Canadian Institute of Actuaries | Institut canadien des actuaires

Document de recherche

LRMoE.jl : Un progiciel qui modélise les sinistres d'assurance au moyen d'un modèle de régression de mélange d'experts

Spark C. Tseung, Andrei L. Badescu,
Tsz Chai Fung et X. Sheldon Lin

Février 2022

Document rp222019

This document is available in English

Points saillants des documents

LRMoE.jl : un progiciel qui modélise les sinistres d'assurance au moyen d'un modèle de régression de mélange d'experts

et

LRMoE : un progiciel R pour la modélisation flexible des sinistres d'assurance au moyen d'un modèle de régression de mélange d'experts

Depuis quelques années, nous, chercheurs à l'Université de Toronto (Andrei L. Badescu, X. Sheldon Lin et des doctorants actuels ou anciens) travaillons à des projets de recherche sur le calcul des réserves et la tarification en assurances IARD. Notre objectif est de concevoir des technologies nouvelles et réalisables et des progiciels prêts à utiliser à l'intention des actuaires exerçant dans ce domaine. Ce projet est l'un des résultats de ces efforts. Son financement a été rendu possible grâce à une subvention pour la recherche universitaire de l'Institut canadien des actuaires (ICA), que nous tenons à remercier.

Dans le cadre de ce projet, nous introduisons un nouveau logiciel statistique libre, conçu sur mesure pour les applications actuarielles, qui permet aux praticiens en actuariat de modéliser et d'analyser la fréquence et la sévérité des sinistres d'assurance au moyen d'un modèle de régression multivarié et non linéaire. Le modèle est très flexible : il peut s'adapter à n'importe quel type de base de données positives et prendre en compte la structure de dépendance que laissent entendre les données, en plus d'être réalisable statistiquement.

Nous avons produit deux documents avec progiciels correspondants : l'un donne une courte description d'un progiciel R, et l'autre, d'un progiciel Julia. R est connu de nombreux actuaires, tandis que Julia est un langage de programmation très efficace qui est communément utilisé par les communautés de l'apprentissage automatique et de l'informatique. Le progiciel Julia s'exécute environ quatre fois plus rapidement que le progiciel R. Ces progiciels peuvent être téléchargés respectivement aux adresses https://github.com/sparktseung/LRMoE.jl et https://github.com/sparktseung/LRMoE. Ils offrent des caractéristiques distinctives qu'on ne peut exploiter avec les progiciels existants, les principales étant une prise en charge plus large des distributions de fréquence et de sévérité et leur version gonflée à zéro, l'estimation de paramètres en présence de censure et/ou de troncature dans les données, et un ensemble de fonctions permettant le calcul des réserves et la tarification en assurance. De plus, les logiciels comportent plusieurs fonctions d'évaluation et de visualisation de modèles qui facilitent la tâche des utilisateurs lorsqu'ils analysent la performance du modèle ajusté et interprètent le modèle dans des contextes d'assurance.

On trouvera de plus amples informations sur le développement des modèles et des méthodes qui sous-tendent nos progiciels dans les ouvrages suivants :

- Fung, T.C., Badescu, A. et X.S. Lin. « A class of mixture of experts models for general insurance: Application to correlated claim frequencies », *ASTIN Bulletin*, **vol. 49**, n° 3, 2019, pp. 647–688.

- Fung, T.C., Badescu, A. et X.S. Lin. « A class of mixture of experts models for general insurance: Theoretical developments », *Insurance: Mathematics and Economics*, **vol. 89**, 2019, p. 111–127.
- Fung, T.C., Badescu, A. et X.S. Lin. « A new class of severity regression models with an application to IBNR prediction », *North American Actuarial Journal*, **vol. 25**, n° 2, 2020, p. 1–26.
- Fung, T.C., Badescu, A. et X.S. Lin. « Fitting censored and truncated regression data using the mixture of experts models », 2021. Sur SSRN : https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3740061.

Ces ouvrages sont disponibles sur demande. Pour toute question ou commentaire, communiquez avec nous.

Andrei L. Badescu et X. Sheldon Lin

Université de Toronto

badescu@utstat.utoronto.ca, sheldon.lin@utoronto.ca

LRMoE.jl : un progiciel qui modélise les sinistres d'assurance au moyen d'un modèle de régression de mélange d'experts

Résumé

Dans le présent document, nous présentons un nouveau progiciel Julia, **LRMoE**, qui est un logiciel statistique conçu sur mesure pour les applications actuarielles et qui permet aux chercheurs et aux praticiens en actuariat de modéliser et d'analyser la fréquence et la sévérité des sinistres d'assurance au moyen du modèle de régression logit pondéré réduit de mélange d'experts (*Logit-weighted Reduced Mixture of Experts*). **LRMoE** offre plusieurs nouvelles fonctionnalités distinctives utiles pour diverses applications actuarielles et qu'on ne peut généralement pas répliquer au moyen des progiciels existants avec les modèles par mélange. Parmi les principales fonctionnalités, mentionnons une prise en charge plus large des distributions de fréquence et de sévérité et de leur version gonflée à zéro, la flexibilité de pouvoir varier les classes de distributions pour les différentes composantes, l'estimation de paramètres en présence de censure et/ou de troncature dans les données, et un ensemble de fonctions de calcul des réserves et de tarification en assurance. De plus, le logiciel comporte plusieurs fonctions d'évaluation et de visualisation de modèles qui facilitent la tâche des utilisateurs lorsqu'ils analysent la performance du modèle ajusté et interprètent le modèle dans des contextes d'assurance.

Mots-clés : Analyse de régression multivariée, censure et troncature, algorithme espérance conditionnelle-maximisation, calcul des réserves et tarification en assurance, Julia.

1. Introduction

Le LRMoE est un modèle de régression flexible créé par Fung et coll. (2019b) et il est considéré comme la version régression d'un modèle de mélange fini avec des poids (appelés *fonction de contrôle*) qui dépendent des covariables. Nous pouvons interpréter le LRMoE comme un outil de classification des titulaires de police en divers sous-groupes dont les probabilités diffèrent. Dépendantes de la composante sous-groupe à laquelle chaque titulaire de police est affecté, les propriétés distributionnelles de la fréquence ou de la sévérité des sinistres sont régies par les fonctions de la composante mélange (appelées *fonction d'experts*). La flexibilité, la parcimonie et la maniabilité mathématique du modèle sont justifiées (voir Fung et coll., 2019b), ce qui démontre le solide fondement théorique du LRMoE dans le contexte général de la modélisation des sinistres d'assurance. Pour certains choix de fonctions d'experts, Fung et coll. (2019a) et Fung et coll. (2020) ont construit des algorithmes espérance-maximisation conditionnelle (EMC) permettant un ajustement efficace des modèles de fréquence et de sévérité et ils ont ainsi démontré l'utilité éventuelle du LRMoE pour le calcul des réserves et la tarification en assurance.

