau est définie comme :

$$E = - \sum_{i \in S} (d_i ln(a_i) + (1 - d_i) ln(1 - a_i))$$

Cette fonction a pour dérivée partielle par rapport à a_i :

$$\frac{\delta E}{\delta a_i} = \frac{d_i - a_i}{a_i(a_i - 1)}$$

puisque :

$$\frac{\delta E}{\delta a_i} = -\left(\frac{d_i}{a_i} - \frac{1 - d_i}{1 - a_i}\right) = \frac{d_i(1 - a_i) - (1 - d_i a_i)}{a_i(a_i - 1)}$$

Nous allons maintenant étudier certaines propriétés des réseaux de neurones.

Le momentum Une version améliorée de l'algorithme de rétropropagatin de l'erreur introduit un terme dit de "momentum". Ce terme permet d'incorporer une part de l'ancienne variation de poids pour le calcul de la nouvelle variation de poids.
Ceci permet notamment d'éviter un problème d'oscillation commun avec l'algorithme de rétropropagation de l'erreur classique, notamment quand la surface d'erreur à une zone minimum très réduite.

Le momentum correspond donc à un terme ajouté au calcul de la correction des poids, égale à $\alpha * \Delta w_{i,j}^{t-1}$, où α est une constante représentant l'influence du momentum, et $\Delta w_{i,j}^{t-1}$ la correction des poids à l'instant $t - 1$.

Règle d'apprentissage Rprop Rprop signifie *Resilient backpropagation*, et est une règle locale d'apprentissage pour les réseaux de neurones PMC. Le principe de base de Rprop est d'utiliser seulement le signe de la dérivée de l'erreur pour calculer le *sens* de la correction de poids, tandis que la *taille* de cette correction sera déterminée exclusivement par un *facteur d'actualisation* $\Delta_{i,j}$:

$$\Delta w_{i,j} = -sign(\frac{\delta E}{\delta w_{i,j}})\Delta_{i,j}$$

La valeur du facteur d'actualisation est calculé, au temps t, par :

$$\Delta_{i,j}^t = \begin{cases} \eta^+ * \Delta_{i,j}^{(t-1)}, & \text{si } \frac{\delta E^{(t-1)}}{\delta w_{i,j}} * \frac{\delta E^{(t)}}{\delta w_{i,j}} > 0 \\ \eta^- * \Delta_{i,j}^{(t-1)}, & \text{si } \frac{\delta E^{(t-1)}}{\delta w_{i,j}} * \frac{\delta E^{(t)}}{\delta w_{i,j}} < 0 \\ \Delta_{i,j}^{(t-1)}, & \text{sinon} \end{cases}$$

avec $0 \leq \eta^- \leq 1 \leq \eta^+$.

En d'autres termes, quand la dérivée partielle du poids $w_{i,j}$ correspondant change de signe, ce qui indique que la dernière correction de poids était trop grande et l'algorithme a sauté" par dessus un minimum local, la valeur d'actualisation décroit d'un facteur η^-. Si la dérivée garde le même signe, la valeur d'actualisation est augmentée d'un certain facteur pour accélérer la convergence.

Méthodes du second ordre et algorithme d'apprentissage SCG L'algorithme de rétro-propagation du gradient est une méthode du premier ordre, dans le sens ou il n'utilise que la dérivée première de l'erreur. Les méthodes du second ordre utilisent elles aussi les dérivées du second ordre. De façon général, une méthode du second ordre toruve généralement une meilleure façon d'arriver à un minimum local qu'une méthode du premier ordre, mais avec un coût en calcul plus élevé.

L'algorithme d'apprentissage SCG, pour *Scaled Conjugate Gradient*, a été développé par Moller en 1993 [34]. L'algorithme SCG fait partie des techniques dites de "gradient conjugué". Moller indique notamment que l'approximation quadratique de l'erreur E dans un voisinage d'un point w, noté qw, est donnée par :

$$E_{qw}(y) = E(w) + E'(w)^T y + \frac{1}{2} y^T E"(w)$$

Minimiser l'erreur E revient à trouver les points critiques solutions du système linéaire définit par Moller :

$$E_{qw}(y) = E"(w)y + E'(w) = 0$$

L'algorithme SCG étant assez complexe, nous redirigeons le lecteur curieux vers la très intéressante publication de Moller *A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning*[34].

6.6 Propriétés et intérêts des réseaux de neurones

Les réseaux de neurones présentent plusieurs propriétés très intéressantes. Les deux plus importantes sont :

Théorème 1. ***Théorème d'approximation universelle***
Toute fonction bornée suffisamment régulière peut être approchée uniformément, avec une précision arbitraire, dans un domaine fini de l'espace de ses variables, par un réseau de neurones avec un nombre fini de neurones et une couche cachée, possédant tous la même fonction d'activation, et un neurone de sortie linéaire.

Théorème 2. ***Théorème d'approximation parcimonieuse***
Si le résultat de l'approximation est une fonction non linéaire des paramètres ajustables, elle est plus parcimonieuse qu'une approximation linéaire de ses paramètres. De plus, pour des réseaux de neurones à fonction d'activation sigmoïdale, l'erreur commise dans l'approximation est indépendante du nombre de variables de la fonction à approcher. Ainsi, pour un nombre de neurones fixé, le nombre de paramètres est proportionnel au nombre de variables de la fonction à estimer.

Le premier théorème a été démontré par Cybenko (1989) [10], Funahashi [18] (1989) et Hornik (1989) [26]. Ce théorème n'est qu'un théorème d'existence et ne donne pas d'indications sur la façon d'approcher la fonction recherchée, notamment sur le nombre de neurones à choisir dans la couche cachée.

Le deuxième théorème, quant à lui, implique qu'il est possible d'approximer correctement davantage de fonctions avec des réseaux de neurones qu'avec des polynômes. L'intérêt est de limiter le nombre d'exemples nécessaires à l'approximation. Ainsi, cette propriété procure aux réseaux de neurones une certaine flexibilité, une bonne capacité de généralisation et une résistance au bruit.
La terme **parcimonie** fait ici référence fait ici référence à la capacité des réseaux de neurones à pouvoir estimer une fonction linéaire avec un nombre restreint de paramètres.

6.7 Conclusion sur la théorie des réseaux de neurones

Les réseaux de neurones, bien qu'étant des outils intéressants de par leur parcimonie et leur approximation universelle, présentent aussi des défauts. Toutefois ces défauts ne sont pas propres aux réseaux de neurones mais communs aux techniques d'apprentissage basées sur une approche supervisée non-linéaire.
Ce sont principalement le risque de sur-apprentissage, le risque de convergence vers un minimum local et l'aspect "boîte noire", ce dernier étant principalement dû à une mauvaise connaissance de ces modèles.

Le sur-apprentissage se produit lorsqu'un modèle s'imprègne trop des caractéristiques de l'échantillon test et perd de son potentiel de généralisation. Dans le cas des réseaux de neurones, ce risque est accru avec le nombres de neurones et de couches cachées et est réduit notamment grâce à la validation croisée, ainsi que par l'agrégation de modèles.

Nous allons par la suite étudier l'application des réseaux de neurones à notre modèle pour le provisionnement individuel.

6.8 Réseaux de neurones - Choix des paramètres optimaux

Nous présentons ici les résultats obtenus pour les réseaux de neurones sur le triangle inclus dans le triangle échantillon.
Nous utilisons le package R `RSNNS`.

Pour chacun de nos triangles de dépenses et de charges, nous optimiserons les paramètres suivants :
- le nombre de neurones par couches, en fixant le nombre de couches cachées à deux,
- l'algorithme d'apprentissage et la fonction d'activation,
- la nécessité ou non d'ajouter une troisième couche cachée,
- le taux d'apprentissage et le momentum.

Le premier paramètre dépendra de l'année de développement, et variera donc en fonction du temps. Les autres paramètres seront fixes sur tous les développements.

Nous allons commencer par regarder les résultats du choix de variables par l'algorithme RFE.

6.8.1 Importance des variables et choix des variables explicatives par la méthode RFE

Nous rappelons que nous utilisons l'algorithme RFE dans le but de retenir les variables les plus pertinentes, pour la prédiction du deuxième développement sur le triangle initial.
Nous rappelons aussi que nous cherchons à prédire les **incréments** de dépenses et les **variations** de charges, que la variable `dev_N` correspond à l'incrément de dépenses au développement d'année N, tandis que la variable `ctb_N` correspond à la variation de charge estimées par les gestionnaires au développement d'année N.
Ces variables ont des versions centrées réduites respectives `dev.n_N` et `ctb.n_N`.

Comme pour les forêts aléatoires, nous calculerons ici l'importance de nos variables pour pouvoir utiliser l'algorithme RFE.
Pour les réseaux de neurones, nous utiliserons la méthode proposée par Garson [19].
L'idée derrière la méthode de Garson est que les poids présents dans un réseau de neurones représentent une quantité d'information considérée par le réseau. Par exemple, les variables explicatives qui ne sont pas jugées discriminantes pour la prédiction d'une variable cibles se verront associées des poids faibles, tandis que des variables fortement discriminantes se verront associées des poids élevés : les poids représentent directement l'importance des variables explicatives pour la prédiction de la variable cible.
Garson identifie l'importance relative des variables explicatives dans la prédiction d'une unique variable cible, dans un réseau à une seule couche cachée, en "déconstruisant" les poids : l'importance relative est directement identifiable par les poids liant le neurone de la couche d'entrée représentant la variable explicative considérée et le neurone de sortie représentant la variable cible.

Si nous notons Q_i l'importance relative associée au i-ème neurone de la couche d'entrée, w_{ij} le poids entre le i-ème neurone de la couche d'entrée et le j-ème neurone de la couche cachée, et v_j le poids entre le j-ème neurone de la couche cachée et le neurone de la couche de sortie, nous obtenons :

$$Q_i = \frac{\sum_{j=1}^{L} |w_{ij}|}{\sum_{r=1}^{N} |w_{r,j}|} * \frac{1}{\sum_{i=1}^{N} \sum_{j=1}^{L} (|w_i v_j| / \sum_{r=1}^{N} |w_{rj}|)}$$

Un point d'attention dans notre cas : nous avons encodé nos variables catégorielles en version *dummy*, comme expliqué dans le premier chapitre. Pour notre réseau de neurones, une variable catégorielle correspondra donc à $n-1$ neurones en couche d'entrée, n étant le nombre de modalité de la variable.
Pour l'application de l'algorithme RFE, nous ne souhaitons pas enlever seulement quelques modalités d'une variable mais bien la variable elle-même, nous modifions donc l'algorithme de Garson pour sommer les importances relatives pour tous les neurones correspondants aux modalités d'une variable, afin d'obtenir l'importance relative globale de cette variable.

Après des tests préliminaires pour nous assurer de la convergence de l'algorithme, nous utilisons le réseau de neurones suivant :

— Une seule couche cachée, avec 5 neurones
— Nombre d'itérations maximales : 500
— Fonction d'activation : Sigmoïde
— Algorithme d'apprentissage : Rétropropagation du gradient de l'erreur

Les autres paramètres seront maintenus par défaut.

Nous obtenons le résultat suivant pour les dépenses :

Variable	Importance relative
Grave	100,00
devN_1	25,98
IPP01_50_surv	18,63
IPP91_100_surv	13,09
IPP71_80_surv	7,25
clos_surv	4,95
Nbvic_surv	4,51
Trauma_surv	3,56
IPP61_70_surv	3,39
IPP51_60_surv	2,78
IPP81_90_surv	1,97
Tetra_surv	1,76
Part_Respons_Conv	1,60
Part_Respons	1,54
Convention	1,33
Code_Clause	0,99

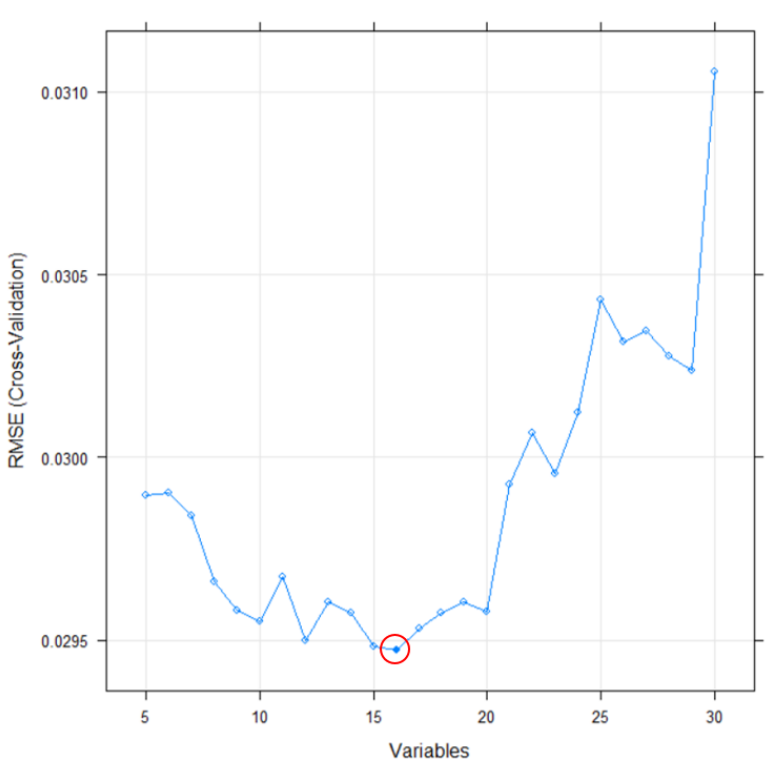


FIGURE 6.4 – Résultat du RFE pour les dépenses, pour les réseaux de neurones. A gauche, l'importance des variables calculée selon l'algorithme de Garson.

Par l'intermédiaire de l'algorithme RFE, nous retenons 16 variables, pour un RMSE d'apprentissage inférieur à 0.0295. Les variables les plus importantes pour les réseaux de neurones sont la variable `grave` et les variables `IPP` indiquant la présence de victimes présentant un taux d'IPP élevé.

Et pour les charges :

Variable	Importance relative
Grave	100,00
ctbN_1	36,77
IPP91_100_surv	28,03
Trauma_surv	8,57
Tetra_surv	5,18
IPP71_80_surv	3,22
IPP51_60_surv	1,92
IPP81_90_surv	1,65
Para_surv	1,48

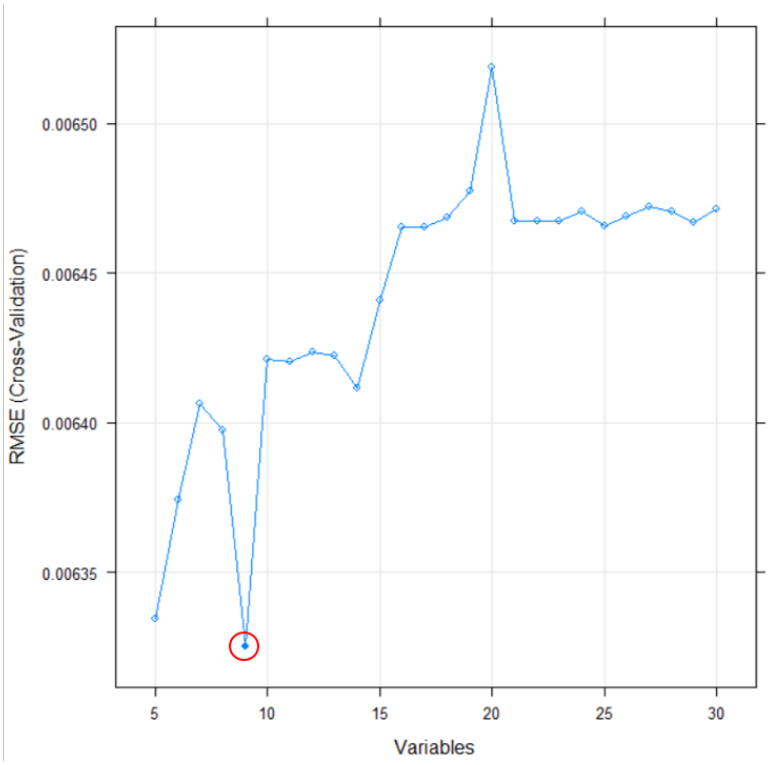


FIGURE 6.5 – Résultat du RFE pour les charges, pour les réseaux de neurones. A gauche, l'importance des variables calculée selon l'algorithme de Garson.

Comme pour les forêts aléatoires, l'algorithme RFE retient ici moins de variables pour les charges que pour les dépenses (9 variables pour un RMSE d'apprentissage de 0.00632). Nous retrouvons encore une fois la variable `grave` ainsi que les variables `IPP`.

Nous allons maintenant nous intéresser au paramètre dynamique en fonction de l'année de développement : le nombre de neurones.

6.8.2 Nombre de neurones et couches cachées

Nous commençons tout d'abord par tester le nombre de neurones par couche cachée. Nous allons tester un réseau de neurones ayant jusqu'à deux couches cachées, à raison d'un à cinq neurones dans la première couche et d'un à trois neurones dans la seconde.

Nous présentons les résultats obtenus avec un gradient de couleur, comme pour les forêts aléatoires :

Dépenses

Nombre de neurones dans la 1ère couche cachée	Nombre de neurones dans la 2ème couche cachée	Développement					
		2	3	4	5	6	7
1	1	0,05168	0,01478	0,01194	0,00519	0,00553	0,02023
1	2	0,04929	0,01407	0,01138	0,00523	0,00617	0,01901
1	3	0,04898	0,01407	0,01126	0,00519	0,00560	0,01936
2	1	0,04870	0,01457	0,01211	0,00525	0,00553	0,01996
2	2	0,04784	0,01352	0,01116	0,00520	0,00569	0,01895
2	3	0,04777	0,01398	0,01205	0,00520	0,00626	0,01834
3	1	0,04829	0,01334	0,01171	**0,00513**	0,00559	0,01796
3	2	0,04570	0,01330	0,01143	0,00548	0,00555	0,01705
3	3	0,04600	0,01343	0,01148	0,00606	0,00555	**0,01703**
4	1	0,04193	0,01336	0,01146	0,00519	0,00558	0,01929
4	2	0,04695	0,01324	0,01170	0,00519	**0,00551**	0,01785
4	3	0,04199	0,01327	0,01135	0,00527	0,00555	0,01762
5	1	**0,04179**	0,01330	0,01155	0,00521	0,00571	0,01967
5	2	0,04374	**0,01244**	**0,01105**	0,00521	0,00574	0,01779
5	3	0,04461	0,01296	0,01109	0,00526	0,00575	0,01801
Architecture retenue		(5,1)	(5,2)	(5,2)	(3,1)	(4,2)	(3,3)

FIGURE 6.6 – Architecture retenue pour les dépenses, pour les réseaux de neurones.