Bien que le progiciel **flexmix** existant en R (Leisch 2004, et Grün et Leisch 2008) puisse effectuer une estimation des paramètres pour certains cas particuliers du LRMoE, il n'offre que des choix limités de fonctions d'experts (Poisson, gaussienne, gamma et binomiale) pour l'ajustement du modèle. Miljkovic et Grün (2016) ont profité de la souplesse qu'offre le progiciel flexmix afin de proposer de nouveaux modèles de mélanges intégrant d'autres fonctions d'experts (comme log-normale, Weibull et Burr), mais les utilisateurs sont toujours contraints de choisir une seule distribution paramétrique pour toutes les composantes.

Dans le présent document, nous présentons un nouveau progiciel en langage Julia, **LRMoE**, qui est conçu sur mesure pour les applications actuarielles et qui permet aux chercheurs et aux praticiens en actuariat de modéliser et d'analyser la fréquence et la sévérité des sinistres d'assurance au moyen du modèle LRMoE. Le progiciel offre plusieurs nouvelles fonctionnalités distinctives utiles pour diverses applications actuarielles et qu'on ne peut généralement pas répliquer au moyen des progiciels existants, dont les suivantes :

- Ajustement rapide : Compte tenu du besoin croissant d'analyser de gros ensembles de données d'assurance comportant des centaines de milliers d'observations, il est essentiel que le progiciel puisse faire l'ajustement des modèles dans un délai raisonnable. Comparativement aux langages traditionnels comme R, la mise en œuvre dans Julia réduit considérablement le temps d'exécution, ce qui permet aux utilisateurs d'obtenir et d'analyser les résultats beaucoup plus rapidement (voir la section 4.1 pour des comparaisons).
- Prise en charge plus large des distributions de fréquence et de sévérité : en plus des distributions de sévérité couvertes par Miljkovic et Grün (2016), le progiciel prend également en charge d'autres fonctions d'experts pour la fréquence qui sont importantes pour la modélisation actuarielle des sinistres, notamment la distribution binomiale négative et la distribution de comptage gamma.
- Distributions gonflées à zéro : Souvent, les actuaires s'intéressent plus à l'analyse des sinistres agrégés de chaque titulaire de police qu'aux analyses séparées de la fréquence et de la sévérité. Dans cette situation, il est courant d'observer trop de zéros, ce qui motive l'utilisation de fonctions d'experts gonflées à zéro dans le LRMoE, et c'est ce qu'offre le progiciel. À noter que pour que le calcul du LRMoE gonflé à zéros soit efficace, il faut définir une variable latente additionnelle (voir la section 2.2 pour plus de détails), ce qui diminue la souplesse qu'offre le progiciel **flexmix**.
- Extensibilité du progiciel : En plus de prendre en charge une grande diversité de distributions, notre progiciel permet aux utilisateurs de définir leurs fonctions d'experts en suivant les instructions simples de la documentation du progiciel. Par conséquent, le progiciel peut être utile non seulement aux actuaires, mais aussi au monde de la recherche en général et à la résolution de problèmes pratiques.
- Possibilités de varier les classes de distributions pour les différentes composantes du mélange : Les données d'assurance peuvent présenter des comportements incohérents pour les petites et les grandes valeurs (queue de la distribution). Cela doit être pris en compte par l'utilisation de distributions différentes. L'une des

approches consiste à choisir deux distributions et à les combiner à l'aide d'une méthode de dépassement de seuil (voir, par exemple, Lee et coll. 2012 et Scollnik et Sun 2012). Une autre approche consiste à recourir à un modèle de mélange fini fondé sur des distributions différentes (voir, par exemple, Blostein et Miljkovic 2019). Le progiciel **LRMoE** s'apparente à la dernière approche, les utilisateurs pouvant sélectionner différentes fonctions d'experts pour les différentes composantes du mélange, ce qui permet une modélisation plus flexible et plus réaliste des données.

- Données incomplètes : Dans de nombreuses applications actuarielles, notamment en réassurance, en gestion du risque opérationnel et en tarification avec franchise et en modélisation des réserves, on observe souvent des données censurées et tronquées qu'il faut pouvoir traiter. La censure et la troncature du LRMoE sont introduites par Fung et coll. (2021) dans le cas où les fonctions d'experts sont des distributions gamma univariées. Le nouveau progiciel élimine cette restriction en permettant aux utilisateurs de prendre en compte des données multivariées qui ont été censurées et/ou tronquées de façon aléatoire à l'aide d'un grand choix de fonctions d'experts.
- Sélection et visualisation du modèle : En plus de la fonction d'ajustement du modèle, le nouveau progiciel offre plusieurs fonctions d'évaluation du modèle (critère d'information d'Akaike [CIA], critère d'information bayésien [CIB]) et de visualisation du modèle (p. ex., probabilités des classes latentes, influence des covariables) qui facilitent la tâche des utilisateurs lorsqu'ils analysent la performance du modèle et interprètent le modèle dans des contextes d'assurance.
- Calcul des réserves et tarification en assurance : Le progiciel comprend également plusieurs fonctions pour le calcul des réserves et la tarification (p. ex., la moyenne, la variance, la valeur au risque [VaR], l'espérance conditionnelle unilatérale [ECU]), qui permettent aux actuaires de procéder simultanément à la tarification d'un grand nombre de contrats d'assurance ayant des caractéristiques différentes, à l'aide de plusieurs principes de primes.

Le présent document est organisé comme suit. À la section 2 , nous examinerons le modèle LRMoE et l'estimation des paramètres au moyen de l'algorithme EMC. À la section 3, nous utiliserons un ensemble de données simulées pour illustrer la procédure de base d'ajustement du progiciel **LRMoE**. À la section 4, nous traiterons d'autres fonctionnalités du progiciel, comme l'initialisation des paramètres, la visualisation du modèle et la fonction de calcul des primes, qui sont illustrées à l'aide d'un ensemble de données d'assurance automobile recueilli en France. Enfin, à la section 5, nous discuterons de certains points. Par souci de concision, nous ne présenterons que les lignes de code les plus pertinentes pour notre nouveau progiciel. On trouvera le code source, la documentation du progiciel et le code de réplication complet pour tous les exemples du présent document aux adresses https://github.com/sparktseung/LRMoE.jl et https://sparktseung.github.io/LRMoE.jl/dev/. Par ailleurs, à l'intention des utilisateurs souhaitant plutôt exécuter le progiciel dans R, nous avons créé un progiciel R correspondant (accéléré par **Rcpp**) pour ajuster LRMoE avec des fonctionnalités similaires. Nous invitons ces lecteurs à consulter Tseung et coll. (2021) pour obtenir une courte description, et https://github.com/sparktseung/LRMoE pour le code et la documentation.