Charges

Nombre de neurones dans la 1ère couche cachée	Nombre de neurones dans la 2ème couche cachée	Développement					
		2	3	4	5	6	7
1	1	0,00545	0,00660	0,00523	0,00376	0,00471	0,00139
1	2	0,00547	0,00660	0,00589	0,00375	0,00481	0,00139
1	3	0,00549	0,00662	0,00519	0,00378	0,00475	0,00139
2	1	0,00545	0,00662	0,00514	0,00375	0,00472	0,00141
2	2	0,00545	0,00663	0,00512	0,00376	0,00475	0,00262
2	3	0,00552	0,00717	0,00514	0,00378	0,00472	0,00139
3	1	**0,00544**	0,00663	0,00511	0,00376	0,00478	0,00139
3	2	0,00545	**0,00660**	0,00510	0,00375	**0,00470**	0,00140
3	3	0,00549	0,00690	**0,00510**	0,00410	0,00473	**0,00138**
4	1	0,00545	0,00660	0,00510	**0,00374**	0,00474	0,00141
4	2	0,00545	0,00719	0,00511	0,00378	0,00475	0,00141
4	3	0,00545	0,00661	0,00511	0,00375	0,00473	0,00140
5	1	0,00548	0,00660	0,00511	0,00375	0,00475	0,00141
5	2	0,00546	0,00675	0,00511	0,00375	0,00474	0,00142
5	3	0,00545	0,00662	0,00510	0,00375	0,00478	0,00140
Architecture retenue		(3,1)	(3,2)	(3,3)	(4,1)	(3,2)	(3,3)

FIGURE 6.7 – Architecture retenue pour les dépenses, pour les réseaux de neurones.

Nous remarquons tout d'abord que pour les dépenses, le nombre de neurones dans la première couche tend à décroître avec les années de développement (de 5 vers 3), tandis que le nombre de neurones dans la seconde couche à plutôt tendance à croître (de 1 vers 3).

Pour les charges, le nombre de neurones dans la première couche reste relativement constant (autour de 3), mais le nombre de neurones dans la seconde couche semble évoluer de façon cyclique, répétant le cycle $(1, 2, 3)$.

Une fois ces paramètres fixés, nous allons maintenant nous intéresser à l'algorithme d'apprentissage utilisé et à la fonction d'activation.

6.8.3 Algorithme d'apprentissage et fonction d'activation

Nous réutilisons les paramètres précédents mais nen testant simultanément deux paramètres :
— La fonction d'activation. Nous testerons les fonctions suivantes : Sigmoïde (para mètre `Logistic`), Tangente hyperoblique (`TanH`), Linéaire (`Linear`) et exponentielle (`Exponential`)
— L'algorithme d'apprentissage. Nous testerons les algorithmes d'apprentissage suivants : Rétropropagation du gradient de l'erreur (`Std Backpropagation`), Rprop et *Scaled Conjugate Gradient*.

Nous obtenons les résultats suivants :

Dépenses

Apprentissage\Activation	Logistic	TanH	Linear	Exponential
Std_Backpropagation	26 397	**24 295**	51 375	24 619
Rprop	35 368	24 317	35 116	30 173
SCG	25 699	40 338	140 829	25 678

FIGURE 6.8 – Résultats pour les algorithmes d'apprentissage et fonctions d'activation testés, pour les dépenses.

Charges

Apprentissage\Activation	Logistic	TanH	Linear	Exponential
Std_Backpropagation	60 929	**60 617**	161 984	60 780
Rprop	79 587	92 468	157 740	76 287
SCG	72 500	2 067 181	2 321 792	65 645

FIGURE 6.9 – Résultats pour les algorithmes d'apprentissage et fonctions d'activation testés, pour les charges.

Nous remarquons que les erreurs de prédiction les plus faibles sont relevés sur les résultats de la fonction d'activation tangente hyperbolique, associée à l'algorithme d'apprentissage de la rétropropagation de l'erreur classique.
Les erreurs de prédiction importantes pour les fonctions d'activation linéaire et exponentielles indiquent un sous-apprentissage important, les algorithmes n'ayant pas convergés avec le nombre d'itérations maximum autorisé. Nous remarquons le même souci pour l'algorithme SCG. De plus en analysant les résultats de façon plus fine, nous voyons que l'algorithme SCG a toujours tendance à sous-estimer les dépenses et les charges.
Nous choisissons donc de conserver l'algorithme de rétropropagation de l'erreur et la fonction d'activation tangente hyperbolique.

Nous allons maintenant tester la sensibilité de nos réseaux de neurones actuels à la présence ou non d'une troisième couche cachée.

6.8.4 Présence d'une troisième couche cachée

Nous analysons maintenant les résultats de nos réseaux de neurones, en limitant nos réseaux à deux ou trois couches cachées. Nous permettons à nos réseaux d'avoir jusqu'à trois neurones dans la troisième couche cachée.
Dans la littérature, nous pouvons trouver que la présence d'une troisième couche cachée n'est généralement pas nécessaire, et peut au contraire entraîner des problèmes de sur-apprentissage.
Nous obtenons les résultats suivants :

Dépenses						
Nombre de couches cachées	RMSE					
2	24 295					
3	28 435					
	Développement					
	2	3	4	5	6	7
Neurones 3è couche	1	2	1	1	1	2

FIGURE 6.10 – Résultats avec une troisième couche cachée, pour les dépenses.

Charges						
Nombre de couches cachées	RMSE					
2	60 617					
3	62 684					
	Développement					
	2	3	4	5	6	7
Neurones 3è couche	2	3	2	2	1	2

FIGURE 6.11 – Résultats avec une troisième couche cachée, pour les charges.

La présence d'une troisième couche cachée semble en effet détériorer les erreurs de prédiction, aussi bien pour les dépenses que pour les charges. Nous nous limitons donc à deux couches cachées.

Nous allons enfin maintenant tester deux derniers paramètres simultanément : le taux d'apprentissage et le momentum

6.8.5 Taux d'apprentissage et momentum

Nous obtenons les résultats suivants, pour le RMSE du dernier développement :

Dépenses				
Tx d'apprentissage\Momentum	**Moment 0**	**Moment 0.1**	**Moment 0.25**	**Moment 0.5**
Taux d'apprentissage 0.1	**25 746**	26 837	29 821	33 941
Taux d'apprentissage 0.25	26 855	35 197	31 678	41 454
Taux d'apprentissage 0.5	33 118	42 741	32 355	30 851

FIGURE 6.12 – Résultats pour le taux d'apprentissage et le momentum, pour les dépenses.

Charges				
Tx d'apprentissage\Momentum	**Moment 0**	**Moment 0.1**	**Moment 0.25**	**Moment 0.5**
Taux d'apprentissage 0.1	58 571	60 964	63 370	61 465
Taux d'apprentissage 0.25	53 889	55 920	**46 030**	51 786
Taux d'apprentissage 0.5	90 581	77 366	81 613	74 931

FIGURE 6.13 – Résultats pour le taux d'apprentissage et le momentum, pour les charges.

Le taux d'apprentissage représente l'intensité à laquelle nos réseaux de neurones s'imprègnent des données fournies. Un taux d'apprentissage trop fort pourrait empêcher le réseau de converger, tandis qu'un taux d'apprentissage trop faible expose nos réseaux à un risque de sur-apprentissage. Le momentum, quant à lui, représente une fraction de l'ancienne variation de poids incorporée dans la nouvelle variation, qui peut empêcher des phénomènes d'oscillations autour d'un minimum (local).

Nous retenons ici un taux d'apprentissage légèrement supérieur pour les charges que pour les dépenses, à 0.25 contre 0.1 ; nous pouvons interpréter cette différence comme la présence d'une volatilité plus importante pour les charges que pour les dépenses, nos réseaux ayant besoin de s'imprégner plus des données.
De plus, nos réseaux de neurones n'ont, semble-t-il, pas d'utilisation pour le terme de momentum pour les dépenses. Ce n'est néanmoins pas le cas pour les charges, où nous obtenons un terme de momentum conservé de 0.25. ici aussi, nous pouvons interpréter cette différence comme une volatilité plus importante des charges que des dépenses.

Pour les prédictions des dépenses, nous observons que la qualité semble décroître en fonction de l'augmentation du taux d'apprentissage, pour un momentum faible. Ce premier semble avoir moins d'impact pour un momentum supérieur à 0.25
Pour les prédictions de charges, pour les valeurs testées, un taux d'apprentissage égal à 0,25 semble préférable à des valeurs supérieures ou inférieures, pour toutes les valeurs de momentum testées.

Nous retenons donc les paramètres fixes suivants pour nos réseaux de neurones :
— Pour les dépenses : deux couches cachées, l'algorithme de rétropropagation du gradient de l'erreur, la fonction d'activation tangente hyperbolique, un taux d'apprentissage égal à 0,1 et un momentum nul.

— Pour les charges : deux couches cachées, l'algorithme de rétropropagation du gradient de l'erreur, la fonction d'activation tangente hyperbolique, un taux d'apprentissage égal à 0,25 et un momentum égal à 0,25.

6.8.6 Vérification de la pertinence du choix de variables par la méthode RFE

Comme dans le cas des forêts aléatoires, nous allons ici vérifier la pertinence de l'utilisation de la méthode RFE pour le choix des variables explicatives.

En utilisant nos paramètres retenus, nous obtenons les RMSE suivants pour le dernier développement :

Dépenses

Choix de variables	RMSE du dernier développement
Avec RFE	25 854
Avec toutes les variables initiales	29 181

Charges

Choix de variables	RMSE du dernier développement
Avec RFE	64 160
Avec toutes les variables initiales	67 854

FIGURE 6.14 – Comparaison des erreurs de prédictions pour le dernier développement, entre le modèle utilisant toutes les variables initiales et le modèle utilisant les variables conservées par l'algorithme RFE.

Nous observons donc qu'en n'utilisant que les variables retenues par l'algorithme RFE à la place de toutes les variables, notre modèle commet moins d'erreurs.

Nous avons maintenant fixé tous les paramètres nécessaires à partir du triangle échantillon, aussi bien pour les forêts aléatoires que pour les réseaux de neurones. Nous allons étudier les résultats de l'application de nos algorithmes au triangle inclus dans sa globalité (alors que nous nous étions restreint à un échantillon), et comparer les résultats obtenus aux données réelles et aux résultats auxquels amène la méthode Chain-Ladder.

Troisième partie

Application du modèle de provisionnement individuel aux données complètes

Dans cette partie, nous allons analyser les résultats obtenus par notre modèle de provisionnement individuel.

Nous allons dans un premier temps comparer les résultats obtenus sur le triangle inclus, aussi bien entre les modèles qu'avec une méthode Chain-Ladder et aux données réelles.
Nous allons ensuite présenter les résultats obtenus sur le triangle dans sa globalité, sans restriction d'années de survenance, ainsi qu'un backtesting en enlevant la dernière diagonale connue.
Enfin, nous présenterons quelques cas possibles d'utilisations de ces prédictions individuelles, avant de conclure.

Chapitre 7

Résultats sur le triangle inclus et comparaisons aux données réelles

Dans ce chapitre, nous analysons les résultats obtenus par notre modèle de provisionnement individuel sur le triangle inclus.
Nous avons cumulé les prédictions pour pouvoir comparer plus facilement nos résultats à ceux obtenus par une méthode Chain-Ladder. Nous vérifions les hypothèses permettant d'appliquer la méthode Chain-Ladder en annexe.

Analyse au niveau individuel Dans un premier temps, nous regarderons les prédictions au niveau individuel. Nous analysons les erreurs de prédictions par année de survenance et de développement par l'intermédiaire du RMSE et du MAE, et la qualité de l'ajustement de la régression par l'intermédiaire du coefficient de détermination R^2.
Nous obtenons ainsi des triangles en vision agrégée. Cela nous permet d'obtenir une visualisation simple de la qualité individuelle des résultats, qui n'est pas sans rappeller le côté visuel des développements d'un Chain-Ladder. Un code couleur permet de montrer quel algorithme, pour une année de survenance et un développement donné, commet la plus grande ou plus petite erreur de prédiction, ou a le meilleur ou le plus mauvais ajustement. Cela nous permettra de comparer les modèles sur les prédictions pour tous les développements.
Nous regarderons aussi les graphiques des valeurs prédites en fonction des valeurs réelles, pour le dernier développement.

Pour un vecteur de valeurs réelles y et un vecteur de valeurs prédites $\hat{y}$ nous aurons les éléments suivants :

- RMSE : $RMSE(y, \hat{y}) = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}}$
- MAE : $MAE(y, \hat{y}) = \frac{\sum_{i=1}^{n}|\hat{y}_i - y_i|}{n}$
- R^2 : $R2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2}$, avec $\overline{y}$ la moyenne de y
- Graphiques prédictions contre valeurs réelles

Nous utilisons notamment le MAE et le RMSE, car ce dernier donne un poids plus important aux erreurs importante : nous pourrons notamment analyser les écarts entre MAE et RMSE pour

détecter les modèles ayant plus de mal à prédire les valeurs des sinistres présentant de fortes dépenses ou charges.

Analyse en vision agrégée Dans un second temps, nous regarderons les prédictions au niveau agrégé via l'erreur relative absolue, par année de survenance et de développement. Nous regarderons aussi l'erreur relative de prédiction globale par développement, l'erreur de réserve à 1 an et l'erreur de réserve à l'ultime.
La réserve à 1 an correspond à l'écart entre la diagonale suivant la dernière diagonale connue et cette dernière, tandis que la réserve à l'ultime est définie comme la valeur du dernier développement moins la valeur de la dernière diagonale connue. Nous aurons ainsi des erreurs relatives sur la réserve à 1 an et la réserve à l'ultime.

Nous aurons donc les éléments suivants :
— Erreur relative : calculée comme le rapport $\frac{\textit{valeur prédite} - \textit{valeur réelle}}{\textit{valeur réelle}}$, en vison agrégée par année de survenance et de développement. Nous souhaitons obtenir la plus petite erreur relative en valeur absolue.
— Erreur relative de prédiction globale : erreur relative agrégée par développement
— Réserve à 1 an
— Réserve à l'ultime

Notations Pour des raisons de simplifications, nous utiliserons les notations suivantes :
— FA : résultats obtenus par les forêts aléatoires,
— PMC : résultats obtenus par le bagging de réseaux de neurones,
— GLM : résultats obtenus par les GLM pénalisés,
— CL : résultats obtenus par méthode Chain-Ladder "naïve" (sans sélection de coefficients),
— ARE : *Absolute Relative Error*, ou erreur relative absolue. Elle est calculé comme le rapport $\left|\frac{\textit{valeur prédite} - \textit{valeur réelle}}{\textit{valeur réelle}}\right|$.
— "Surv" pour l'année de survenance.
— "Dev N" dans certains graphiques pour "Développement d'année N".

Nous allons tout d'abord analyser les résultats obtenus sur les dépenses.

7.1 Résultats de prédiction des dépenses

7.1.1 Erreurs individuelles

RMSE

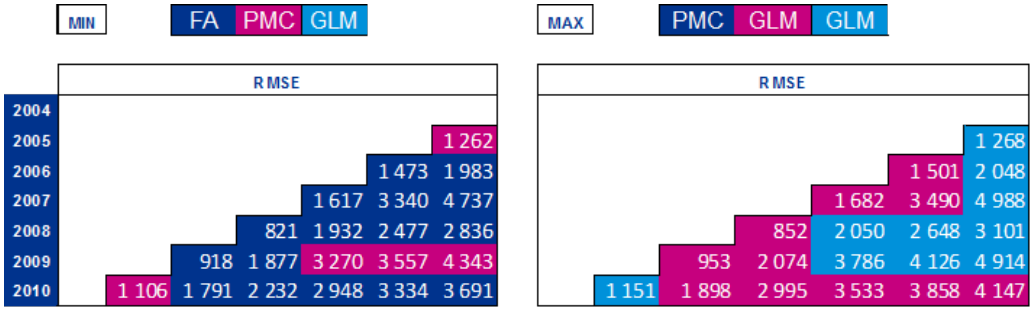


FIGURE 7.1 – Erreurs individuelles de prédiction : RMSE.

Nous remarquons tout d'abord que les résultats obtenus par les forêts aléatoires semblent les plus précis, d'un point de vue du RMSE (sur-représentation du modèle FA dans le triangle "min"). Nous notons par exemple que sur l'année de survenance 2010, le modèle FA obtient un RMSE de 3 691 pour le dernier développement, contre 4 147 pour le modèle GLM.
Le modèle qui commet le plus d'erreurs sur la première diagonale prédite est le modèle PMC. C'est aussi lui qui commet le plus d'erreurs pour la dernière année de survenance.

En regardant de plus près l'évolution du meilleur RMSE au fur et à mesure des années de survenances et de développement, nous remarquons des erreurs plus importantes pour les développements éloignés des survenances récentes.
Nous retrouverons cette analyse pour tous nos résultats : plus on avance dans les années de survenances et de développements, plus on avance dans l'incertain.

MAE

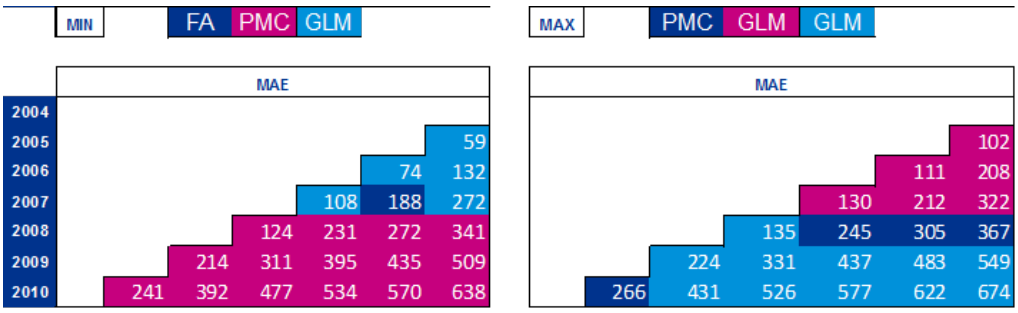


FIGURE 7.2 – Erreurs individuelles de prédiction : MAE.

Cette fois, le modèle qui semble commettre le moins d'erreurs individuelles est celui obtenu par bagging de réseaux de neurones, sauf sur les années de survenances 2005 à 2007 qui semblent favoriser les GLM pénalisés, pour des MAE allant de 59 à 272 contre 102 à 322 pour le modèle PMC. Toutefois, le modèle GLM commet les plus grandes erreurs sur les années de survenances 2008 à 2010, pour des MAE respectifs de 549 et 674, contre 509 et 638 pour le modèle PMC.
Notamment, nous pouvons interpréter la différence de résultats obtenus en comprenant le RMSE et le MAE de la façon suivante : le modèle basé sur les réseaux de neurones commet en moyenne moins d'erreurs que ceux des autres modèles. Mais les erreurs commises sont plus significatives sur les sinistres ayant des dépenses élevées.
Enfin, nous remarquons ici aussi une détérioration de la qualité de prédiction au fur et à mesure des années de survenance et de développements.

R^2

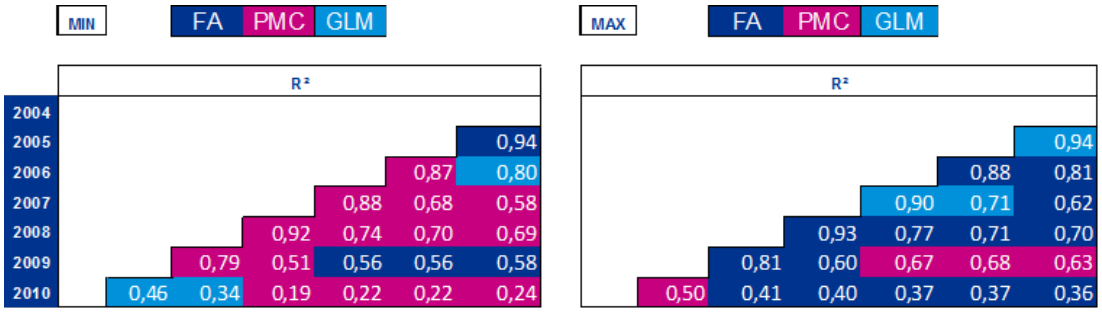


FIGURE 7.3 – Qualité d'ajustement de la régression : R2.