2. Modèle LRMoE et estimation des paramètres

Nous présentons ici un aperçu du modèle LRMoE proposé dans Fung et coll. (2019b) et nous examinons l'algorithme EMC pour l'estimation des paramètres. Par souci de concision, nous supposerons aux sections 2.1 et 2.2 que toutes les variables de réponse (fréquence ou sévérité des sinistres) sont observées exactement. À la section 2.3, nous traiterons de la troncature et de la censure des données pour le modèle LRMoE.

2.1 Modèle logit réduit pondéré de mélange d'experts (LRMoE)

Soit $\boldsymbol{x}_i = (x_{i0}, x_{i1}, \ldots, x_{iP})^T$ le vecteur de covariables de longueur (P +1) pour le titulaire de police i (i = 1, 2, …, n) avec terme constant $x_{i0} = 1$, et $\boldsymbol{y}_i = (y_{i1}, y_{i2}, \ldots, y_{iD})^T$ le vecteur de longueur D des variables de réponse, qui peuvent être la fréquence ou la sévérité des sinistres. Soit $\boldsymbol{X} = (x_1, x_2, \ldots, x_n)^T{}^T$ et $\boldsymbol{Y} = (y_1, y_2, \ldots, y_n)^T$ toutes les covariables et réponses pour un groupe de n titulaires.

D'après les covariables, le titulaire i est classé dans l'une des g classes de risque latentes par une fonction de contrôle logit

$$\pi_j(\boldsymbol{x}_i; \boldsymbol{\alpha}_j) = \frac{exp(\boldsymbol{\alpha}_j^T \boldsymbol{x}_i)}{\sum_{j'=1}^{g} exp(\boldsymbol{\alpha}_{j'}^T \boldsymbol{x}_i)}, \quad j = 1, 2, \ldots, g, \tag{1}$$

où $\boldsymbol{\alpha}_j = (\alpha_{j0}, \alpha_{j1}, \ldots, \alpha_{jP})^T$ est un vecteur de coefficients de régression pour la classe latente j.

Étant donné l'attribution de la classe latente $j \in \{1, 2, \ldots, g\}$, les variables $\boldsymbol{y}_i = (y_{i1}, y_{i2}, \ldots, y_{iD})^T$ de réponse sont conditionnellement indépendantes. Pour $d = 1, \ldots, D$ et $i = 1, \ldots, n$, la fonction de densité de probabilité de la d^e dimension (ou fonction de masse de probabilité) de y_{id} est donnée par

$$g_{jd}(y_{id}; \delta_{jd}, \psi_{jd}) = \delta_{jd} \mathbf{1}\{y_{id} = 0\} + (1 - \delta_{jd}) f_{jd}(y_{id}; \psi_{jd}) \tag{2}$$

où $\delta_{jd} \in [0, 1]$ est un paramètre représentant la probabilité d'attribuer une observation à une composante avec surreprésentation de zéros (c.-à-d. une masse de probabilité en zéro), $\mathbf{1}\{y_{id} = 0\}$ est la fonction indicatrice et $f_{jd}(y_{id}; \psi_{jd})$ est la densité d'une distribution couramment utilisée avec paramètres ψ_{jd} pour modéliser les sinistres d'assurance. Le tableau 1 donne la liste des distributions paramétriques prises en charge par le progiciel **LRMoE**. À noter qu'une masse de probabilité δ_{jd} en zéro permet une modélisation plus réaliste des données d'assurance qui comportent une surreprésentation de zéros. Comme convention de nommage, nous désignerons par g_{jd} la fonction de distribution de composantes/experte, et par f_{jd} sa partie positive, bien que les distributions de fréquence des sinistres (p. ex., Poisson) aient également une masse de probabilité en zéro.

Tableau 1 : Distributions prises en charge par le LRMoE

Racine	Distribution	$f_{jd}(y)$	Paramètres
Gamma	Gamma	$\frac{1}{\theta^m \Gamma(m)} y^{m-1} e^{-y/\theta}$	$m > 0,\ \theta > 0$
Lnorm	Log-normale	$\frac{1}{y\sigma\sqrt{2\pi}} \exp\left[-\frac{(\log y - \mu)^2}{2\sigma^2}\right]$	$\mu \in \mathbb{R},\ \sigma > 0$
Invgauss	Gaussienne inverse	$\sqrt{\frac{\lambda}{2\pi y^3}} \exp\left[-\frac{\lambda(y-\mu)^2}{2\mu^2 y}\right]$	$\mu > 0,\ \lambda > 0$
Weibull	Weibull	$\frac{k}{\lambda}\left(\frac{y}{\lambda}\right)^{k-1} \exp\left[-\left(\frac{y}{\lambda}\right)^k\right]$	$k > 0,\ \lambda > 0$
Burr	Burr	$e^{-\lambda}\frac{\lambda^y}{y!}$	$k > 0,\ c > 0,\ \lambda > 0$
Poisson	Poisson	$\binom{y+n-1}{n-1} p^n (1-p)^y$	$\lambda > 0$
nbinom	Binomiale négative	$\int_0^{ms} \frac{1}{\Gamma(ys)} u^{ys-1} \exp\{-u\} du$	$n > 0,\ 0 < p < 1$
gammacount	Comptage gamma	$-\int_0^{ms} \frac{1}{\Gamma((y+1)s)} u^{(y+1)s-1} \exp\{-u\}$ du	$m > 0,\ s > 0$
ZI-root	Toutes les distributions précédentes	$g_{jd} = \delta_{jd}\mathbf{1}\{y = 0\} + (1 - \delta_{jd}) f_{jd}$	$0 < \delta_{jd} < 1$

Sous LRMoE, la fonction de densité de probabilité conditionnelle (ou fonction de masse de probabilité) de 5 y_i étant donné les covariables x_i est donnée par

$$h(\boldsymbol{y}_i; \boldsymbol{x}_i, \boldsymbol{\alpha}, \boldsymbol{\delta}, \boldsymbol{\Psi}) = \sum_{j=1}^{g} \left[\pi_j(\boldsymbol{x}_i; \boldsymbol{\alpha}_j) \times \prod_{d=1}^{D} g_{jd}(y_{id}; \delta_{jd}, \boldsymbol{\psi}_{jd}) \right] \qquad (3)$$

où $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_g)^T$ est une matrice $g \times (P+1)$ de poids de mélange avec $\boldsymbol{\alpha}_g = (0, 0, \ldots, 0)^T$ qui représente la classe par défaut, $\boldsymbol{\delta} = (\delta_{jd})_{1 \le j \le g, 1 \le d \le D}$ est une matrice $(g \times D)$ de masses de probabilité en zéro par composante et par dimension, et $\boldsymbol{\Psi} = \{\boldsymbol{\psi}_{jd} : 1 \le j \le g, 1 \le d \le D\}$ est une liste de paramètres de la partie positive f_{jd} par composante et par dimension. À noter que $\boldsymbol{\alpha}_g = (0, 0, \ldots, 0)^T$ rend le modèle identifiable, c'est-à-dire qu'il existe une application injective entre les distributions de régression et les paramètres (voir Jiang et Tanner 1999, et Fung et coll. 2019a). À noter que le nombre total de paramètres du modèle est donné par $(g-1) \times (P+1) + g \times D + \sum_{j=1}^{g} \sum_{d=1}^{D} \#\{\boldsymbol{\psi}_{jd}\}$, où $\#\{\boldsymbol{\psi}_{jd}\}$ est la longueur de $\boldsymbol{\psi}_{jd}$.

Pour un groupe de *n* titulaires de police, la fonction de vraisemblance est donnée par

$$L(\boldsymbol{\alpha}, \boldsymbol{\delta}, \boldsymbol{\Psi}; \boldsymbol{X}, \boldsymbol{Y}) = \prod_{i=1}^{n} h(\boldsymbol{y}_i; \boldsymbol{x}_i, \boldsymbol{\alpha}, \boldsymbol{\delta}, \boldsymbol{\Psi}) = \prod_{i=1}^{n} \left\{ \sum_{j=1}^{g} \left[\pi_j(\boldsymbol{x}_i; \boldsymbol{\alpha}_j) \times \prod_{d=1}^{D} g_{jd}(y_{id}; \delta_{jd}, \psi_{jd}) \right] \right\}. \tag{4}$$

2.2 Estimation des paramètres

Pour l'estimation des paramètres des modèles à mélange fini, on utilise couramment l'algorithme espérance-maximisation (EM) (voir, par exemple, Dempster et coll. 1977 et McLachlan et Peel 2004). Toutefois, pour le modèle LRMoE, l'étape M nécessite la maximisation d'une fonction non concave sur tous les éléments de $\boldsymbol{\alpha}$. Fung et coll. (2019a) utilise donc l'algorithme EMC (Meng et Rubin 1993) qui divise l'étape M en plusieurs sous-étapes. Nous décrivons ci-dessous l'algorithme EMC intégré au progiciel **LRMoE**.

Nous désignons par $\boldsymbol{\Phi} = (\boldsymbol{\alpha}, \boldsymbol{\delta}, \boldsymbol{\Psi})$ les paramètres à estimer. Pour $i = 1, 2, \ldots, n$, nous introduisons un vecteur aléatoire latent $\boldsymbol{Z}_i = (Z_{i1}, Z_{i2}, \ldots, Z_{ig})^T$, où $Z_{ij} = 1$ si $\boldsymbol{y}_i$ provient de la *j*e distribution des composantes et $Z_{ij} = 0$ sinon. Pour $d = 1, 2, \ldots, D$, nous posons $Z_{ij} = Z_{ijd0} + Z_{ijd1}$, où $Z_{ijd0} = 0$ et $Z_{ijd1} = 1$ si la *d*e dimension de $\boldsymbol{y}_i$ provient de la partie positive f_{jd} de la *j*e composante, et $Z_{ijd0} = 1$ et $Z_{ijd1} = 0$ si elle provient de la surreprésentation de zéros δ_{jd}. La fonction de log-vraisemblance des données complètes est donnée par

$$\begin{aligned} l^{\text{com}}(\boldsymbol{\Phi}; \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}) &= \sum_{i=1}^{n} \sum_{j=1}^{g} Z_{ij} \left\{ \log \pi_j(\boldsymbol{x}_i; \boldsymbol{\alpha}_j) + \sum_{d=1}^{D} \log g_{jd}(y_{id}; \delta_{jd}, \psi_{jd}) \right\} \\ &= \sum_{i=1}^{n} \sum_{j=1}^{g} Z_{ij} \log \pi_j(\boldsymbol{x}_i; \boldsymbol{\alpha}_j) + \sum_{i=1}^{n} \sum_{j=1}^{g} \sum_{d=1}^{D} \{ Z_{ijd0} \log \delta_{jd} + Z_{ijd1} \log(1 - \delta_{jd}) \} \\ &\quad + \sum_{i=1}^{n} \sum_{j=1}^{g} \sum_{d=1}^{D} Z_{ijd1} \log f_{jd}(y_{id}; \psi_{jd}). \end{aligned}$$

Pour chaque $i = 1, 2, \ldots, n$, le vecteur aléatoire Zi suit une distribution multinomiale avec paramètre de comptage égal à 1 et probabilités $(\pi_1(\boldsymbol{x}_i; \boldsymbol{\alpha}_1), \pi_2(\boldsymbol{x}_i; \boldsymbol{\alpha}_2), \ldots, \pi_g(\boldsymbol{x}_i; \boldsymbol{\alpha}_g))$. Sachant que $Z_{ij} = 1$, la distribution conditionnelle de Z_{ijd0} est une Bernoulli avec probabilité δ_{jd}. Par conséquent, à la *t*e itération, les espérances a posteriori de Z_{ij}, Z_{ijd0} et de Z_{ijd1} sont

$$
\begin{aligned}
z_{ij}^{(t)} &= \mathsf{E}\{Z_{ij}|\boldsymbol{\Phi}^{(t-1)}, \boldsymbol{X}, \boldsymbol{Y}\} = \mathsf{P}\{Z_{ij} = 1|\boldsymbol{\Phi}^{(t-1)}, \boldsymbol{X}, \boldsymbol{Y}\} \\
&= \frac{\pi_j(\boldsymbol{x}_i; \boldsymbol{\alpha}_j^{(t-1)}) \times \prod_{d=1}^{D} g_{jd}(y_{id}; \delta_{jd}^{(t-1)}, \boldsymbol{\psi}_{jd}^{(t-1)})}{\sum_{j'=1}^{g} \pi_{j'}(\boldsymbol{x}_i; \boldsymbol{\alpha}_{j'}^{(t-1)}) \times \prod_{d=1}^{D} g_{j'd}(y_{id}; \delta_{j'd}^{(t-1)}, \boldsymbol{\psi}_{j'd}^{(t-1)})},
\end{aligned} \tag{6}
$$