Le modèle présentant la meilleure qualité d'ajustement (le R^2 maximum) semble ici être celui obtenu par les forêts aléatoires. Notamment pour la dernière année de survenance sur le dernier développement, le modèle FA obtient un R^2 égal à 0,36, et le modèle PMC seulement 0,24.
Les prédictions sont d'assez bonnes qualités pour la première diagonale prédite pour les années de développements 2005 à 2009, et se détériorent assez rapidement au fil des développements, notamment pour les années de survenance 2009 et 2010.

Pour toutes nos erreurs individuelles, nous remarquons notamment qu'aucun modèle de l'emporte uniformément sur toutes les années de survenance, pour tous les développements. Une idée intéressante serait de pouvoir agréger nos différents modèles entre eux, afin d'obtenir un meilleur modèle sur toutes les années de survenances, pour tous les développements.

Nous regardons maintenant le graphiques des prédictions individuelles en fonction des valeurs réelles, pour le dernier développement :

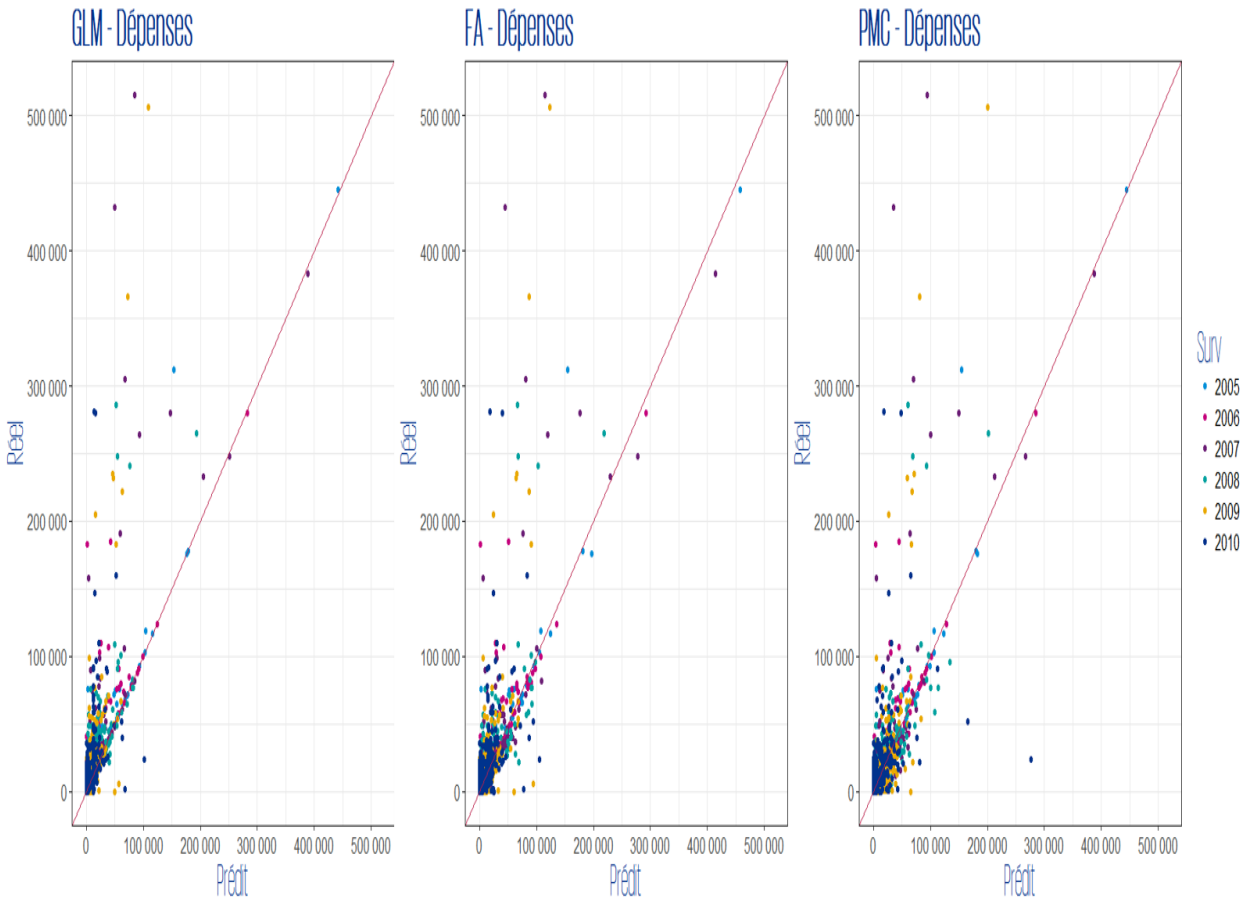


FIGURE 7.4 – Valeurs prédites contre valeurs réelles - Dépenses.

Un ajustement parfait entraînerait que tous les points ici présentés seraient sur la première bissectrice, d'équation $y = x$, représentée ici en rouge.
Un premier point intéressant est que les trois modèles semblent relativement proches entre eux. La plupart des points loin de la droite $y = x$ le sont pour tous les modèles. Nous pouvons toutefois voir le peu de points sous la droite $y = x$ pour le modèle issu des GLM pénalisés, ce qui indique une sous-évaluation des dépenses, tandis que le modèle basé sur le bagging de réseaux de neurones présente une certaine sur-évaluation pour certains sinistres.

Nous remarquons aussi que les modèles semblent commettre moins d'erreurs sur les survenances plus anciennes, que sur les survenances plus récentes. Cela peut s'expliquer simplement par le fait que sur les survenances plus anciennes, le développement des sinistres est plus important. Il y a donc moins d'incertitude sur la projection de ces sinistres.
Si nous effectuons un grossissement sur les intervalles de valeurs $[0; 100000]$:

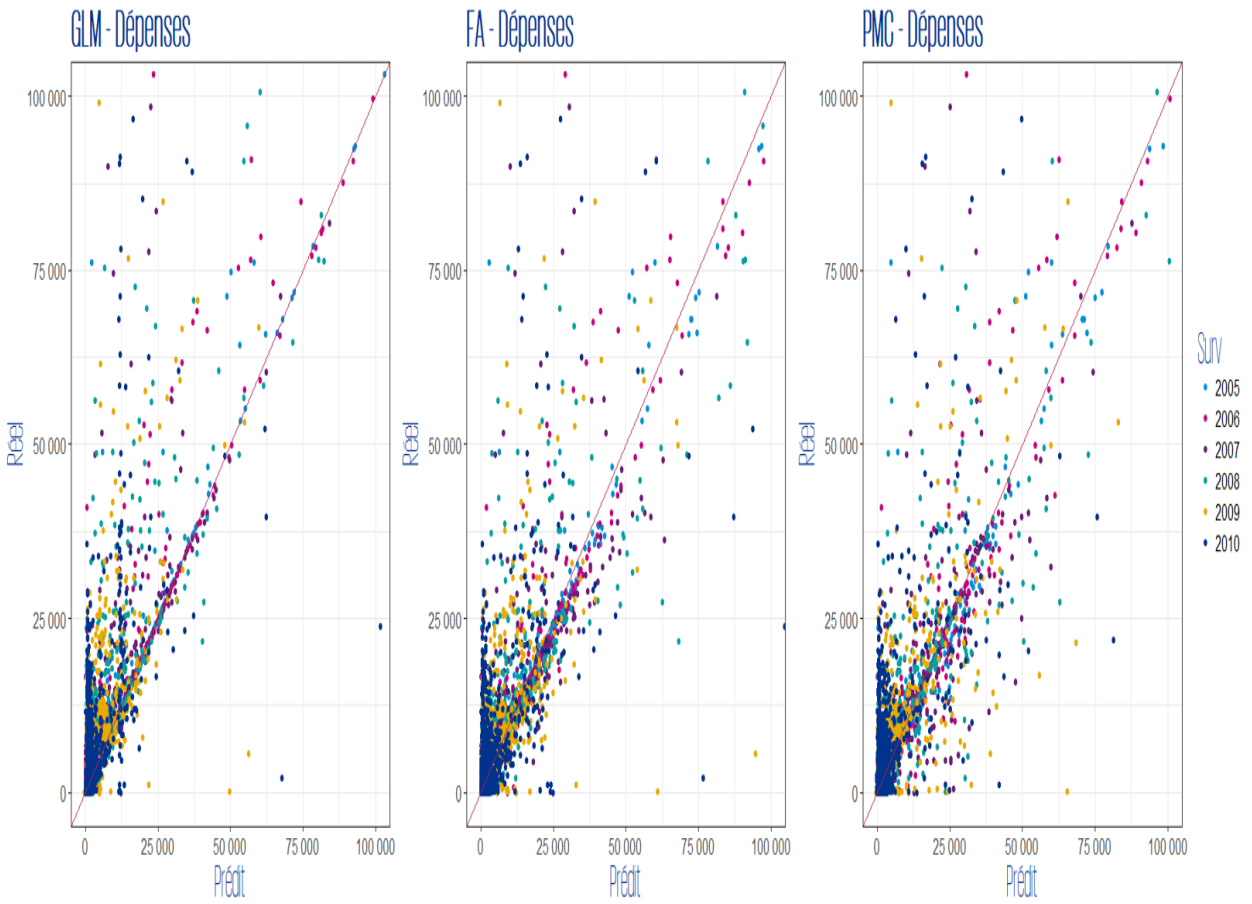


FIGURE 7.5 – Valeurs prédites contre valeurs réelles - Dépenses.

Le modèle GLM sur-estime peu les prédictions de dépenses sur l'intervalle considéré. Nous observons aussi une plus grande volatilité des préditions pour le modèle PMC, qui semblent toutefois être relativement symétriques par rapport à la première bissectrice et donc de compenser les erreurs en vision agrégée.

7.1.2 Erreurs agrégées

Nous regardons les prédictions d'un point de vue agrégé, par année de survenance et de développement.

ARE

Tous les modèles l'emportent au moins une fois sur une survenance et un développement. Néanmoins,

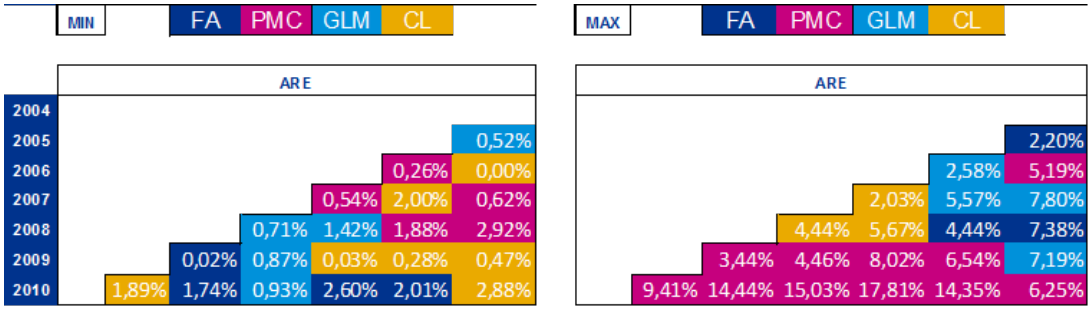


FIGURE 7.6 – Erreurs agrégées de prédiction : ARE.

sur le dernier développement, le modèle qui semble commettre le moins d'erreur au global semble être le Chain-Ladder : sur la dernière année de survenance considérée, il commet 2,88% d'erreur, et notre plus grosse erreur est commise par le modèle PMC avec 6,25% d'erreur. Sur les dernières années de survenances, le modèle obtenu par les baggings de réseaux de neurones semble être de moindre qualité en vision agrégée.

Nous pouvons regarder l'erreur relative de prédiction globale, regroupée par année de développement :

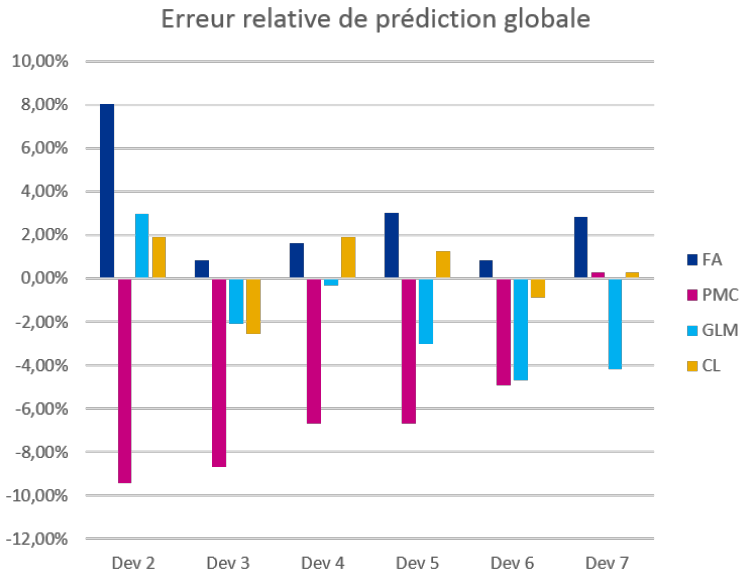


FIGURE 7.7 – Erreurs agrégées de prédiction : Erreurs de prédiction globale.

Une valeur négative représente ici une sous-évaluation par le modèle, au niveau agrégé. Nous remarquons principalement que le bagging de réseaux de neurones entraîne une sous-évaluation assez importante en vision agrégée, sauf sur le dernier développement. Le modèle obtenu par les forêts aléatoires est relativement stable à partir du troisième développement, et comparable au Chain-Ladder. Enfin, le modèle issu des GLM pénalisés semble lui aussi sous-évaluer les dépenses mais pour les derniers développements.

Erreur de réserve à 1 an

RESERVE 1 AN (M€)									
	Réel	FA		PMC		GLM		CL	
		M€	Ecart %	M€	Ecart %	M€	Ecart %	M€	Ecart %
2004	-	-	-	-	-	-	-	-	-
2005	0,64	0,97	51,0%	0,94	46,5%	0,72	12,1%	0,93	44,5%
2006	0,97	0,70	-28,2%	0,94	-3,7%	0,62	-36,5%	0,68	-30,1%
2007	1,30	1,56	19,8%	1,23	-5,6%	1,16	-10,7%	1,57	21,1%
2008	1,44	1,58	9,5%	1,27	-12,0%	1,52	5,5%	1,94	34,5%
2009	2,92	2,92	-0,1%	2,61	-10,7%	2,73	-6,5%	2,80	-4,3%
2010	4,16	4,59	10,4%	3,65	-12,2%	4,32	3,9%	4,26	2,5%
TOTAL	11,44	12,32	7,7%	10,63	-7,0%	11,07	-3,2%	12,18	6,5%

FIGURE 7.8 – Erreurs agrégées de prédiction : Réserve à 1 an.

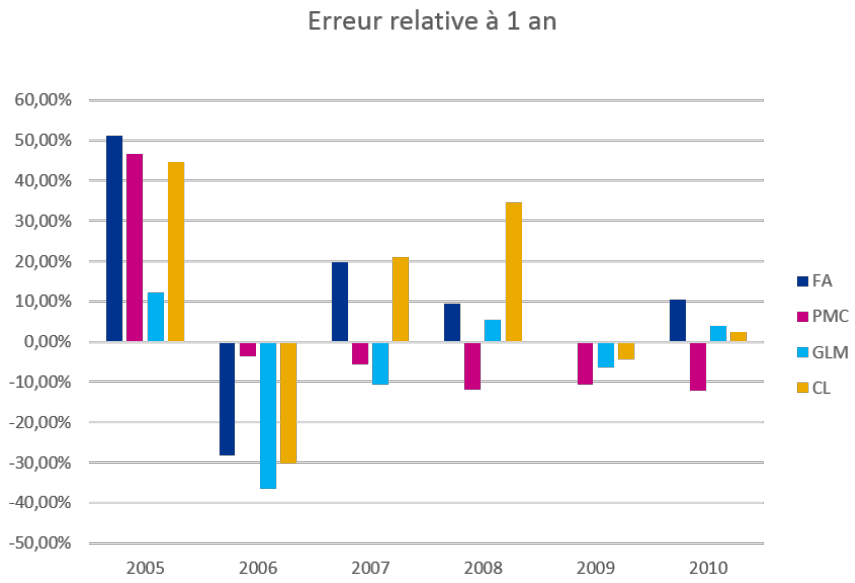


FIGURE 7.9 – Erreurs agrégées de prédiction : Erreur de réserve à 1 an.

Nous mettons en surbrillance, dans le tableau 7.8, pour chaque année de survenance, l'erreur la plus faible en valeur absolue.
En valeur absolue, les modèles sont relativement proches entre eux. Le modèle FA sur-évalue la réserve à 1 an pour les dépenses de 7,7%, tandis que le modèle PMC sous-évalue la sous-évalue de 7%. Notamment, pour chaque année de survenance, le modèle FA tend à surestimer cette réserve et le modèle PMC à la sous-estimer. Si nous regardons sur l'ensemble des années de survenance, le modèle commettant la plus petite erreur de réserve à 1 an est le modèle GLM, avec 3,2% d'erreur.

Erreur de réserve à l'ultime

	RESERVE ULTIME (M€)							
	Réel	FA		PMC		GLM		CL	
		M€	Ecart %	M€	Ecart %	M€	Ecart %	M€	Ecart %
2004	-	-	-	-	-	-	-	-	-
2005	0,64	0,97	51,0%	0,94	46,5%	0,72	12,1%	0,93	44,5%
2006	1,56	1,78	13,7%	2,31	47,7%	1,31	-16,5%	1,56	0,0%
2007	3,71	3,47	-6,6%	3,61	-2,7%	2,47	-33,4%	3,25	-12,4%
2008	4,19	5,22	24,6%	4,59	9,7%	3,71	-11,3%	5,04	20,2%
2009	7,51	7,98	6,2%	7,16	-4,6%	6,52	-13,1%	7,44	-0,9%
2010	11,41	12,04	5,5%	10,62	-6,9%	10,71	-6,1%	11,04	-3,2%
TOTAL	29,02	31,45	8,4%	29,24	0,8%	25,44	-12,3%	29,27	0,8%

FIGURE 7.10 – Erreurs agrégées de prédiction : Réserve à l'ultime.

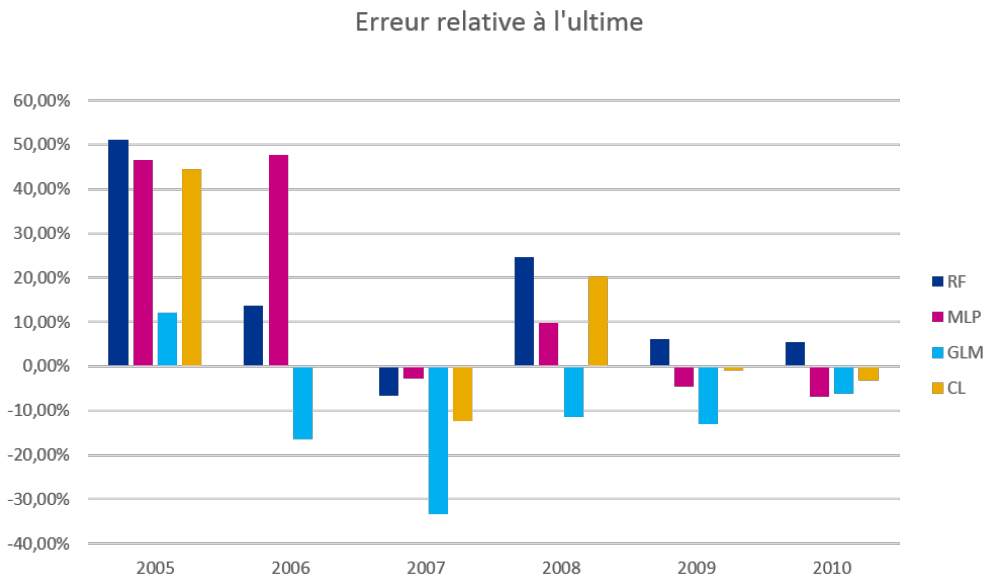


FIGURE 7.11 – Erreurs agrégées de prédiction : Erreur de réserve à l'ultime.