$$
\begin{aligned}
z_{ijd0}^{(t)} &= \mathsf{E}\{Z_{ijd0}|\boldsymbol{\Phi}^{(t-1)}, \boldsymbol{X}, \boldsymbol{Y}\} \\
&= \mathsf{P}\{Z_{ijd0} = 1|\boldsymbol{\Phi}^{(t-1)}, \boldsymbol{X}, \boldsymbol{Y}, Z_{ij} = 1\} \times \mathsf{P}\{Z_{ij} = 1|\boldsymbol{\Phi}^{(t-1)}, \boldsymbol{X}, \boldsymbol{Y}\} \\
&= \frac{\delta_{jd}^{(t-1)} \mathbf{1}\{y_{id} = 0\}}{\delta_{jd}^{(t-1)} \mathbf{1}\{y_{id} = 0\} + (1 - \delta_{jd}^{(t-1)}) F_{jd}(0; \boldsymbol{\psi}_{jd}^{(t-1)})} \times z_{ij}^{(t)},
\end{aligned} \tag{7}
$$

et

$$
z_{ijd1}^{(t)} = z_{ij}^{(t)} - z_{ijd0}^{(t)} \tag{8}
$$

où F_{jd} est la fonction de répartition de la partie positive f_{jd}.

Étape MC

À l'étape MC, nous cherchons à maximiser $Q(\boldsymbol{\Phi}; \boldsymbol{\Phi}^{(t-1)}, \boldsymbol{X}, \boldsymbol{Y})$, qui peut être décomposée en trois parties comme suit

$$
Q(\boldsymbol{\Phi}; \boldsymbol{\Phi}^{(t-1)}, \boldsymbol{X}, \boldsymbol{Y}) = Q_{\boldsymbol{\alpha}}^{(t)} + Q_{\boldsymbol{\delta}}^{(t)} + Q_{\boldsymbol{\Psi}}^{(t)} \tag{9}
$$

où

$$
Q_{\boldsymbol{\alpha}}^{(t)} = \sum_{i=1}^{n} \sum_{j=1}^{g} z_{ij}^{(t)} \log \pi_j(\boldsymbol{x}_i; \boldsymbol{\alpha}_j), \tag{10}
$$

$$
Q_{\boldsymbol{\delta}}^{(t)} = \sum_{i=1}^{n} \sum_{j=1}^{g} \sum_{d=1}^{D} \left\{ z_{ijd0}^{(t)} \log \delta_{jd} + z_{ijd1}^{(t)} \log(1 - \delta_{jd}) \right\}, \tag{11}
$$

et

$$
Q_{\boldsymbol{\Psi}}^{(t)} = \sum_{i=1}^{n} \sum_{j=1}^{g} \sum_{d=1}^{D} z_{ijd1}^{(t)} \log f_{jd}(y_{id}; \boldsymbol{\psi}_{jd}). \tag{12}
$$

Pour maximiser $Q_{\boldsymbol{\alpha}}^{(t)}$, nous utilisons la même maximisation conditionnelle que celle décrite dans Fung et coll. (2019a). Nous la maximisons d'abord par rapport à $\boldsymbol{\alpha}_1$ avec $\boldsymbol{\alpha}_j$ fixé

à $\boldsymbol{\alpha}_j^{(t-1)}$ pour $j = 2, 3, \ldots, g-1$. La prochaine étape consiste à maximiser par rapport à $\boldsymbol{\alpha}_2$ avec $\boldsymbol{\alpha}^{(t)}$ à jour et un autre $\boldsymbol{\alpha}_j$ fixé à $\boldsymbol{\alpha}_j^{(t-1)}$ pour $j = 3, 4, \ldots, g-1$. Le processus se poursuit jusqu'à ce que tous les $\boldsymbol{\alpha}$ aient été mis à jour. Pour obtenir chaque $\boldsymbol{\alpha}_j^{(t)}$, on utilise la méthode des moindres carrés repondérés itérativement (Jordan et Jacobs 1994) jusqu'à la convergence.

Pour $Q_\delta^{(t)}$, chaque δ_{jd} peut-être mis à jour en utilisant la solution suivante :

$$\delta_{jd}^{(t)} = \frac{\sum_{i=1}^{n} z_{ijd0}^{(t)}}{\sum_{i=1}^{n} (z_{ijd0}^{(t)} + z_{ijd1}^{(t)})}. \qquad (13)$$

La maximisation de $Q^{(t)}$ peut aussi être divisée en plus petits problèmes par composante *j* et par dimension *d*. Pour chaque $j = 1, \ldots, g$ et $d = 1, \ldots, D$, le problème se réduit à maximiser la log-vraisemblance pondérée d'une fonction experte où les poids de chaque observation sont donnés par $z^{(t)}$. Par conséquent, pour la mise à jour de chaque $\boldsymbol{\psi}^{(t)}$, les solutions ne sont disponibles que pour des distributions très spéciales (p. ex., Poisson, log-normale). L'optimisation numérique est utilisée dans la plupart des cas, surtout lorsque les observations $\boldsymbol{y}_i$ ne sont pas observées exactement (voir la section 2.3).

Comme il en a été question dans McLachlan et Peel (2004), un mélange de distributions de la sévérité peut avoir une vraisemblance non bornée, ce qui donne lieu à de faux modèles avec des valeurs estimées pour les paramètre extrêmement grandes ou petites. Dans le progiciel **LRMoE**, nous adoptons la même approche maximale a posteriori que dans Fung et coll. (2020), qui utilise des distributions a priori appropriées pour pénaliser les valeurs estimées des paramètres ajustés (voir la section 3). Si nous incluons des fonctions de pénalité, c'est pour éviter d'obtenir de faux modèles du fait que la fonction de log-vraisemblance est non bornée. La pénalité elle-même devrait être assez faible pour qu'elle ait un impact négligeable sur le modèle ajusté. Par contre, les fonctions de pénalité évitent les paramètres qui divergent vers des valeurs déraisonnables et augmentent la robustesse du modèle ainsi obtenu.

Nous recommandons aux lecteurs qui souhaitent obtenir des informations détaillées sur les justifications et les exécutions de consulter la section 4 de Fung et coll. (2020).

2.3 LRMoE avec censure et troncature

La censure et la troncature sont courantes dans les ensembles de données d'assurance et elles doivent être traitées. Par exemple, lorsqu'une limite d'assurance s'applique, les montants des sinistres supérieurs à la limite prennent uniquement pour valeur cette limite, créant ainsi une censure à droite des données complètes sur les sinistres; lorsqu'une franchise de police s'applique, les montants des sinistres inférieurs à la franchise ne sont pas déclarés à l'assureur, ce qui entraîne une troncature à gauche.