Le modèle MLP semble globalement l'emporter, avec 0,8% d'erreur au global, au même niveau que le Chain-Ladder.
Le modèle PMC commet le moins d'erreurs sur les années de survenance 2007 et 2008 (respectivement 2,7% et 9,7% d'erreur, en valeurs absolue), mais présente plus de difficultés à prédire les deux dernières années de survenances.

Nous allons maintenant analyser les résultats des prédictions pour les charges.

7.2 Résultats de prédiction des charges

7.2.1 Erreurs individuelles

RMSE

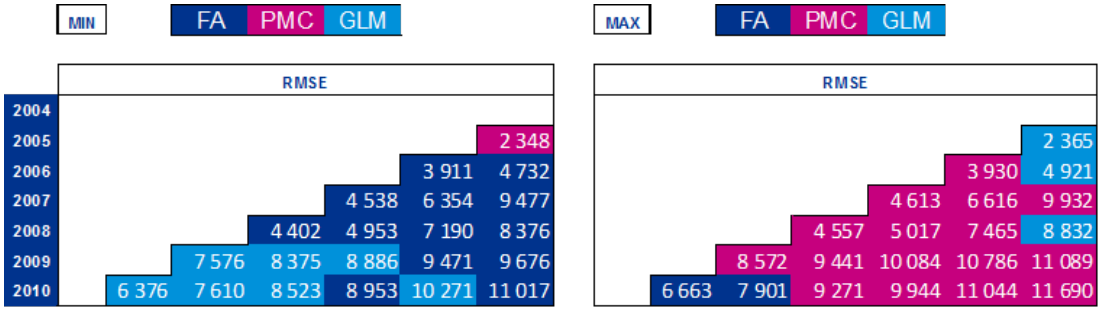


FIGURE 7.12 – Erreurs individuelles de prédiction : RMSE.

Une première observation est de comparer les RMSE obtenus sur les charges et ceux obtenus sur les dépenses. Les premiers sont comparativements plus élevés : les charges sont plus importantes que les dépenses, et les modèles ont plus de mal à prédire les charges que les dépenses.
Cela s'explique notamment par le fait que les charges sont plus volatiles que les dépenses de par leur nature : elles sont estimées par les gestionnaires, peuvent évoluer de manière brutale entre deux développements et sont évaluées lors de la première année de développement avec un forfait.

Parmi nos modèles, celui basé sur les forêts aléatoires semble procure les meilleurs résultats du point de vue du RMSE, notamment pour le dernier développement. Sur la dernière année de survenance, les modèles offrent des performances comparables, avec un RMSE égal à 11 017 pour le dernier développement pour le meilleur modèle (FA) contre 11 690 pour le moins bon modèle (PMC).
Le modèle GLM l'emporte sur les survenances récentes pour les premiers développements, tandis que le modèle PMC présente généralement les erreurs les plus importantes.

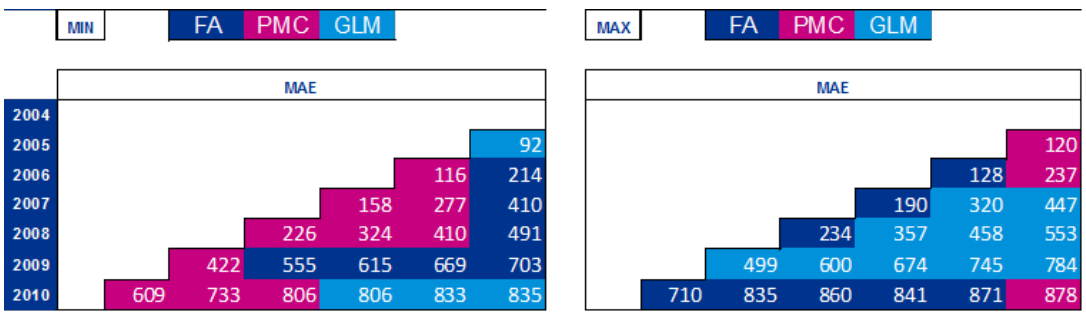


FIGURE 7.13 – Erreurs individuelles de prédiction : MAE.

MAE Ici aussi, le modèle basé sur les forêts aléatoires l'emporte au niveau du dernier développement, pour les années de survenances 2006 à 2009. Toutefois, le modèle PMC commet moins d'erreurs en vision MAE qu'en vision RMSE, ce qui pourrait indiquer des erreurs plus importantes sur les sinistres à charges élevées.

R^2

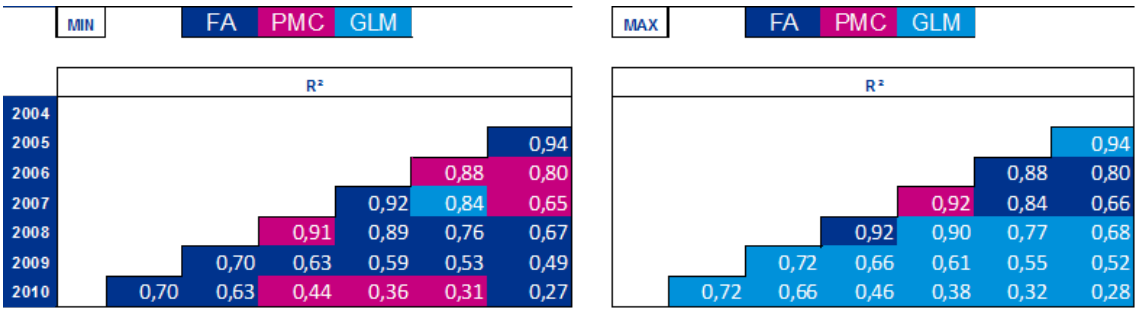


FIGURE 7.14 – Qualité d'ajustement de la régression : R2.

La meilleure qualité d'ajustement est ici obtenue par le modèle GLM, bien que les modèles soient très proches entre eux. Si nous regardons le dernier développement, nous observons au maximum 0,01 d'écart entre le meilleur et le moins bon R^2 obtenu. Comme pour les dépenses, la qualité d'ajustement est satisfaisante pour les survenances anciennes, avec un R^2 maximum à 0,94 pour le dernier développement de l'année de survenance 2005. Elle se détériore relativement rapidement par la suite. La qualité d'ajustement pour le dernier développement de la dernière année de survenance, en 2010 semble relativement médiocre.
Globalement, la qualité des prédictions individuelles se détériore au fil des développements et des années de survenances.

Comparons graphiquement les prédictions des modèles et les valeurs réelles pour le dernier développement :

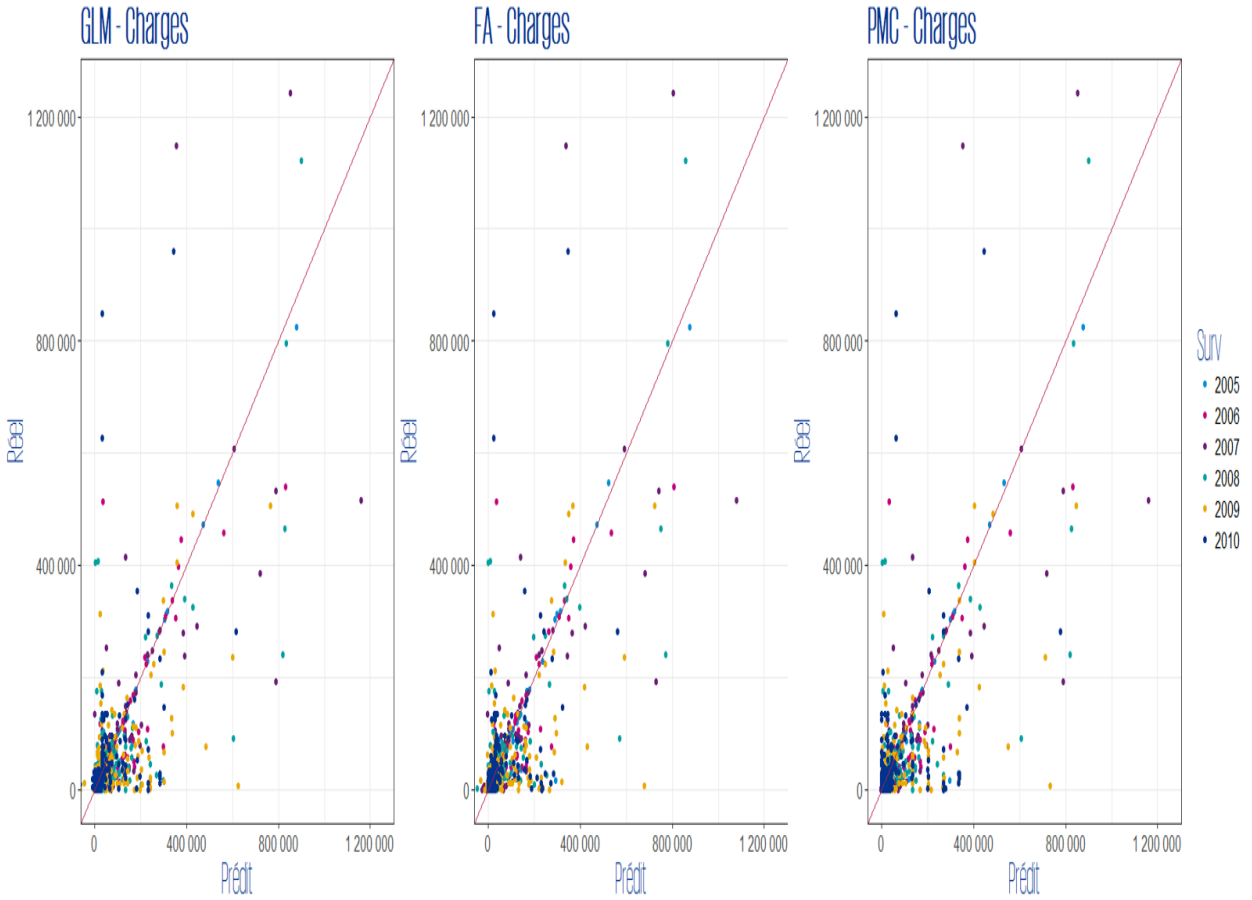


FIGURE 7.15 – Valeurs prédites contre valeurs réelles - Charges.

Les modèles présentent des prédictions graphiquement très proches les unes des autres. Notons toutefois que nous observons une volatilité bien plus importante dans la prédiction des charges que dans la prédiction des dépenses.

Si nous zoomons sur l'intervalle de valeurs $[0; 400000]$:

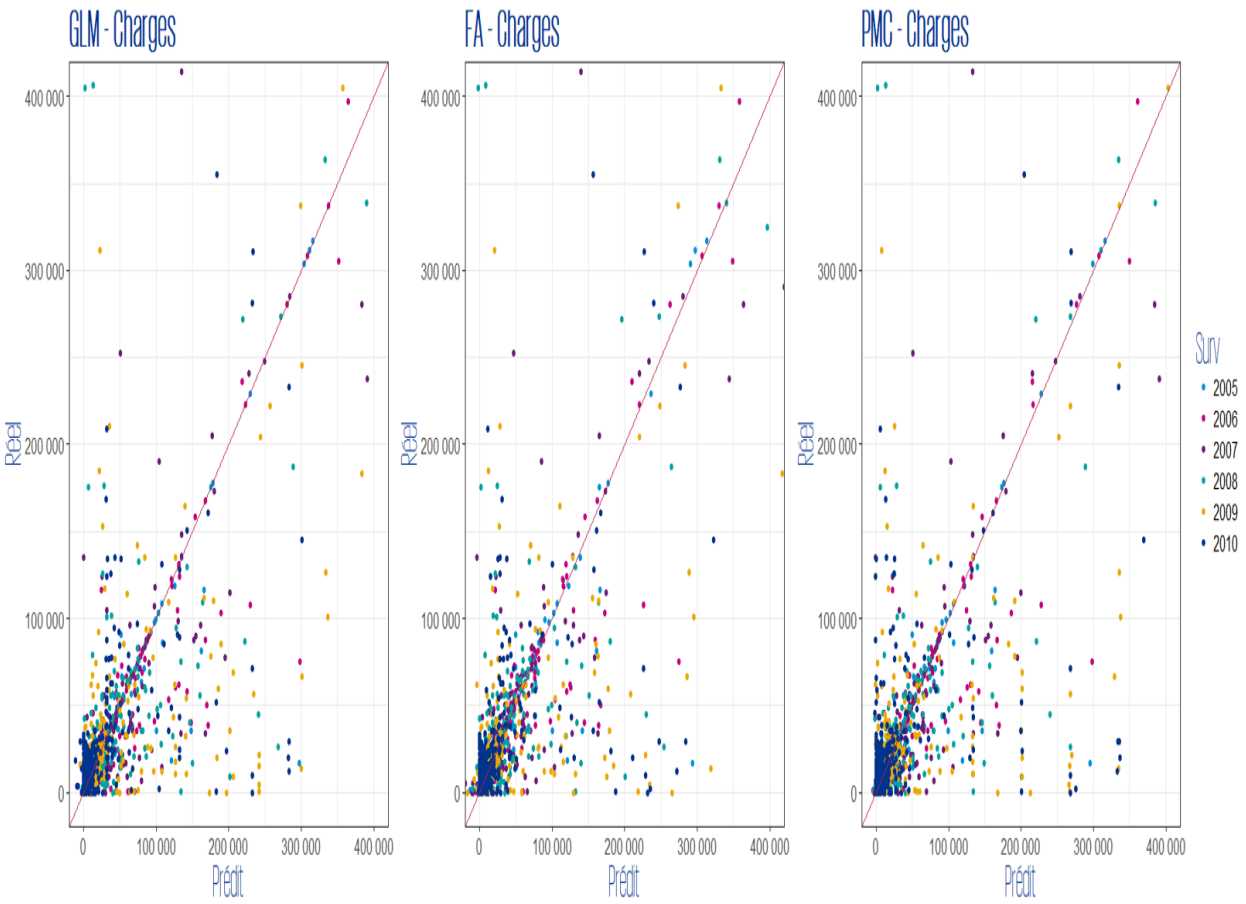


FIGURE 7.16 – Valeurs prédites contre valeurs réelles - Charges.

La qualité de prédiction est moindre sur les années de survenances récentes.

Nous analysons maintenant les résultats de prédictions pour les erreurs agrégées.

7.2.2 Erreurs agrégées

ARE

La méthode Chain-Ladder obtient les erreurs les plus faibles pour certaines années de survenances,

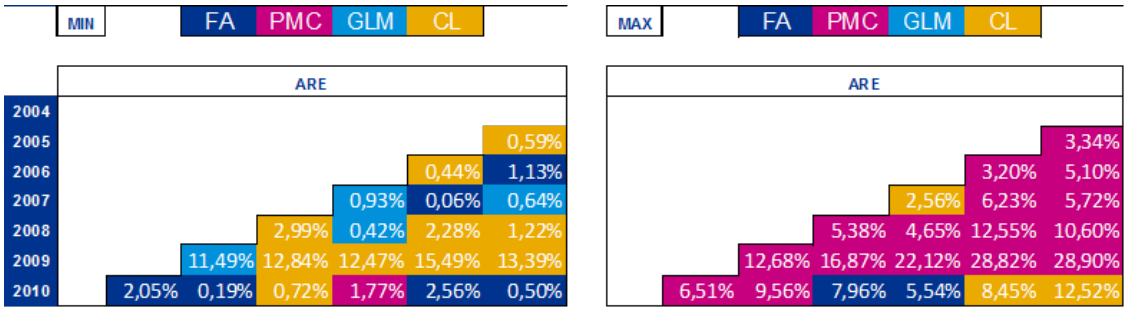


FIGURE 7.17 – Erreurs agrégées de prédiction : ARE.

tandis que le modèle basé sur le bagging de réseaux de neurones surestime de façon conséquente les charges.
Par exemple, pour l'année de survenance 2009, le modèle PMC débute très proche du meilleur modèle, avec 12,68% d'erreur contre 11,49% pour le modèle GLM, mais finit à 28,90% d'erreur pour le dernier développement contre 13,39% pour le Chain-Ladder.
Le modèle FA obtient des résultats très intéressants sur la dernière année de survenance, avec seulement 0,5% d'erreur relative pour le dernier développement contre 12,52% pour le Chain-Ladder.

Si nous regardons l'erreur de prédiction globale aggrégée par développement :

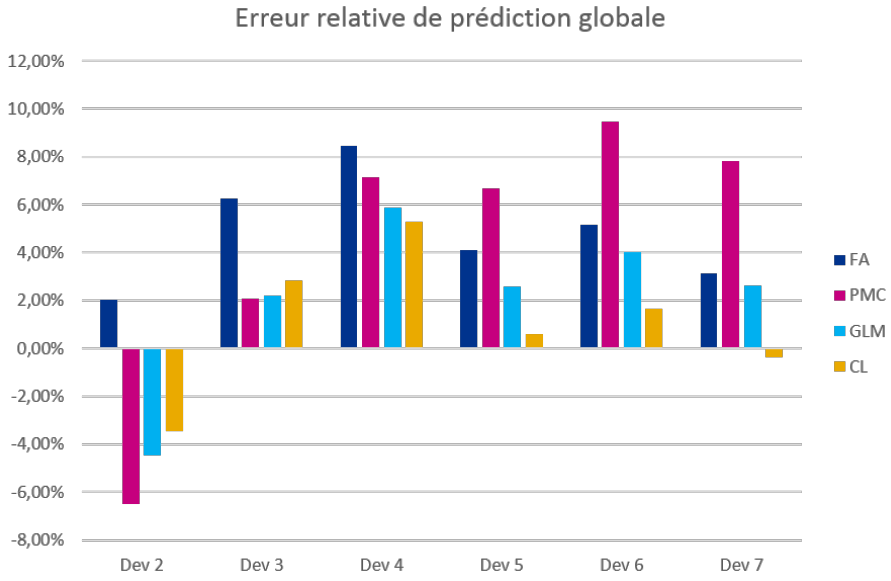


FIGURE 7.18 – Erreurs agrégées de prédiction : Erreurs de prédiction globale.

La plupart des modèles sous-estiment le deuxième développement au niveau agrégé, puis surestiment les développements ultérieurs, notamment le modèle PMC qui surestime de plus de 6% chacun des 4 derniers développements.

Erreur de réserve à 1 an

	RESERVE 1 AN (M€)								
	Réel	FA		PMC		GLM		CL	
		M€	Ecart %	M€	Ecart %	M€	Ecart %	M€	Ecart %
2004	-	-	-	-	-	-	-	-	-
2005	-0,93	-0,81	-13,4%	-0,31	-66,6%	-0,72	-22,8%	-0,82	**-11,7%**
2006	-0,61	-0,46	-25,2%	0,03	-105,2%	-0,39	-35,6%	-0,52	**-14,5%**
2007	-0,73	-1,15	57,3%	-0,24	-67,3%	-0,96	**30,3%**	-1,35	83,8%
2008	-1,60	-0,84	-47,4%	-0,45	-71,9%	-0,72	-55,0%	-0,96	**-39,9%**
2009	-3,43	-0,84	-75,5%	-0,65	-81,0%	-0,91	**-73,4%**	-0,90	-73,8%
2010	3,46	3,87	**11,9%**	2,15	-37,8%	2,57	-25,8%	2,76	-20,1%
TOTAL	-3,85	-0,23	-94,0%	0,53	-113,8%	-1,14	-70,5%	-1,79	**-53,5%**

FIGURE 7.19 – Erreurs agrégées de prédiction : Réserve à 1 an.

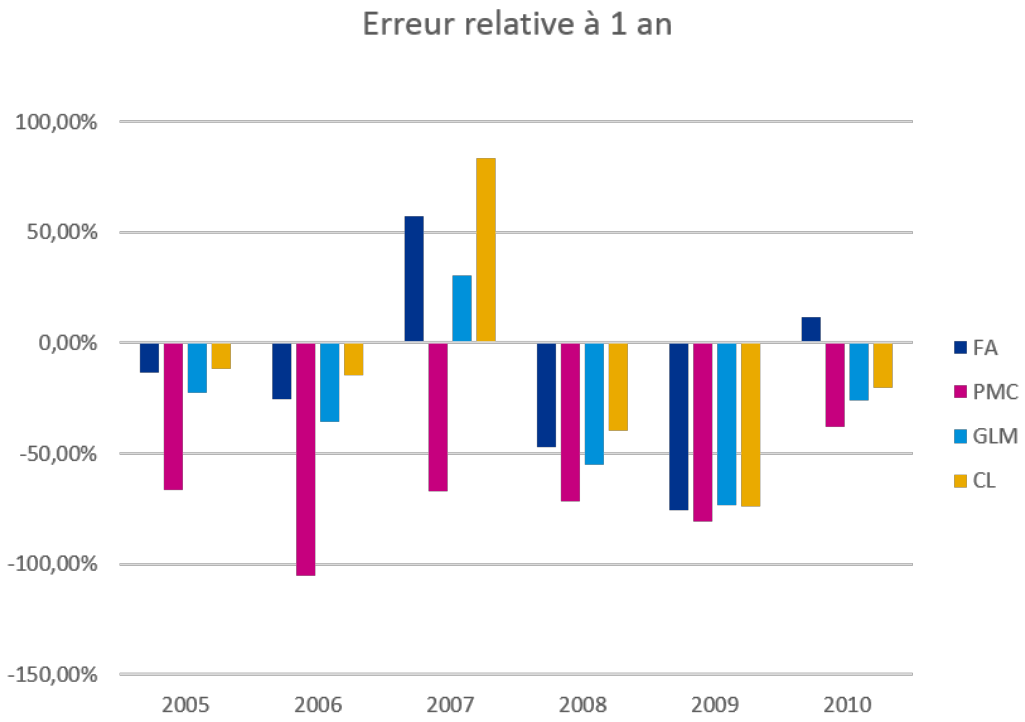


FIGURE 7.20 – Erreurs agrégées de prédiction : Erreur de réserve à 1 an.

Nous remarquons ici que le modèle qui commet le moins d'erreurs sur la réserve à 1 an est la méthode Chain-Ladder, qui sous-estime la réserve à 1 an de -53,5% (la réserve à 1 an réelle est ici faible, ce qui explique les grands écarts en pourcentages). Les prédictions réalisées par les modèles FA et GLM sont proches, respectivement à -70,5% et -94%. Le modèle PMC semble lui plus éloigné du réel, et prédit une réserve à 1 an positive et non pas négative.

Erreur de réserve à l'ultime

	RESERVE ULTIME (M€)								
	Réel	FA		PMC		GLM		CL	
		M€	Ecart %	M€	Ecart %	M€	Ecart %	M€	Ecart %
2004	-	-	-	-	-	-	-	-	-
2005	-0,93	-0,81	-13,4%	-0,31	-66,6%	-0,72	-22,8%	-0,82	**-11,7%**
2006	-1,59	-1,37	**-13,5%**	-0,62	-61,2%	-1,08	-31,9%	-1,37	-13,6%
2007	-2,19	-2,54	16,2%	-0,90	-58,8%	-2,05	**-6,6%**	-2,90	32,4%
2008	-3,27	-2,91	-11,0%	-1,18	-63,8%	-2,53	-22,9%	-3,52	**7,4%**
2009	-7,12	-3,73	-47,6%	-1,85	-74,1%	-3,47	-51,2%	-4,68	**-34,3%**
2010	1,43	1,34	**-6,3%**	0,28	-80,2%	-0,76	-153,1%	-0,83	-158,0%
TOTAL	-13,68	-10,03	-26,7%	-4,58	-66,5%	-10,60	-22,5%	-14,11	**3,2%**

FIGURE 7.21 – Erreurs agrégées de prédiction : Réserve à l'ultime.

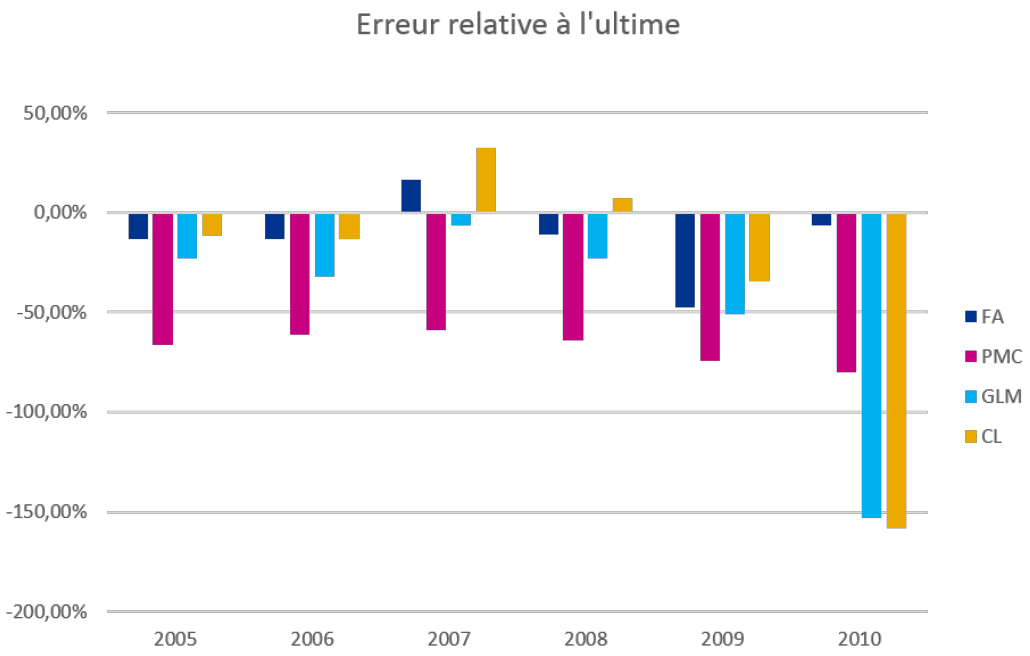


FIGURE 7.22 – Erreurs agrégées de prédiction : Erreur de réserve à l'ultime.

La méthode Chain-Ladder présente l'erreur la plus faible pour la réserve ultime, à 3,2%. Toutefois, cette dernière profite de nombreux effets de compensations sur les diverses années de survenances, qui semblent moins marqués pour le modèle FA. Le modèle PMC semble sous-estimer la réserve à l'ultime de manière relativement constante et conséquente, sur les différentes années de survenances, en moyenne plus de 60% d'écart.

En analysant nos résultats, nous observons que nos modèles offrent des prédictions globales d'un ordre de grandeur proche de la méthode Chain-Ladder. Toutefois il semble délicat de les départager de manière unanime.
Sur les dépenses, tous les modèles offrent des prédictions de très bonne qualité sur la première projection calendaire après la dernière diagonale connue, et le modèle PMC semble être le modèle commetant le moins d'erreurs pour la prédiction à l'ultime.
Sur les charges, les modèles ont plus de difficultés du fait de la volatilité inhérente à ces estimations, mais le modèles FA présente les résultats les plus intéressants.

Nous allons maintenant regarder les prédictions de nos modèles sur le triangle total.

Chapitre 8

Résultats sur le triangle total

Pour appliquer nos modèles sur le triangle total, nous allons d'abord définir nos nouveaux paramètres optimaux. Nous allons raisonner comme sur le triangle inclus, en appliquant une validation croisée à 10 sous-échantillons sur le triangle échantillon.
Nous conserverons certains paramètres mis en place sur le triangle inclus. Ceux que nous allons réévaluer sont :

- Pour les réseaux de neurones : le nombre de neurones par couche cachée,
- Pour les forêts aléatoires : le nombre de variables testées à chaque séparation (paramètre `mtry`),
- Pour les GLM pénalisés : le paramètre de pénalisatoin λ et le paramètre de mélange Ridge/Lasso α.

Ensuite, nous validerons ces modèles par une étape de back-testing : nous utiliserons ces modèles pour prédire la dernière diagonale connue, à partir du triangle exclus de cette diagonale.
Enfin, nous analyserons les prédictions totales sur notre triangle initial total.

8.1 Nouveaux paramètres

Nous commençons par trouver les nouveaux paramètres optimaux dépendant des années de développements. Nous procédons comme sur le triangle inclus, en utilisant les paramètres fixes trouvés sur celui-ci.

Pour rappel, ces derniers sont :

- Pour les forêts aléatoires : règle de segmentation `variance`, nombres d'individus minimum dans les feuilles terminales égal à 10 pour les dépenses et 1 pour les charges.
- Pour les réseaux de neurones : 2 couches cachées, fonction d'activation tangente hyperbolique, algorithme de rétropropagation du gradient de l'erreur. Pour les dépenses, un taux d'apprentissage de 0.1 et un momentum nul. Pour les charges un taux d'apprentissage et un momentum égaux à 0.25.

Nous obtenons les résultats suivants :

Dépenses												
Développement	2	3	4	5	6	7	8	9	10	11	12	13
Alpha	0.5	0	0	0	0.05	0	0	0	0	0.5	0	0
Lambda	0.00201	0.01002	0.04810	0.01403	0.02806	0.01203	0.06012	0.22245	0.07014	0.01002	1	0.04348

Charges												
Développement	2	3	4	5	6	7	8	9	10	11	12	13
Alpha	0	0.2	0	0.25	0	0.05	0.05	0.05	0	0.05	0	0.05
Lambda	0.00201	0.00401	0.02606	0.00201	0.01002	1	1	0.33467	0.08818	0.162333	1	1

FIGURE 8.1 – Paramètres alpha et lambda optimaux pour le triangle total

Paramètres pour le modèle GLM Comme sur le triangle inclus, notre modèle privilégie la pénalisation Ridge à la pénalisation LASSO (paramètre α proche de 0).

Dépenses												
Développement	2	3	4	5	6	7	8	9	10	11	12	13
Mtry	5	3	2	3	1	1	1	1	1	1	1	1

Charges												
Développement	2	3	4	5	6	7	8	9	10	11	12	13
Mtry	6	6	1	1	1	1	1	1	1	1	1	1

FIGURE 8.2 – Paramètre mtry pour le triangle total

Paramètres pour le modèle des forêts aléatoires Comme sur le triangle inclus, nous remarquons que le paramètre `mtry` décroit au fur et à mesure des années de survenance. Néanmoins, cette fois-ci pour les charges, nous avons un `mtry` (nombre de variables testées à chaque segmentation) important pour la prédiction de la deuxième année de développement, égal à 6.

Dépenses												
Développement	2	3	4	5	6	7	8	9	10	11	12	13
Neurones couche 1	3	4	4	3	3	4	3	3	3	3	3	3
Neurones couche 2	1	3	3	3	2	1	2	2	2	3	1	1

Charges												
Développement	2	3	4	5	6	7	8	9	10	11	12	13
Neurones couche 1	3	3	3	4	4	5	5	5	4	3	4	5
Neurones couche 2	3	3	1	2	2	3	2	2	3	1	1	3

FIGURE 8.3 – Architecture optimale pour le triangle total

Paramètres pour le modèle des réseaux de neurones Sur les dépenses, nous remarquons que le nombre de neurones de la première couche semble relativement stabilisé à 3, tandis que le nombre de neurones de la deuxième couche varie. Le nombre de neurones total semble décroître au fur et à mesure des développements.
Pour les charges, le nombre de neurones semble bien plus volatile, nous ne détectons pas de tendance

spécifique en fonction de l'année de développement.

Une fois nos paramètres optimaux définis pour le triangle total, nous regardons les résultats obtenus pour le back-testing sur la dernière diagonale connue.

Backtesting sur la dernière diagonale Nous analysons ici la prédiction par nos modèles de provisionnement individuel, ainsi que par la méthode Chain-Ladder, de la dernière diagonale connue. Nous allons prédire la dernière diagonale connue, pour les années de survenances 2005 à 2015, à partir du triangle total privé de cette diagonale.

Commençons par les dépenses :

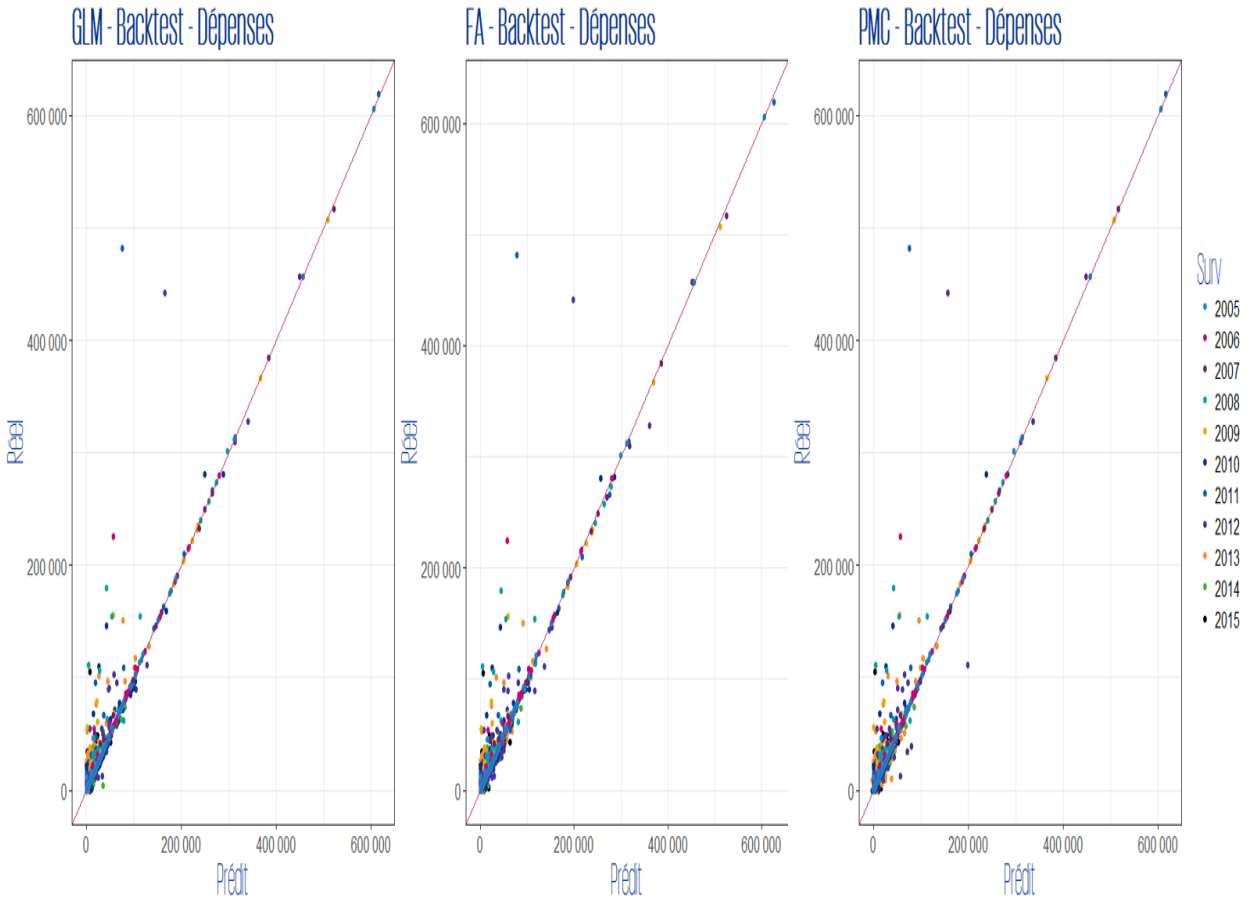


FIGURE 8.4 – Backtesting - Dépenses - Prédictions contre valeurs réelles

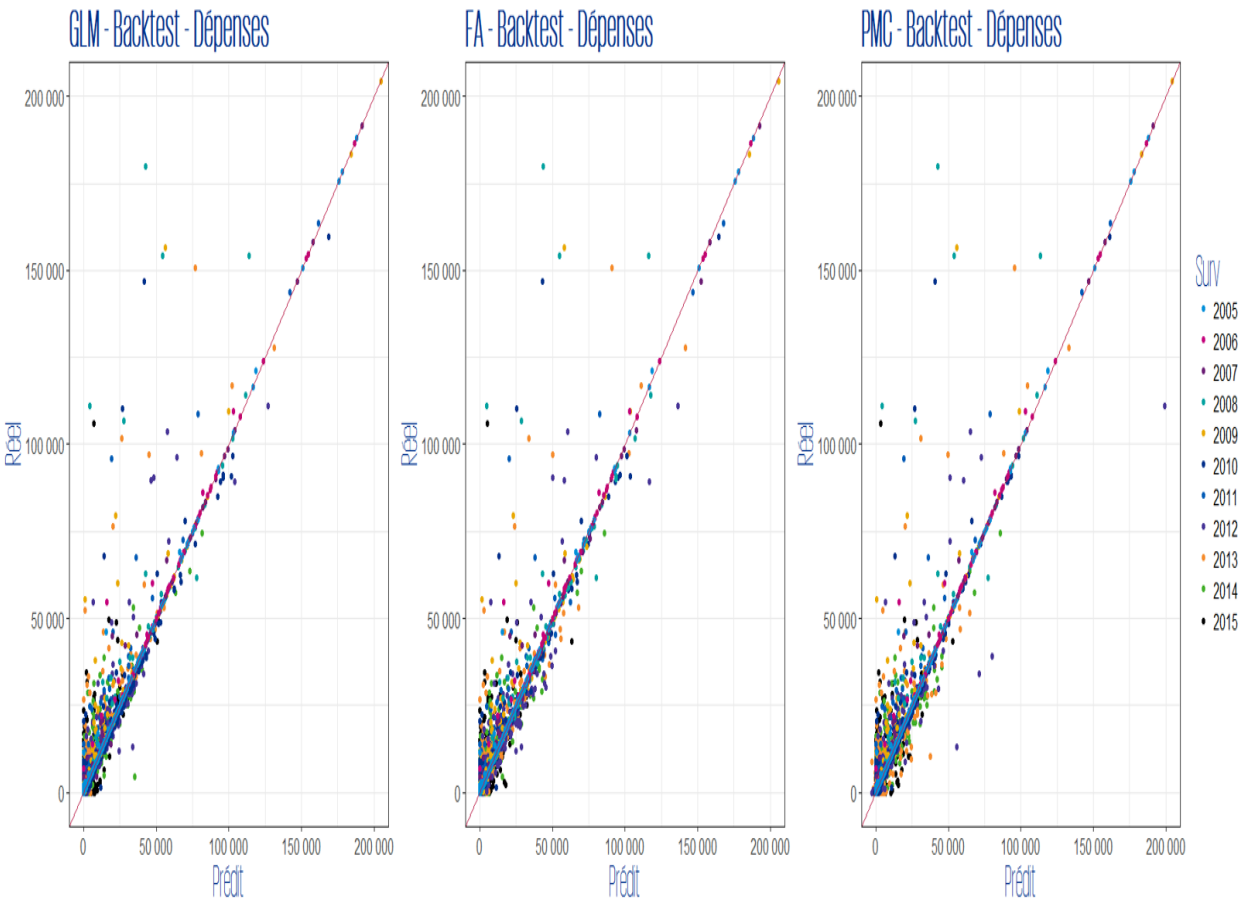


FIGURE 8.5 – Backtesting - Dépenses - Prédictions contre valeurs réelles

	RMSE			R^2			MAE		
	FA	PMC	GLM	FA	PMC	GLM	FA	PMC	GLM
2005	**228,30**	229,14	228,70	**1,00**	1,00	1,00	5,98	16,65	**5,27**
2006	**1 200,83**	1 202,66	1 202,56	**0,95**	0,95	0,95	16,78	20,38	**16,25**
2007	265,84	**254,78**	260,99	1,00	**1,00**	1,00	**15,63**	23,05	16,18
2008	**1 602,38**	1 615,81	1 611,84	**0,92**	0,92	0,92	47,85	**36,35**	47,18
2009	**1 018,67**	1 043,73	1 039,45	**0,98**	0,97	0,97	44,80	**37,71**	45,48
2010	**1 139,14**	1 193,20	1 156,76	**0,94**	0,93	0,94	**60,81**	203,72	69,68
2011	**3 064,83**	3 089,61	3 090,13	**0,83**	0,83	0,83	84,73	89,72	**83,63**
2012	**2 030,69**	2 414,44	2 246,70	**0,84**	0,76	0,81	**104,81**	179,93	118,30
2013	**1 214,18**	1 305,74	1 326,24	**0,85**	0,83	0,84	**150,16**	238,27	161,17
2014	**838,34**	886,14	872,32	**0,83**	0,82	0,81	**229,28**	271,47	235,43
2015	**1 273,20**	1 293,28	1 281,86	**0,47**	0,46	0,47	280,05	358,60	**273,43**

FIGURE 8.6 – Backtesting - Dépenses - Erreurs individuelles

Les prédictions sont d'une qualité plus que correcte sur les années de survenances anciennes, quand nous regardons la qualité d'ajustement via le R^2. Ce dernier est supérieur à 0,83 pour toutes les années de suvenances hormis la dernière.
De par le MAE, nous remarquons notamment une dégradation assez rapide des prédictions sur les 5 dernières années de survenance.