Fung et coll. (2021) ont traité du modèle LRMoE avec censure et troncature, où toutes les distributions de composantes sont des Gamma. Pour l'estimation des paramètres avec censure et troncature des données, l'algorithme EMC de la section 2.2 est légèrement

modifié, avec une étape E supplémentaire pour éliminer les incertitudes découlant de la censure et de la troncature. Puisque l'objectif premier du présent document est de démontrer l'application du progiciel **LRMoE**, nous omettrons les détails et invitons les lecteurs intéressés à consulter l'ouvrage cité.

Pour toutes les distributions incluses dans le progiciel **LRMoE**, celui-ci peut effectuer l'estimation des paramètres en cas de troncature et de censure des données. Par conséquent, les données que l'utilisateur doit saisir pour les modèles de mélange diffèrent légèrement de celles qu'il doit saisir dans les progiciels existants. À la section 3, nous donnons un exemple détaillé de l'ajustement de modèle dans notre progiciel.

2.4 Tarification et calcul de réserves dans le LRMoE

La structure du modèle LRMoE permet de calculer facilement des quantités pertinentes pour la tarification et le calcul des réserves actuarielles (voir Fung et coll. 2019b). Au niveau des titulaires de police, les moments et les mesures courantes de la dépendance de $\boldsymbol{y}_i$ (p. ex., le tau de Kendall et le rho de Spearman) peuvent être calculés sous des formes simples. La VaR et l'ECU peuvent aussi être résolues numériquement sans grande difficulté. On peut appliquer plusieurs principes pour calculer les primes des contrats d'assurance, dont la prime pure, la prime basée sur l'écart-type, la prime basée sur l'espérance limitée et la prime *stop-loss*. Les mesures de risque peuvent également être calculées pour chaque titulaire de police (p. ex., la VaR à 99 %). Au niveau du portefeuille, on peut effectuer une simulation pour obtenir la distribution des sinistres agrégés de l'ensemble des titulaires, ce qui est utile pour calculer les primes et la réserve totale pour sinistres. Le processus de simulation est facilité par le simulateur de données inclus dans notre progiciel (voir la section 4.4).

3. Exemple : Ensemble de données simulées

Nous montrerons ici comment ajuster un modèle LRMoE à l'aide d'un ensemble de données simulées joint au progiciel. Le progiciel et l'ensemble de données peuvent être installés et chargés comme suit :

```
# Install and load package
> using Pkg, JLD2
> Pkg.add(url="https://github.com/sparktseung/LRMoE.jl")
> using LRMoE
# Load demo data
> @load "X_obs.jld2" X_obs
> @load "Y_obs.jld2" Y_obs
```

Pour ce qui est de traiter la troncature et la censure des données, les données que l'utilisateur doit saisir relativement à la réponse ***Y*** diffèrent de celles qu'il doit saisir dans les progiciels existants. Pour chaque dimension *d* de l'observation ***yi***, au lieu d'une seule entrée numérique, un quadruple $0 \leq t_{id}^{l} \leq y_{id}^{l} \leq y_{id}^{u} \leq t_{id}^{u} \leq \infty$ est nécessaire, où t_{id}^{l} et t_{id}^{u} sont les bornes inférieure et supérieure de la troncature, et y_{id}^{l} et y_{id}^{u} sont les bornes

inférieure et supérieure de la censure. La valeur exacte de y_{id} se situe entre les bornes de la censure, c'est-à-dire que $y_{id}^l \leq y_{id} \leq y_{id}^u$. Pour un échantillon de taille *n*, une matrice de taille *n x* (4*D*) est nécessaire, où chaque bloc *n* _ 4 décrit une dimension de ***Y***.

Nous présentons ci-après des exemples de lignes de Y_obs dans l'ensemble de données de démonstration. Ces lignes de données illustrent trois scénarios possibles de troncature ou de censure des données. Pour ce qui est de la première ligne, les deux dimensions de ***yi*** sont observées exactement sans troncature, donc
$t_{i1}^l = t_{i2}^l = 0$, $y_{i1}^l = y_{i1}^u = y_{i1}$, $y_{i2}^l = y_{i2}^u = y_{i2}$ et $t_{i1}^u = t_{i2}^u = \infty$.

Pour ce qui est de la deuxième ligne, la deuxième dimension de ***yi*** est tronquée à 5 (p. ex., du fait de l'imposition d'une franchise de police) mais elle n'est pas censurée, donc $t_{i2}^l = 5$. Quant à la troisième ligne, la deuxième dimension de ***yi*** est censurée à droite à 100 (p. ex., du fait de l'application d'une limite d'assurance) et la valeur exacte de y_{i2} est inconnue, donc $y_{i2}^l = 100$ et $y_{i2}^u = \infty$.

> Y_obs[[1,6003,7847],:]
3 x 8 DataFrame

Row	Tl_1	Yl_1	Yu_1	Tu_1	Tl_2	Yl_2	Yu_2	Tu_2
1	0.0	6.0	6.0	Inf	0.0	89.0332	89.0332	Inf
2	0.0	8.0	8.0	Inf	5.0	37.4133	37.4133	Inf
3	0.0	7.0	7.0	Inf	0.0	100.0	Inf	Inf

Tableau 2 : Description de l'ensemble de données de démonstration

Covariable[1]	**Nom**	**Description**
x_{i0}	intercept	Constante 1
x_{i1}	sex	1 pour homme, 0 pour femme
x_{i2}	agedriver	Âge du conducteur : 20-80
x_{i3}	agecar	Âge du véhicule : 0-10
x_{i4}	region	1 pour région urbaine, 0 pour région rurale
Réponse	**Nom**	**Description**
y_{i1}	Y[,1]	Nombre de sinistres relatifs à la branche d'assurance 1
y_{i2}	Y[,2]	Gravité des sinistres relatifs à la branche d'assurance 2
Index des lignes	**Y[,1]**	**Y[,2]**
1–6000	Aucune censure ni troncature	Aucune censure ni troncature
6001–8000	Aucune censure ni troncature	Troncature à gauche à 5[2]
8001–10000	Aucune censure ni troncature	Censure à droite à 100

[1] Toutes les covariables sont générées de façon aléatoire, indépendante et uniforme.

[2] L'ensemble complet de données (X, Y) contient 10 000 lignes. En raison de la troncature à gauche Y[,2], 163 lignes de données sont retirées et l'ensemble de données observé (X_obs, Y_obs) compte seulement 9 837 lignes.