Le modèle qui semble commettre généralement le moins d'erreur est encore une fois le modèle FA, notamment sur les sinistres présentant les dépenses les plus larges, quand nous comparones le RMSE et le MAE.

Si nous regardons l'erreur relative absolue de prédiction à 1 an :

	PRED 1 AN (M€)								
	Réel	FA		PMC		GLM		CL	
		M€	Ecart %	M€	Ecart %	M€	Ecart %	M€	Ecart %
2005	16,19	16,19	0,0%	16,43	1,4%	16,17	-0,1%	16,18	-0,1%
2006	15,37	15,16	-1,4%	15,07	-1,9%	15,14	-1,5%	15,16	-1,4%
2007	16,75	16,85	0,6%	16,79	0,2%	16,84	0,6%	16,88	0,8%
2008	14,97	14,72	-1,7%	14,40	-3,8%	14,67	-2,0%	14,75	-1,4%
2009	14,25	14,13	-0,9%	13,68	-4,0%	14,08	-1,2%	14,17	-0,6%
2010	12,64	12,74	0,8%	12,97	2,6%	12,74	0,8%	12,75	0,9%
2011	13,45	13,23	-1,6%	12,74	-5,3%	13,09	-2,7%	13,32	-1,0%
2012	11,58	11,58	0,0%	11,48	-0,9%	11,55	-0,3%	11,67	0,8%
2013	10,81	10,57	-2,2%	8,63	-20,2%	10,39	-3,8%	10,73	-0,7%
2014	9,35	9,03	-3,3%	5,34	-42,8%	8,95	-4,2%	9,26	-0,9%
2015	6,62	5,66	-14,5%	7,99	20,7%	5,13	-22,5%	6,44	-2,8%
TOTAL	141,98	139,85	-1,5%	135,52	-4,5%	138,76	-2,3%	141,32	-0,5%

FIGURE 8.7 – Backtesting - Dépenses - Erreurs relatives absolues à 1 an

Globalement, les modèles semblent fournir des prédictions compétitives avec un Chain-Ladder, d'un point de vue agrégé. Toutes les prédictions au global sont en dessous de 5% d'erreur absolue.

Si nous regardons maintenant les charges :

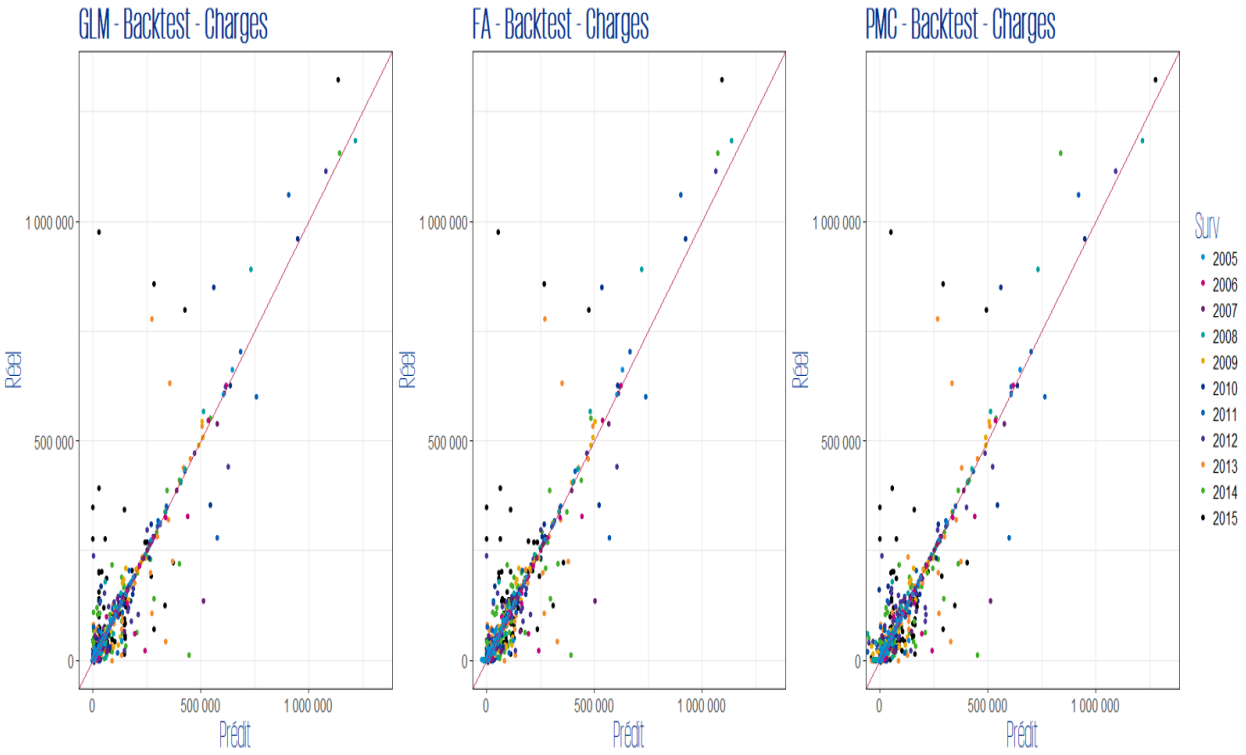


FIGURE 8.8 – Backtesting - Charges - Prédictions contre valeurs réelles

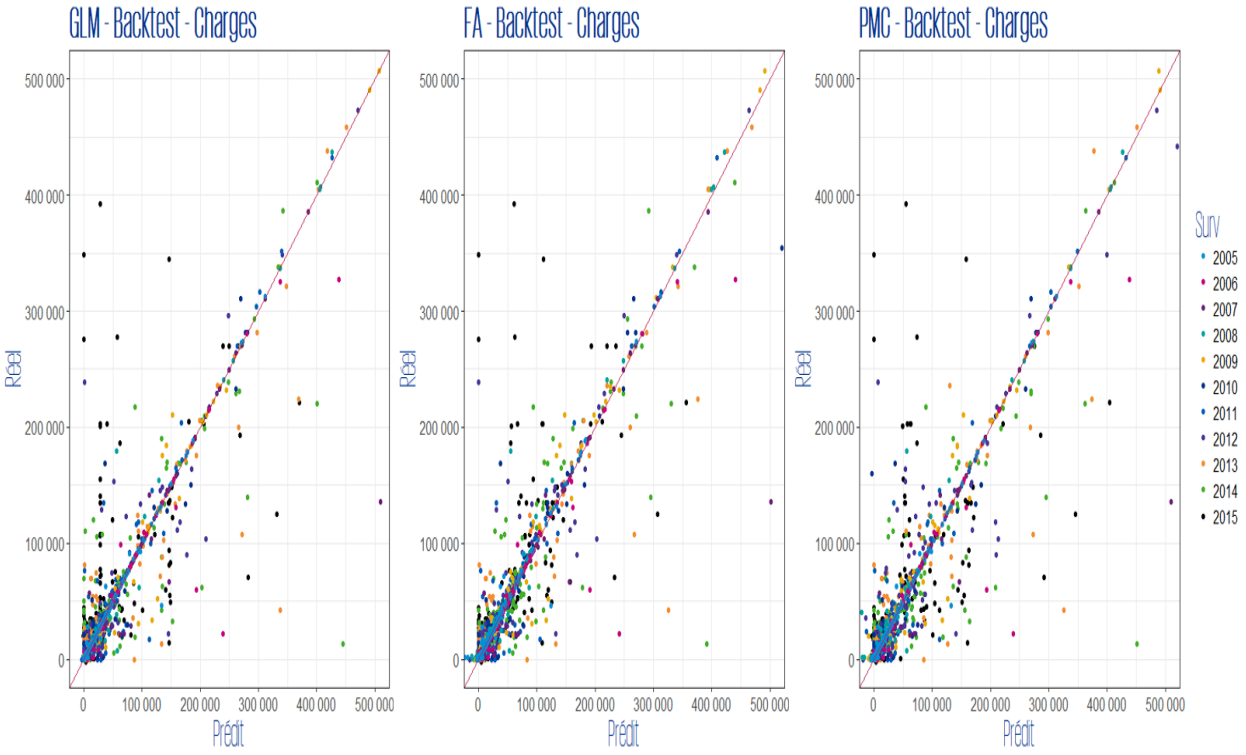


FIGURE 8.9 – Backtesting - Charges - Prédictions contre valeurs réelles

Nous remarquons une prédiction plus difficile pour les charges que pour les dépenses, et une

	RMSE			R^2			MAE		
	FA	PMC	GLM	FA	PMC	GLM	FA	PMC	GLM
2005	692,91	675,47	**648,42**	0,99	0,99	**0,99**	**29,20**	121,08	39,03
2006	1 973,09	**1 964,96**	1 965,01	**0,95**	0,95	0,95	41,09	46,40	**37,09**
2007	**2 677,00**	2 723,13	2 720,37	**0,93**	0,92	0,92	46,33	115,16	**39,71**
2008	1 790,47	1 932,53	**1 618,61**	**0,99**	0,98	0,99	**63,22**	109,18	72,28
2009	1 293,27	1 497,67	**1 251,00**	0,98	0,98	**0,98**	68,25	220,18	**66,89**
2010	2 921,45	3 088,44	**2 846,04**	0,95	0,94	**0,95**	101,18	104,24	**94,24**
2011	**3 079,52**	3 305,22	3 121,11	**0,95**	0,94	0,95	**123,06**	276,29	132,80
2012	**2 954,12**	3 292,32	3 043,04	**0,94**	0,92	0,93	**147,22**	277,24	150,43
2013	**5 455,91**	5 640,07	5 456,50	**0,82**	0,80	0,82	**241,87**	584,39	246,73
2014	**4 160,43**	5 257,30	4 605,24	**0,90**	0,85	0,89	344,73	538,87	**325,18**
2015	**10 470,26**	10 617,57	11 060,16	**0,66**	0,63	0,61	**780,21**	1 045,70	869,55

FIGURE 8.10 – Backtesting - Charges - Erreurs individuelles

volatilité plus importante.
Le modèle qui semble l'emporter du point de vue des erreurs individuelles et de l'ajustement est le modèle FA sur les années de survenances 2011 à 2015. Pour les survenances plus anciennes, il semble moins aisé de départager les modèles.

Enfin si nous regardons l'erreur de prédiction à 1 an :

	PRED 1 AN (M€)							
	Réel	FA		PMC		GLM		CL	
		M€	Ecart %	M€	Ecart %	M€	Ecart %	M€	Ecart %
2005	17,11	16,72	**-2,3%**	15,43	-9,9%	16,48	-3,7%	16,40	-4,2%
2006	17,31	17,94	3,6%	17,82	**2,9%**	17,90	3,4%	17,94	3,6%
2007	18,39	19,00	**3,3%**	20,50	11,5%	19,02	3,4%	19,03	3,5%
2008	19,73	18,83	-4,6%	19,28	**-2,3%**	18,77	-4,9%	18,38	-6,8%
2009	18,03	17,64	-2,2%	20,33	12,7%	17,70	**-1,9%**	17,53	-2,8%
2010	18,01	17,33	-3,8%	17,72	**-1,6%**	17,49	-2,9%	17,38	-3,5%
2011	18,90	18,77	**-0,6%**	16,58	-12,3%	18,76	-0,7%	18,70	-1,0%
2012	17,77	17,45	-1,8%	17,10	-3,8%	17,48	**-1,6%**	17,37	-2,3%
2013	20,22	19,52	**-3,5%**	25,45	25,9%	19,51	-3,5%	19,30	-4,5%
2014	21,47	21,49	**0,1%**	19,21	-10,5%	21,58	0,5%	21,38	-0,4%
2015	23,71	19,14	-19,3%	20,58	**-13,2%**	19,42	-18,1%	19,48	-17,8%
TOTAL	210,65	203,82	-3,2%	209,99	**-0,3%**	204,10	-3,1%	202,90	-3,7%

FIGURE 8.11 – Backtesting - Charges - Erreurs relatives absolues à 1 an

Cette fois-ci, pour chaque année de survenance, au moins un de nos modèles obtient une erreur de prédiction à 1 an inférieur à celle obtenue par la méthode Chain-Ladder. Toutefois aucun modèle ne l'emporte sur toutes les années de survenances, et chacun semble avoir ses qualités propres pour prédire certaines survenances spécifiques.
Hormis la dernière année de survenance 2015, les modèles FA et GLM ccommettent moins de 4,4% d'erreur absolue pour chaque année de survenance. Le modèle PMC commet quant à lui moins d'erreur au global, mais semble présenter plus de variations au fur et à mesure des années de survenances.

Regardons maintenant les prédictions pour le dernier développement du triangle total, le douzième développement, obtenues par nos modèles.

8.2 Prédictions globales obtenues

Les prédictions de dépenses sont les suivantes au dernier développement :

	Ultime prédit - Dépenses (M€)			
	FA	PMC	GLM	CL
	M€	M€	M€	M€
2004	18,14	18,14	18,14	18,14
2005	16,24	16,16	16,23	16,24
2006	15,48	15,34	15,45	15,47
2007	17,02	16,79	16,94	17,01
2008	15,41	15,07	15,30	15,40
2009	15,08	14,52	14,92	15,10
2010	14,05	13,35	13,68	13,85
2011	15,71	14,82	15,14	15,57
2012	14,99	13,21	13,85	14,31
2013	16,12	13,97	14,41	15,20
2014	17,89	14,76	14,33	15,44
2015	18,30	15,32	14,25	16,38
2016	16,95	14,55	12,14	15,26
TOTAL	211,37	195,98	194,78	203,36

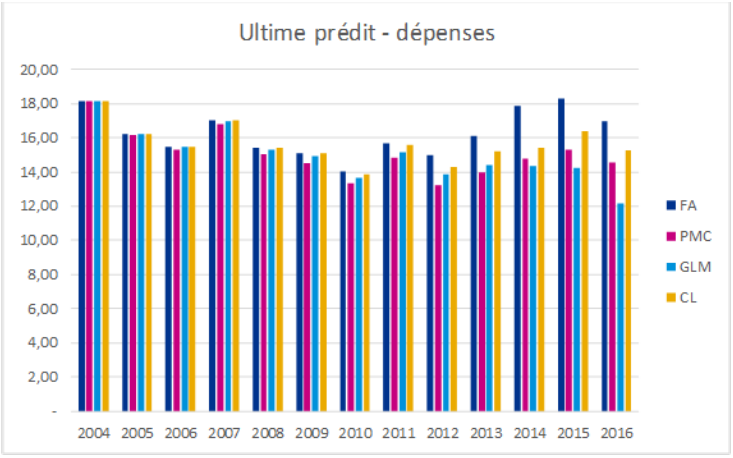


FIGURE 8.12 – Prédictions obtenues pour les dépenses - Triangle total

Les prédictions de dépenses sont relativement proches entre tous les modèles et la méthode Chain-Ladder pour les survenances très anciennes, notamment jusqu'à l'année de survenance 2007. Après cette année de survenance, nous observons une tendance : le modèle FA estime des dépenses globalement plus importantes que les autres modèles, tandis que le modèle PMC estime des dépenses moins importantes que les autres modèles. Par exemple, pour l'année de survenance 2013, le modèle FA prédit 16,12 millions d'euros de dépenses cumulées, tandis que le modèle PMC en prédit 13,97 millions.
A partir de l'année de survenance 2014, c'est le modèle GLM qui prédit les dépenses globalement les plus faibles.

Pour les prédictions de charges au dernier développement :

	Ultime prédit - Charges (M€)			
	FA	PMC	GLM	CL
	M€	M€	M€	M€
2004	19,91	19,91	19,91	19,91
2005	17,13	16,42	17,12	17,13
2006	17,28	16,56	16,98	16,90
2007	18,43	17,61	18,04	17,90
2008	19,59	18,84	19,35	19,10
2009	17,36	16,88	17,04	16,67
2010	17,44	15,78	16,50	16,05
2011	17,79	16,19	17,03	16,37
2012	16,27	14,89	15,42	14,90
2013	18,32	17,35	17,34	16,30
2014	18,98	18,12	17,91	16,80
2015	19,86	20,31	19,57	17,90
2016	15,30	14,85	13,11	13,16
TOTAL	233,64	223,70	225,31	219,08

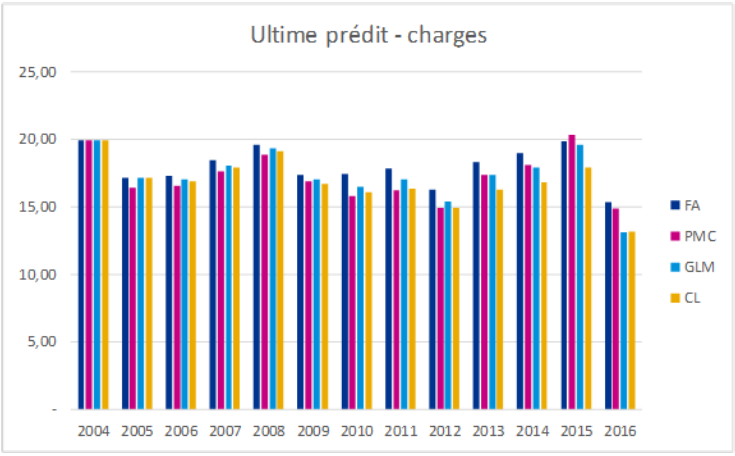


FIGURE 8.13 – Prédictions obtenues pour les charges - Triangle total

Les prédictions de charges semblent globalement proches entre les différents modèles, toutefois des différences semblent apparaître à partir de l'année de survenance 2010.
Pour les années de survenances 2005 à 2013, le modèle PMC prédit des dépenses globalement inférieures aux autres modèles, tendance qui s'inverse à partir de 2014. Le modèle FA semble avoir le même comportement que sur les dépenses, et prédit des charges globalement supérieures aux autres modèles.
Notamment, sur l'ensemble des années de survenances, ce dernier prédit une charge à l'ultime une charge égale à 233,64 millions d'euros, contre 223,70 pour le modèle PMC, 225,31 pour le modèle GLM et 219,08 pour le Chain-Ladder.

Il nous semble notamment intéressant de regarder les charges associées aux sinistres passant en "grave", i.e. les sinistres qui ne présentaient pas de caractère "grave" dans les données connues mais présentant au moins une charge dépassant les 300k euros dans les prédictions.

	FA	PMC	GLM
Charge totale des sinistres passant en graves	40,18	44,12	43,86
% charge totale	17,20%	19,72%	19,47%

FIGURE 8.14 – Charges totales obtenues sur les sinistres devenant graves.

Le modèle FA semble prédire une charge totale moins importante pour les sinistres devenant graves que les deux autres modèles.

Enfin, nous obtenons finalement les PSAP suivantes, en utilisant les dernières dépenses cumulées connues :

	Réel	FA		PMC		GLM		CL	
	Dépenses	Ultime	PSAP	Ultime	PSAP	Ultime	PSAP	Ultime	PSAP
2004	18,14	19,91	1,8	19,91	1,8	19,91	1,8	19,91	1,8
2005	16,19	17,13	0,9	16,42	0,2	17,12	0,9	17,13	0,9
2006	15,37	17,28	1,9	16,56	1,2	16,98	1,6	16,90	1,5
2007	16,75	18,43	1,7	17,61	0,9	18,04	1,3	17,90	1,2
2008	14,97	19,59	4,6	18,84	3,9	19,35	4,4	19,10	4,1
2009	14,25	17,36	3,1	16,88	2,6	17,04	2,8	16,67	2,4
2010	12,64	17,44	4,8	15,78	3,1	16,50	3,9	16,05	3,4
2011	13,45	17,79	4,3	16,19	2,7	17,03	3,6	16,37	2,9
2012	11,58	16,27	4,7	14,89	3,3	15,42	3,8	14,90	3,3
2013	10,81	18,32	7,5	17,35	6,5	17,34	6,5	16,30	5,5
2014	9,35	18,98	9,6	18,12	8,8	17,91	8,6	16,80	7,5
2015	6,62	19,86	13,2	20,31	13,7	19,57	13,0	17,90	11,3
2016	1,32	15,30	14,0	14,85	13,5	13,11	11,8	13,16	11,8
TOTAL	161,43	233,64	72,2	223,70	62,3	225,31	63,9	219,08	57,7

FIGURE 8.15 – PSAP obtenues sur le triangle total

Au global, les PSAP les plus faibles sont obtenues par la méthode Chain-Ladder, à 219,08 millions d'euros. Les PSAP obtenues sont globalement 8% plus importantes pour le modèle PMC, à 223,70 millions d'euros, et 10% pour le modèle GLM, à 225,31 millions d'euros et concentrés sur les 4 dernières années de survenances. Celles obtenues par le modèle FA sont globalement 25% plus importantes, à 233,64 millions d'euros et sont principalement réparties sur les années de survenances 2010 à 2016.

Nous concluons que nos prédictions sont cohérentes avec celles obtenues par la méthode Chain-Ladder, tout du moins en vision agrégée.
Toutefois, nous ne devons pas oublier que nos modèles fournissent des prédictions individuelles. De telles prédictions peuvent être d'une grande valeur pour les assureurs non-vie. En effet, elles présentent une qualité d'ajustement de régression plus que satisfaisante sur la dernière diagonale connue pour les survenances 2004 à 2014, sont de plus cohérentes globalement avec des méthodes agrégées et sont de plus obtenues avec un coût en temps de calcul moindre notamment pour les modèles GLM et FA.

Nous concluons ce chapitre par des utilisations possibles des prédictions individuelles obtenues.

8.3 Pistes de réflexion sur l'utilisation des prédictions

Nous présentons ici des possibilités d'utilisations de nos prédictions individuelles.

Regroupement en groupes homogènes de risque Une première utilisation possible de nos prédictions individuelles serait le regroupement en groupes homogènes de risque.

Nous partons de nos variables explicatives ainsi que de nos prédictions, par exemple les prédictions de dépenses.
Le but est, sans variable cible, de regrouper les sinistres ayant des caractéristiques proches ; c'est un problème de classification non-supervisée.

Nous représentons ci-dessous des éléments obtenus à partir d'une carte auto-adaptative de Kohonen, qui est un type de réseau de neurones adapté à ce type de problèmes : Nous représentons

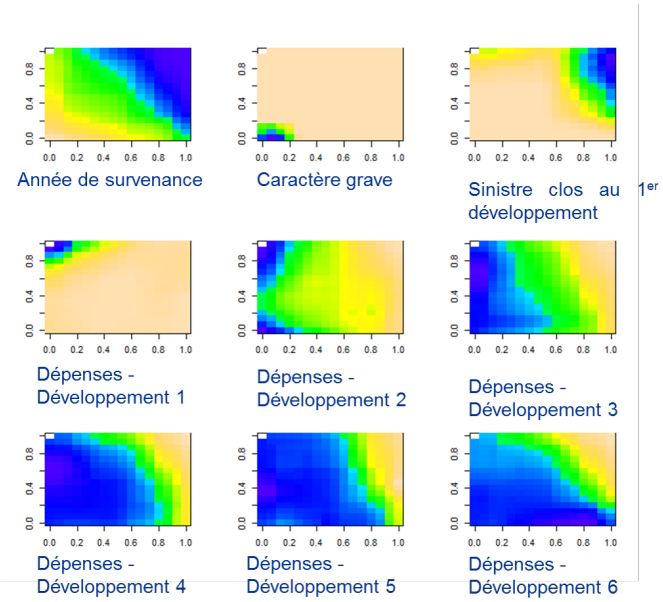


FIGURE 8.16 – Carte auto-adaptative de Kohonen pour la classification non-supervisée.

nos données sur les deux premiers axes obtenus par la carte de Kohonen. Les zones indiquent des sinistres présentant des caractéristiques supposées similaires par notre réseau de neurones.
De nombreuses autres méthodes de classification non-supervisée seraient ici appliquables, comme par exemple l'algorithme des K-plus proches voisins ou bien encore l'algorithme DBSCAN.

Détection du caractère grave de nouveaux sinistres et contribution des variables explicatives Une fois les charges prédites jusqu'à l'ultime, nous pouvons mettre à jour la variable `Grave`. Pour rappel, dans notre étude, un sinistre présente un charactère `Grave` s'il présente au moins

un développement une charge supérieure à 300k euros à un développement donné.
Nous pouvons créer un modèle cherchant à prédire la probabilité qu'un sinistre soit grave, par exemple un modèle de classification basé sur des forêts aléatoires, comme proposé par Malley et al[31]. Nous utilisons ici seulement nos variables explicatives, sans les dépenses ou les charges, pour prédire notre variable `grave` actualisée.

Nous pouvons obtenir la contribution de chaque variable à cette prédiction, comme le représente le graphique suivant :

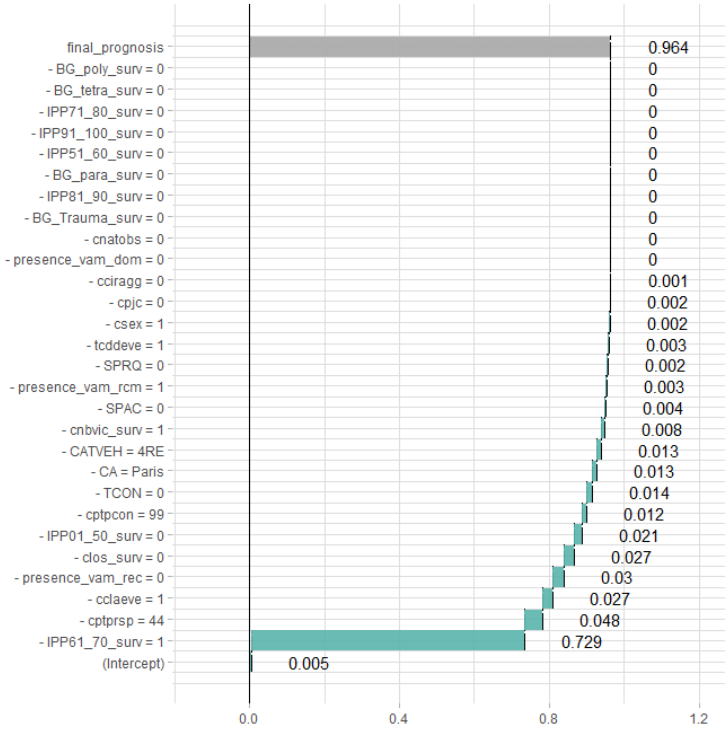


FIGURE 8.17 – Prédiction du caractère grave d'un nouveau sinistre et contribution de chaque variable à cette prédiction

Nous avons ici un nouveau sinistre, présentant notamment les caractéristiques suivantes :
— Présence de victimes ayant un taux d'IPP compris entre 61 et 70 (variable `IPP61_70_surv = 1`)
— Responsabilité entière de l'assuré (variable `Part_Respons = 44`)
Ce sinistre est prédit comme présentant un caractère grave à 96,4% (valeur de `final_prognosis`, la principale contribution à cette prédiction venant de la présence de victimes présentant un taux d'IPP important (72,9 % des 96,4%).
Une telle décomposition pourrait servir d'outil de décision aux gestionnaires, et permettre de sélectionner les sinistres à suivre avec attention. Nous allons maintenant conclure notre étude.

Conclusion

Dans ce mémoire, nous avons élaboré et mis en place un modèle de prédiction de charges et de dépenses dans un but de provisionnement individuel. Nous avons appliqué ce modèle à des données réelles pour une branche à duration longue de responsabilité civile corporelle. Cette base de données est fournie par un assureur français, et nous disposions de 29 variables catégorielles à un niveau individuel ainsi que de 13 années de survenances. Nous avons comparé les résultats obtenus par des méthodes agrégées (Chain-Ladder) à ceux auxquels ont mené des méthodes individuelles (GLM pénalisés, forêts aléatoires, réseaux de neurones).

Nous avons dans un premier temps explicité notre démarche de prédiction des dépenses et des charges. Celle-ci repose sur la prédiction année de développement par année de développement, sur le même principe qu'une méthode Chain-Ladder. Elle est toutefois innovante dans le fait qu'elle s'applique à chaque sinistre individuellement et non pas à un niveau agrégé. Nous pouvons ainsi utiliser de l'information individuelle jusqu'alors peu exploitée au niveau du provisionnement.
Au niveau de l'approche individuelle, nous utilisons plusieurs méthodes : une approche classique basée sur des GLM pénalisés, et une approche permettant de capter des phénomènes d'interactions non-linéaires grâce à des algorithmes d'apprentissage statistique : les forêts aléatoires et les réseaux de neurones.

La première approche était déjà en partie développée dans des mémoires d'actuariat, par exemple ceux de Worthington [48] ou Henin [25]. Toutefois, nous l'agrémentons par l'utilisation de facteurs individuels, comme par exemple le taux d'AIPP des victimes ou le type de séquelles rencontrées par ces derniers (paraplégie, traumatismes crâniens...).

Enfin, après avoir fixé nos paramètres optimaux grâce à la construction d'un échantillon de nos données, nous pouvons, par l'intermédiaire des dépenses et des charges réelles, comparer les résultats obtenus par les différents modèles entre eux, ainsi qu'à ceux obtenus par une méthode Chain-Ladder.

Les méthodes utilisées dans ce mémoire sont intéressantes d'un point de vue théorique, notamment les méthodes d'apprentissage statistique. Nous observons, en particulier lors de notre étape de backtesting, une capacité de prédiction de bonne qualité pour tous nos modèles.

Toutefois, nous devons émettre des réserves sur leurs utilisations. En effet, ces derniers sont sensibles au risque de sur-apprentissage et aux minima locaux, bien que certaines techniques, comme la validation croisée, ou l'utilisation de certains paramètres, comme le momentum, permettent théoriquement de réduire ces risques.

Nous nous interrogeons aussi sur la pertinence de l'utilisation de modèles très délicats à paramétrer comme les réseaux de neurones. Bien que ceux-ci offrent des résultats très pertinents, notamment sur les charges (par exemple 0,3% d'erreur relative toutes années de survenances confondues pour les réseaux de neurones), ils peuvent être très sensibles aux paramètres utilisés et très longs à

entraîner. Notamment, notre bagging de 100 réseaux de neurones nécessitait 26 heures, tandis que notre modèle basé sur les forêts aléatoires ne nécessitait que 30 minutes, et que le modèle basé sur les GLM pénalisés seulement quelques minutes. Les trois modèles offrants des résultats relativement comparables, le temps de calcul supplémentaire semble peu justifiable.

Nous pouvons aussi souligner que la modélisation utilisée est perfectible : nous ne considérons que les variables lors de l'année de survenance des sinistres et ne faisons pas de projections sur les variables pouvant évoluer temporellement. Par exemple le taux d'AIPP d'une victime augmentant au bout de quelques années car son état se trouve aggravé : nous ne considérons la valeur de ce taux qu'à la date de survenance. La mise en place d'une étape supplémentaire, à chaque développement, d'actualisation de ces variables aurait été très lourde dans la modélisation : nous aurions dû prendre en compte de nombreuses interactions entre ces variables et les variables de dépenses et de charges que nous cherchions à prédire. Une modélisation alternative pour pallier à ce problème aurait par exemple été d'introduire des matrices d'état (processus markoviens).

De plus, nous ne modélisons pas la déclaration de sinistres tardifs (IBNR). Ceux-ci pourraient être non négligeables pour le provisionnement, surtout pour les branches à duration longue.
Toutefois, malgré ces réserves, nous obtenons des résultats d'une qualité non négligeable, notamment quand nous regardons les prédictions à 1 an. Que ce soit sur les prédictions de dépenses ou de charges, nous obtenons une erreur relative au global inférieur à 5% pour tous nos modèles.

D'autres approches seraient intéressantes à considérer. Par exemple, nous pourrions envisager d'agréger nos différents modèles afin d'obtenir un modèle unique regroupant les points fort de ceux-ci tout en améliorant la qualité de prédiction, en utilisant par exemple la méthode COBRA proposée par Biau et al [4]. Cette dernière est une méthode d'agrégation non-linéaire de modèles de régression, basée sur une notion de proximité (distance) entre valeurs réelles et valeurs prédites par les modèles à agréger.

Nous pourrions aussi envisager d'ajouter de l'information, par exemple grâce aux dates de déclaration et de clôture des sinistres, ce qui nous permettrait notamment de mettre en place des éléments de la théorie des modèles de durée ou des séries temporelles, qui nécessiteraient toutefois une granularité temporelle plus fine. Un autre point à considérer serait la prédiction simultanée des dépenses et des charges, ainsi que leur corrélation.

Les assureurs sont en recherche permanente de nouvelles méthodes de provisionnement, dans le but d'obtenir des estimations fiables et peu volatiles de leurs provisions. Les utilisations rendues possibles par des méthodes de provisionnement individuelles sont multiples. Les prédictions individuelles peuvent ainsi aussi bien être utilisées pour créer une segmentation des sinistres en groupes de risques homogènes, ou un système de suivi et de détection de cas atypiques.

L'intérêt ne serait pas ici de remplacer directement les méthodes agrégées, qui ont les avantages conséquents d'être très robustes et simples à mettre en œuvre, mais bien d'ajouter de l'information supplémentaire grâce à des méthodes individuelles dans le but de développer des outils d'aide au suivi et à la gestion des sinistres. Nous pourrions aussi sortir du cadre du provisionnement et utiliser les résultats obtenus dans d'autres problématiques actuarielles, comme par exemple la tarification.

De nombreux travaux et études restent à réaliser dans ce domaine, et la recherche actuarielle y consacre de plus en plus d'importance. La multiplicité des approches concernant le provisionnement individuel est impressionnante et nous pousse à nous interroger sur les bases mêmes du provisionnement en assurance non-vie, ainsi que sur la représentation des risques associés, quitte à conceptualiser de nouveaux paradigmes.

Annexe

Chapitre 9

Vérification des hypothèses Chain-Ladder

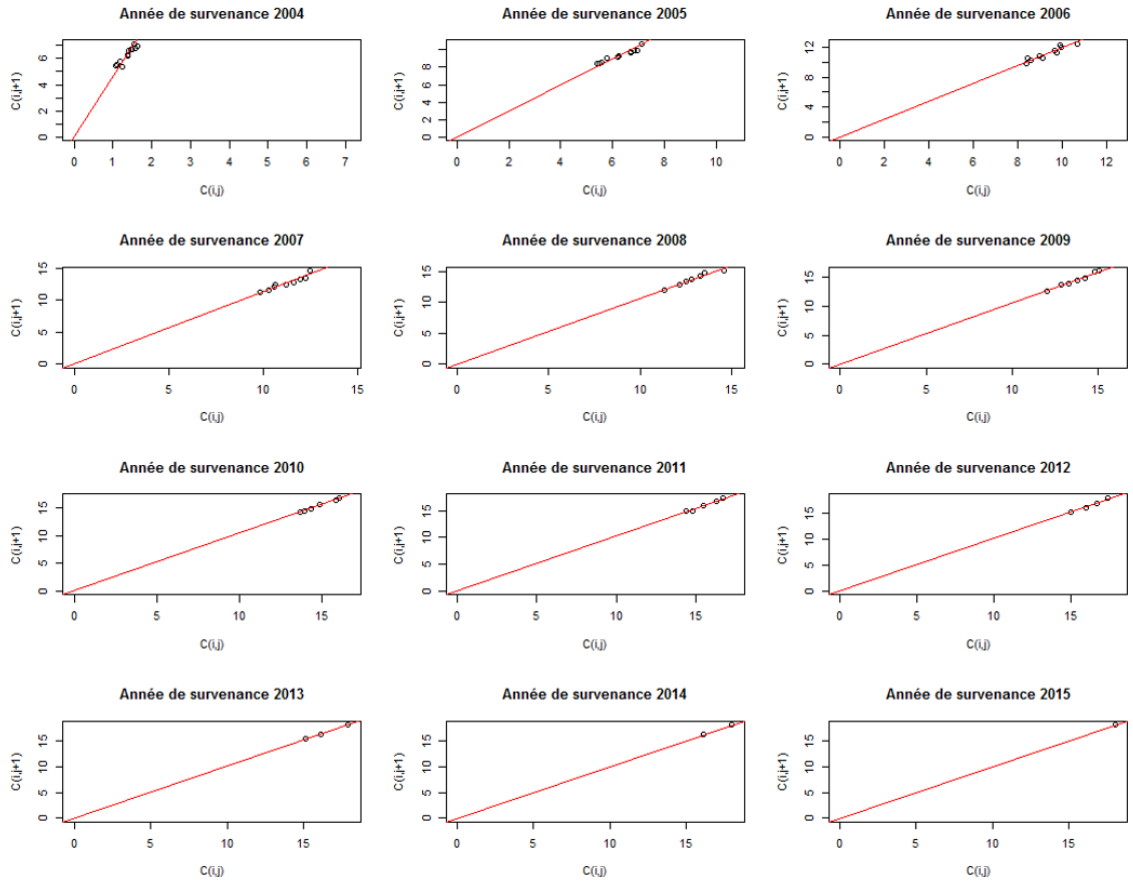


FIGURE 9.1 – Dépenses (en millions) - Vérification des hypothèses Chain-Ladder QQ-plot.