Tableau 3 : Vrai modèle de l'ensemble de données de démonstration

<table>
<tr><td colspan="3">Coefficients de régression logit :</td></tr>
<tr><td colspan="3">$$\boldsymbol{\alpha} = \begin{bmatrix} -0.50 & 1.00 & -0.05 & 0.10 & 1.25 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$</td></tr>
<tr><td colspan="3">Distributions des composantes :</td></tr>
<tr><td></td><td>j = 1</td><td>j = 2</td></tr>
<tr><td>d = 1</td><td>Poisson
$(\lambda = 6)$</td><td>Comptage gamma avec surreprésentation de zéros
$(\delta = 0.20, m = 30, s = 0.50)$</td></tr>
<tr><td>d = 2</td><td>Log-normale $(\mu = 4.0, \sigma = 0.30)$</td><td>Gaussienne inverse
$(\mu = 20.0, \lambda = 20.0)$</td></tr>
</table>

Tableau 4 : Modèle ajusté 1 de l'ensemble de données de démonstration

Coefficients de régression logit :		
$\hat{\alpha} = \begin{bmatrix} -0.4318 & 1.0688 & -0.0502 & 0.0951 & 1.1967 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$		
Distributions des composantes :		
	j = 1	*j* = 2
d = 1	Poisson $(\lambda = 6.016)$	Comptage gamma avec surreprésentation de zéros $(\delta = 0.206, m = 29.956, s = 0.489)$
d = 2	Log-normale $(\mu = 4.001, \sigma = 0.297)$	Gaussienne inverse $(\mu = 20.304, \lambda = 21.756)$

Tableau 5 : Modèle ajusté 2 de l'ensemble de données de démonstration

Coefficients de régression logit :		
$\tilde{\alpha} = \begin{bmatrix} -0.4023 & 1.0643 & -0.0499 & 0.0955 & 1.1788 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$		
Distributions des composantes :		
	j = 1	*j* = 2
d = 1	Poisson avec surreprésentation de zéros $(\delta = 0.003, \lambda = 6.103)$	Poisson avec surreprésentation de zéros $(\delta = 0.206, \lambda = 30.585)$
d = 2	Burr $(k = 1.309, c = 5.232, \lambda = 58.799)$	Gamma $(k = 1.624, \theta = 12.398)$

Pour ajuster un modèle LRMoE, l'utilisateur n'a qu'à spécifier ce qui suit : l'estimation initiale des coefficients de régression logit (alpha_init) et les distributions de composantes à utiliser pour chaque dimension et chaque composante, ainsi qu'une estimation initiale de leurs paramètres (comp_init).

À titre illustratif, nous supposons d'abord que le choix des distributions des composantes par l'utilisateur coïncide avec celui du vrai modèle et que les premières estimations des paramètres sont proches des vraies valeurs. Le code ci-dessous en donne un exemple. Avec deux composantes et cinq covariables, l'estimation initiale alpha_init est une matrice 2 × 5 dans laquelle les entrées de zéro indiquent une estimation non informative. Pour ce qui est des distributions de composantes, comp_init est une matrice 2 × 2; ici, le nombre de lignes (ou de colonnes) correspond au nombre de composantes (ou de dimensions de la réponse). Chaque entrée de comp_init indique un choix de fonction d'experts. Dans ce cas, y_{i1} est le mélange d'une Poisson avec moyenne λ = 10,0 et d'une distribution de comptage gamma avec coefficient de surreprésentation de zéros δ = 0,50, paramètre de forme m = 40 et

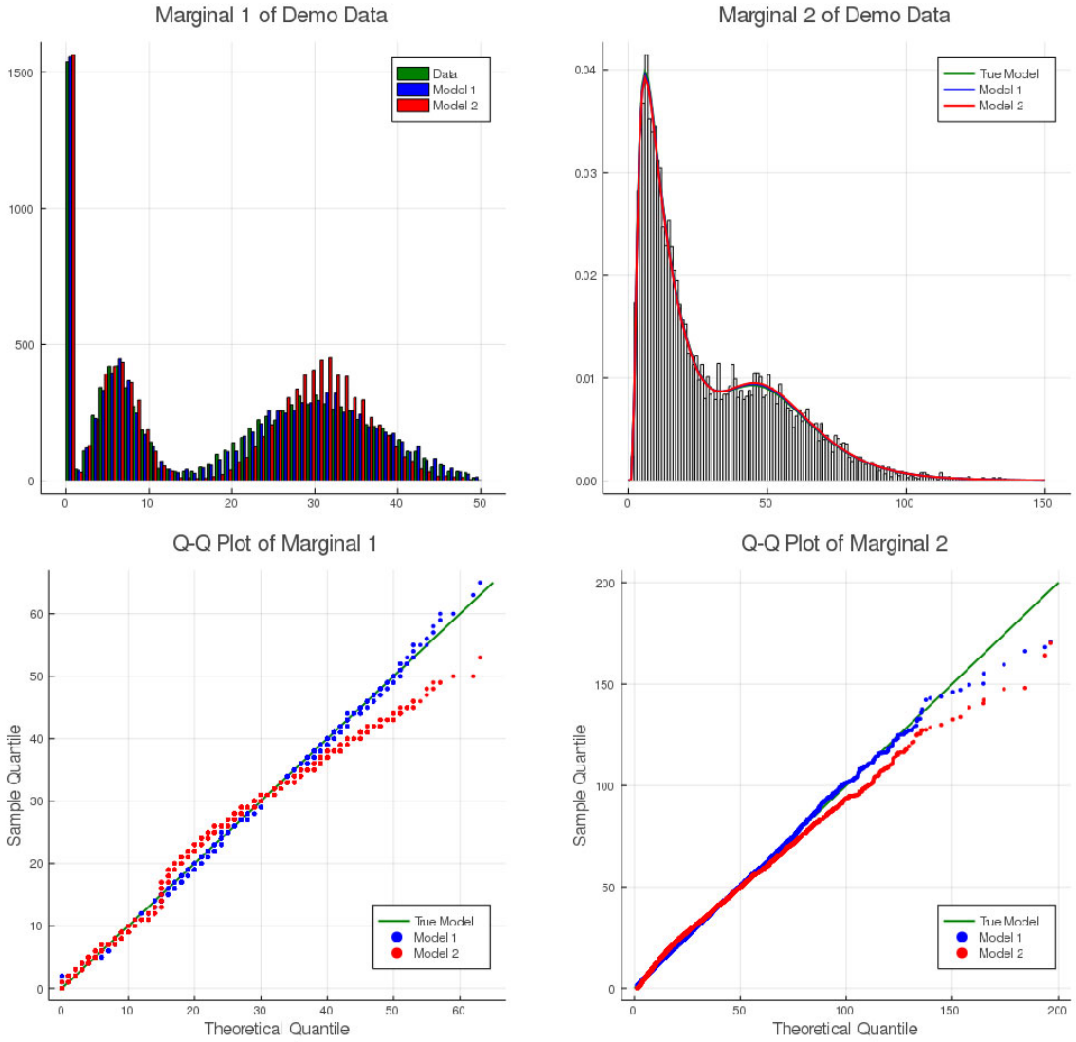
paramètre de dispersion s = 0,80. De même, y_{i2} est le mélange d'une distribution log-normale (μ = 3,0; σ = 1,0) et d'une gaussienne inverse (μ = 15,0; λ = 15,0).

```
# Assume a non-informative guess
alpha_init = fill(0.0, 2, 5)

# Correctly specified component distributions

model_init = [PoissonExpert(10.0)                ZIGammaCountExpert(0.50, 40, 0.80);

                LogNormalExpert(3.0, 1.0) InverseGaussianExpert(15.0, 15.0)]
```