Tous les couples $(C_{i,j}, C_{i,j+1})_{i=0,\cdots,n-j-1}$ semblent bien être alignés sur une droite passant par l'origine, de coefficient directeur le coefficient de développement retenu par la méthode Chain-Ladder.

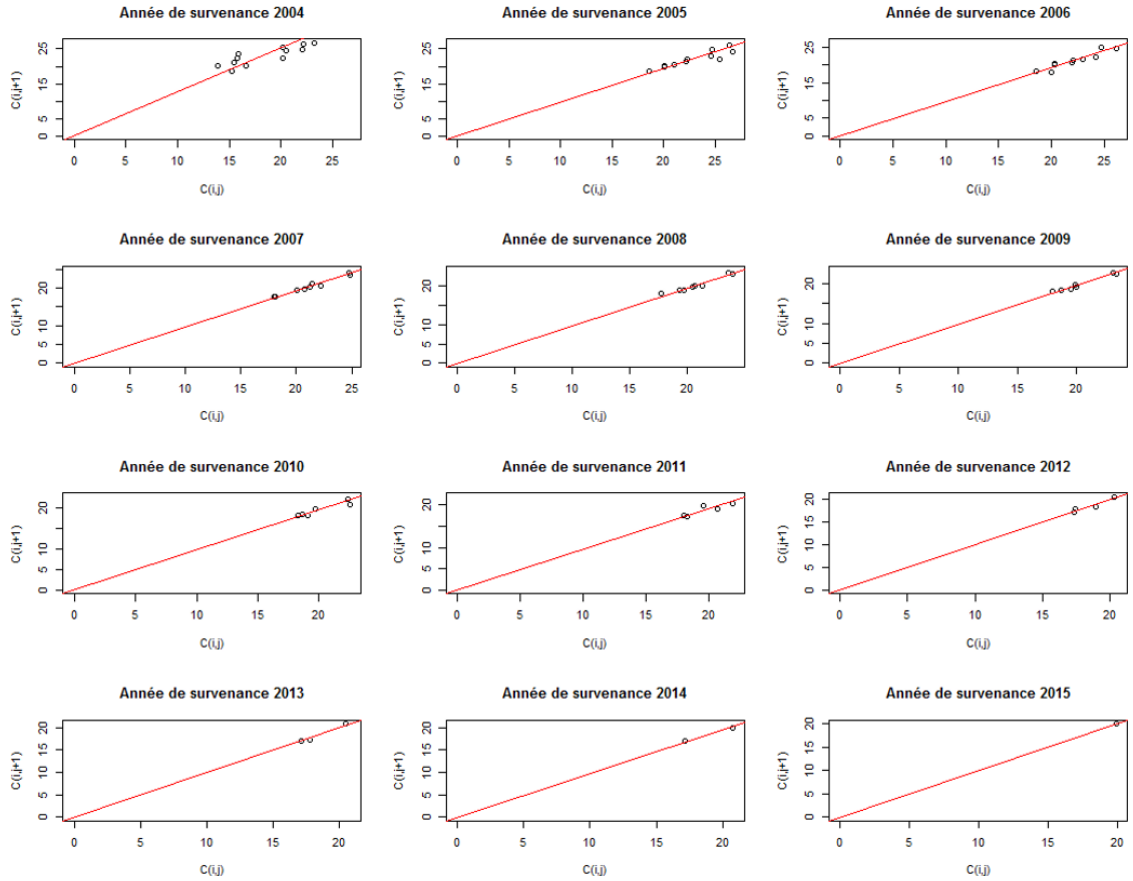


FIGURE 9.2 – Charges (en millions) - Vérification des hypothèses Chain-Ladder QQ-plot.

Tous les couples $(C_{i,j}, C_{i,j+1})_{i=0,\cdots,n-j-1}$ semblent bien être alignés sur une droite passant par l'origine, de coefficient directeur le coefficient de développement retenu par la méthode Chain-Ladder.

Chapitre 10

Variables explicatives

Variable	Description	Type de variable
Part_Respons	Part de responsabilité de l'assuré	Catégorielle
Convention	Convention de l'assuré	Catégorielle
Part_Respons_Conv	Part de responsabilité conventionnelle	Catégorielle
Code_Clause	Code clause événement	Catégorielle
Nature_Obs	Nature de l'obstacle	Catégorielle
Code_Proj	Présence ou non de projection de victime	Catégorielle
Type_Conducteur	Type de conducteur (assuré, conjoint...)	Catégorielle
Sexe	Sexe du conducteur	Catégorielle
Circ_Aggrav	Présence ou non de circonstance aggravante	Catégorielle
Sit_PRQ	Situation contrat PRQ	Catégorielle
Sit_PAC	Situation contrat PAC	Catégorielle
Presence_RCM	Présence ou non de garantie RC Matérielle	Catégorielle
Presence_DOM	Présence ou non de garantie Dommage	Catégorielle
Presence_REC	Présence ou non de recours	Catégorielle
Grave	Caractère grave du sinistre	Catégorielle
Clos_surv	Sinistre clos ou non à la survenance	Catégorielle
Nbvic_surv	Nombre de victimes à la survenance	Catégorielle
Para_surv	Présence ou non de victime paralysée à la survenance	Catégorielle
Poly_surv	Présence ou non de victime polytraumatisée à la survenance	Catégorielle
Tetra_surv	Présence ou non de victime tétraplégique à la survenance	Catégorielle
Trauma_surv	Présence ou non de victime traumatisée à la survenance	Catégorielle
IPP01_50_surv	Nombre de victimes avec un taux d'AIPP entre 1% et 50% à la survenance	Catégorielle
IPP51_60_surv	Nombre de victimes avec un taux d'AIPP entre 51% et 60% à la survenance	Catégorielle
IPP61_70_surv	Nombre de victimes avec un taux d'AIPP entre 61% et 70% à la survenance	Catégorielle
IPP71_80_surv	Nombre de victimes avec un taux d'AIPP entre 71% et 80% à la survenance	Catégorielle
IPP81_90_surv	Nombre de victimes avec un taux d'AIPP entre 81% et 90% à la survenance	Catégorielle
IPP91_100_surv	Nombre de victimes avec un taux d'AIPP entre 91% et 100% à la survenance	Catégorielle
Cour_Appel	Cour d'appel la plus proche	Catégorielle
Cat_Vehicule	Catégorie du véhicule	Catégorielle
devN_X	Incrément de dépense pour le développement X	Continue réelle
ctbN_X	Variation de charge pour le développement X	Continue réelle

FIGURE 10.1 – Liste des variables explicatives initiales

Chapitre 11

Modélisation utilisée

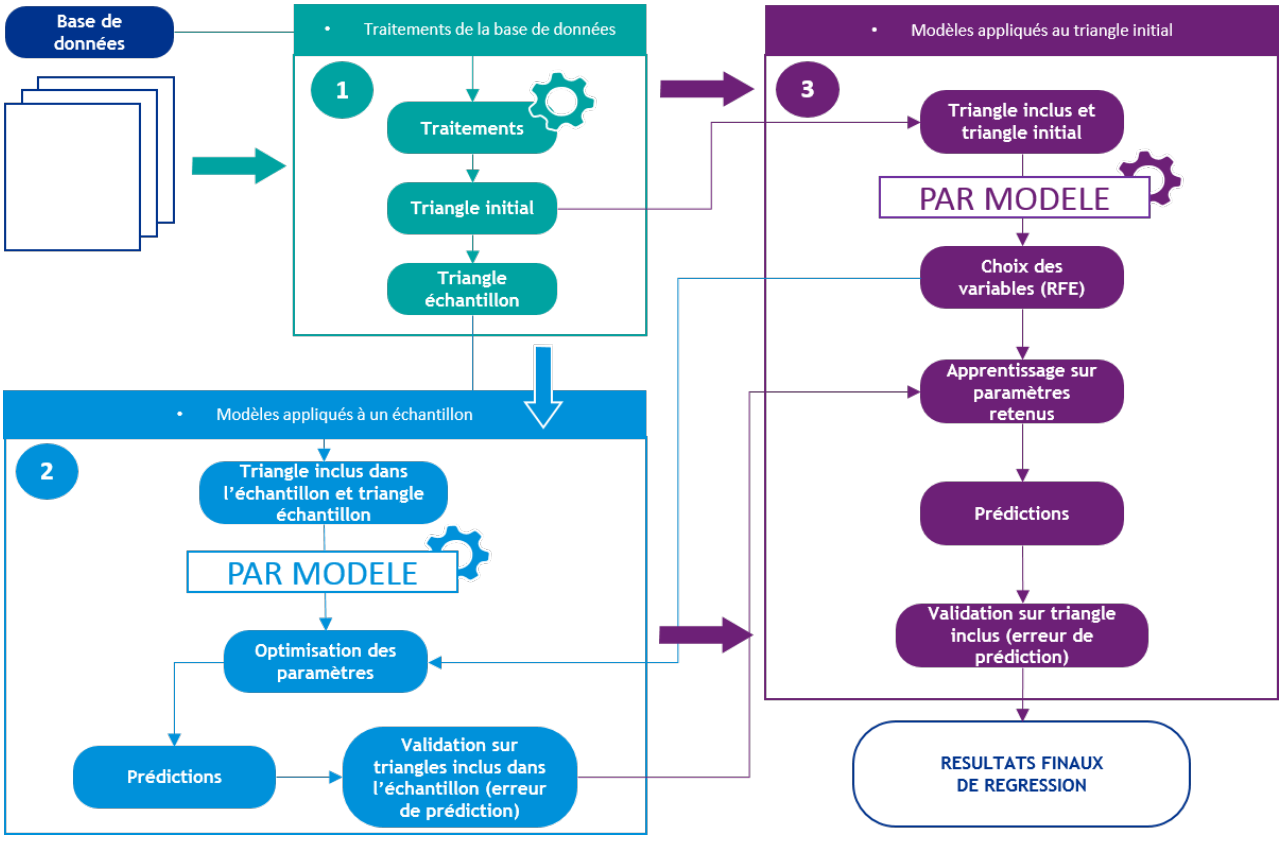


FIGURE 11.1 – Représentation de la modélisation

Triangle	Nombre de lignes	Périmètre	Construction	Utilité
Initial	255 900	2004 - 2016	Ensemble des données initiales	-
Echantillon (20%)	51 180	2004 - 2016	Echantillonage stratifié	Test de paramètres
Inclus	145 671	2004 - 2010	Périmètre 2004 - 2010	Validation des modèles Comparaison à des données réelles
Inclus dans l'échantillon (20%)	29 286	2004 - 2010	Périmètre 2004 - 2010 de l'échantillon stratifié	Validation des modèles Comparaison à des données réelles

FIGURE 11.2 – Synthèse des différents triangles

Chapitre 12

Paramètres fixes retenus

	Dépenses	Charges
Nombre d'arbres	500	500
Règle de segmentation	Variance	Variance
Nombre maximum d'individus dans les feuilles terminales	10	1

FIGURE 12.1 – Paramètres fixes retenus pour les forêts aléatoires

	Dépenses	Charges
Nombre d'itérations	500	500
Nombre de couches	2	2
Algorithme d'apprentissage	Rétropropagation classique du gradient de l'erreur	Rétropropagation classique du gradient de l'erreur
Fonction d'activation de la couche cachée	Tangente hyperbolique	Tangente hyperbolique
Taux d'apprentissage	0,1	0,25
Momentum	0	0,25

FIGURE 12.2 – Paramètres fixes retenus pour les réseaux de neurones

Table des figures

Bibliographie

[1] Katrien Antonio and Richard Plat. Micro-level stochastic loss reserving for general insurance. *Scandinavian Actuarial Journal*, 2014(7) :649–669, 2014.

[2] Elja Arjas. The claims reserving problem in non-life insurance : Some structural ideas. *Astin Bulletin*, 19(2) :139–152, 1989.

[3] Eugène Astesan. *Université de Paris. Faculté de droit. Les Réserves techniques des sociétés d'assurances contre les accidents d'automobiles, thèse pour le doctorat... présentée... par Eugène Astesan...* R. Pichon et R. Durand-Auzias, 1938.

[4] Gérard Biau, Aurélie Fischer, Benjamin Guedj, and James Malley. Cobra : a nonlinear aggregation strategy. *arXiv preprint arXiv :1303.2236*, 2013.

[5] Ronald L Bornhuetter and Ronald E Ferguson. The actuary and ibnr. In *Proceedings of the casualty actuarial society*, volume 59, pages 181–195, 1972.

[6] Leo Breiman. Bagging predictors. *Machine learning*, 24(2) :123–140, 1996.

[7] Leo Breiman. Random forests. *Machine learning*, 45(1) :5–32, 2001.

[8] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[9] David S Broomhead and David Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, DTIC Document, 1988.

[10] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4) :303–314, 1989.

[11] Damien Drieskens, Marc Henry, Jean-François Walhin, and Jürgen Wielandts. Stochastic projection for large individual losses. *Scandinavian Actuarial Journal*, 2012(1) :1–39, 2012.

[12] PD England and RJ Verrall. Stochastic claims reserving in general insurance. http, 2007.

[13] Peter D England and Richard J Verrall. Stochastic claims reserving in general insurance. *British Actuarial Journal*, 8(03) :443–518, 2002.

[14] Scott E Fahlman. An empirical study of learning speed in back-propagation networks. 1988.

[15] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *icml*, volume 96, pages 148–156, 1996.

[16] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1) :1, 2010.

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

[18] Ken-Ichi Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural networks*, 2(3) :183–192, 1989.

[19] G David Garson. Interpreting neural-network connection weights. *AI expert*, 6(4) :46–51, 1991.

[20] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1) :3–42, 2006.

[21] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[22] Svend Haastrup and Elja Arjas. Claims reserving in continuous time ; a nonparametric bayesian approach. *ASTIN Bulletin : The Journal of the IAA*, 26(2) :139–164, 1996.

[23] Leigh Joseph Halliwell. Chain-ladder bias : its reason and meaning. *CAS, Variance*, pages 214–247, 2007.

[24] Bor Harej, Roman Gachter, and Salma Jamal. Individual claim development with machine learning. `http://www.actuaries.org/panama2017/docs/papers/1a_ASTIN_Paper_Harej.pdf`, 2017.

[25] Pierre Henin. Un modele de provisionnement ligne a ligne en assurance responsabilite civile. `http://www.ressources-actuarielles.net/EXT/ISFA/1226-02.nsf/0/3e8abe833b6ac5eec12581430021ec76/$FILE/HENIN.pdf`.

[26] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5) :359–366, 1989.

[27] Jinlong Huang, Chunjuan Qiu, and Xianyi Wu. Stochastic loss reserving in discrete time : Individual vs. aggregate data models. *Communications in Statistics-Theory and Methods*, 44(10) :2180–2206, 2015.

[28] Christian Roholte Larsen. An individual claims reserving model. *ASTIN Bulletin : The Journal of the IAA*, 37(1) :113–132, 2007.

[29] Olivier Lopez, Xavier Milhaud, Pierre-E Thérond, et al. Tree-based censored regression with applications in insurance. *Electronic journal of statistics*, 10(2) :2685–2716, 2016.

[30] Thomas Mack. Distribution-free calculation of the standard error of chain ladder reserve estimates. *Astin bulletin*, 23(02) :213–225, 1993.

[31] James D Malley, Jochen Kruppa, Abhijit Dasgupta, Karen G Malley, and Andreas Ziegler. Probability machines : consistent probability estimation using nonparametric learning machines. *Methods of Information in Medicine*, 51(1) :74, 2012.

[32] P McCullagh and JA Nelder. Generalised linear modelling. *Chapman and Hall, London. Negro, JJ & Hiraldo, F.(1992) Sex ratios in broods of the lesser kestrel Falco naumanni. Ibis*, 134 :190–191, 1983.

[33] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4) :115–133, 1943.

[34] Martin Fodslette Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural networks*, 6(4) :525–533, 1993.

[35] John Ashworth Nelder and R Jacob Baker. *Generalized linear models*. Wiley Online Library, 1972.

[36] Ragnar Norberg. Prediction of outstanding liabilities in non-life insurance. *Astin Bulletin*, 23(01) :95–115, 1993.

[37] Ragnar Norberg. Prediction of outstanding liabilities ii. model variations and extensions. *Astin Bulletin*, 29(01) :5–25, 1999.

[38] Christian Partrat, E Lecoeur, JM Nessi, E Nisipasu, and O Reiz. Provisionnement technique en assurance non-vie. *Economica*, 2007.

[39] Mathieu Pigeon, Katrien Antonio, and Michel Denuit. Individual loss reserving with the multivariate skew normal framework. *Astin Bulletin*, 43(03) :399–428, 2013.

[40] Swiss Re. Non-life claims reserving : Improving on a strategic challenge. *Sigma*, 2 :1–39, 2008.

[41] Frank Rosenblatt. The perceptron : A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6) :386, 1958.

[42] Stig Rosenlund et al. Bootstrapping individual claim histories. *ASTIN Bulletin-Actuarial Studies in non LifeInsurance*, 42(1) :291, 2012.

[43] Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2) :197–227, 1990.

[44] Magda Schiegl. A model study about the applicability of the chain ladder method. *Scandinavian Actuarial Journal*, 2015(6) :482–499, 2015.

[45] Greg Taylor, Gráinne McGuire, and James Sullivan. Individual claim loss reserving conditioned by case estimates. *Annals of Actuarial Science*, 3(1-2) :215–256, 2008.

[46] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[47] Andrei Nikolaevich Tikhonov. On the solution of ill-posed problems and the method of regularization. In *Doklady Akademii Nauk*, volume 151, pages 501–504. Russian Academy of Sciences, 1963.

[48] G. Worthington. Developpement d'une methodologie de provisionnement sinistre par sinistre en assurance non-vie et articulation avec les methodes agregees. `http://www.ressources-actuarielles.net/EXT/ISFA/1226-02.nsf/0/ebf01a77f3db7044c1257dda002dac24/$FILE/WORTHINGTON.002.pdf/WORTHINGTON.pdf`.

[49] Marvin N Wright, Theresa Dankowski, and Andreas Ziegler. Random forests for survival analysis using maximally selected rank statistics. *arXiv preprint arXiv :1605.03391*, 2016.

[50] Mario V Wuthrich. Machine learning in individual claims reserving. *Browser Download This Paper*, 2016.

[51] Mario V Wuthrich. Neural networks applied to chain-ladder reserving. *To be published*, 2017.

[52] Xiao Bing Zhao, Xian Zhou, and Jing Long Wang. Semiparametric model for prediction of individual claim loss reserving. *Insurance : Mathematics and Economics*, 45(1) :1–8, 2009.

[53] XiaoBing Zhao and Xian Zhou. Applying copula models to individual claim loss reserving methods. *Insurance : Mathematics and Economics*, 46(2) :290–299, 2010.

[54] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society : Series B (Statistical Methodology)*, 67(2) :301–320, 2005.