La fonction d'ajustement du LRMoE peut être appelée de la manière indiquée ci-après, ce qui produira un modèle ajusté ainsi que la log-vraisemblance et les critères d'information CIA et CIB. Le résultat peut être visualisé par une fonction summary) ou par les méthodes de Julia usuelles.

```
# Call fitting function
result_1 = fit_LRMoE(Y_obs, X_obs, alpha_init, model_init)

# Result summary
summary(result_1) Model: LRMoE
Fitting converged after 7 iterations
Dimension of response: 2
Number of components: 2
```

Figure 1 : Résultats de l'ajustement de DemoData



*Cette figure est seulement disponible en anglais.

Loglik: -73153.32112999677
Loglik (no penalty): -73147.35233984645
AIC: 146320.7046796929
BIC: 146414.22545854456

 # Inspect fitted model
result_1.model_fit.alpha

2 x 5 Array{Float64,2}:

-0.431755	*1.06875*	*-0.0501667*	*0.0951105*	*1.19667*
0.0	*0.0*	*0.0*	*0.0*	*0.0*

2 result_1.model_fit.comp_dist
x 2 Array:

PoissonExpert(6.01676) *ZIGammaCountExpert(0.206196, 29.9564, 0.488854)*

LogNormalExpert(4.00105, 0.296734) *InverseGaussianExpert(20.3037, 21.7569)*

Les résultats de l'ajustement du modèle sont résumés au tableau 4. Les estimations des paramètres sont assez proches des vraies valeurs. Compte tenu des bruits aléatoires simulés et de la perte d'information due à la censure et à la troncature, la fonction d'ajustement est en mesure d'identifier le modèle véritable lorsqu'il est connu.

En pratique, lorsque le vrai modèle sous-jacent n'est pas connu, l'utilisateur doit effectuer une analyse préliminaire de l'ensemble de données pour déterminer la spécification du modèle et l'initialisation des paramètres. Le tableau 5 renferme les estimations des paramètres d'un autre modèle LRMoE spécifié par l'utilisateur pour l'ensemble de données de démonstration, modèle qui présente des distributions de composantes assez différentes par rapport au vrai modèle.

Les valeurs ajustées de log-vraisemblance des modèles 1 et 2 sont respectivement de -73 147,35 et -74 491,24. La figure 1 présente une comparaison graphique des deux modèles. Bien que les deux modèles aient une performance d'ajustement similaire en ce qui concerne la sévérité des sinistres, le modèle 2 est nettement plus mauvais pour l'ajustement des valeurs petites et extrêmes de la fréquence des sinistres.

4. Exemple : Ensemble de données réelles

Nous montrons ici comment ajuster un modèle LRMoE à un ensemble de données réelles d'assurance. Étant donné que nous avons abordé la procédure d'ajustement de base à la section précédente, nous traiterons ici d'autres fonctionnalités de notre progiciel, y compris l'initialisation des paramètres, l'incertitude du modèle, la simulation, les fonctions de calcul actuariel des primes et la visualisation du modèle.

4.1 Ensemble de données et temps de calcul

Tout au long de cette section, nous utiliserons un ensemble de données sur les sinistres d'assurance automobile en France, freMTPLfreq et freMTPLsev, compris dans le progiciel R **CASdatasets** (Dutang et Charpentier 2019). Les procédures d'analyse exploratoire et de nettoyage des données se trouvent sur le site Web de notre progiciel. L'ensemble nettoyé de données compte 412 609 observations. La distribution des montants des sinistres indique une forte surreprésentation de zéros, car moins de 4 % des titulaires de police ont enregistré un sinistre. En ce qui concerne les montants positifs des sinistres, la distribution est asymétrique à droite, multimodale et a une queue lourde. Les covariables utilisées pour la modélisation sont décrites au tableau 6.

Avant de poursuivre, nous faisons un commentaire sur le temps de calcul de notre progiciel. Comparativement à l'analyse de l'ensemble de données de démonstration à la section 3, l'analyse de freMTPLfreq et freMTPLsev s'apparente à un problème de modélisation actuarielle réaliste comportant de nombreuses covariables et observations. Dans un tel cas, le langage de programmation Julia est nettement plus avantageux que les langages statistiques traditionnels comme R. Certains repères standards se trouvent sur https://julialang.org/benchmarks/.

Notre expérience confirme que Julia est le plus performant des langages : il faut environ 20 heures dans R (avec optimisation dans **Rcpp**) pour ajuster un modèle de cette section, tandis que Julia prend moins de cinq heures pour effectuer la même procédure.

Pour ce qui est des progiciels R correspondants que nous avons créés, nous invitons les lecteurs à consulter Tseung et coll. (2021).

4.2 Initialisation des paramètres

Vu que la procédure d'ajustement du LRMoE passe par une optimisation multivariée, une bonne initialisation des paramètres mènera souvent à une convergence plus rapide, comparativement à une estimation non informative.

Tableau 6 : Description des données sur l'assurance automobile en France

Covariable	**Nom**	**Description**
x_{i0}	Intercept	Constante 1. Classe par défaut des variables catégoriques.
x_{i1}	CarAge	Âge du véhicule en années. Domaine : $0 \sim 100$
x_{i2}	DriverAge	Âge du conducteur en années. Domaine : $18 \sim 99$
$x_{i3} \sim x_{i,13}$	Power	Puissance du véhicule, sous forme de variable catégorique ordonnée : $\mathrm{d} \sim \mathrm{o}$. La valeur par défaut est « d ».
$x_{i,14} \sim x_{i,19}$	Brand	Marque du véhicule : 7 catégories. La valeur par défaut est « Fiat ».
$x_{i,20}$	Gas	Type de carburant du véhicule : diesel ou essence ordinaire. La valeur par défaut est « Diesel ».
$x_{i,21} \sim x_{i,29}$	Region	Région d'émission de la police en France : 10 catégories. La valeur par défaut est « Aquitaine ».
Réponse	**Nom**	**Description**
y_{i1}	ClaimAmount	Montant du sinistre du titulaire de police

Gui et coll. (2018) ont proposé une procédure d'initialisation pour un mélange de distributions d'Erlang d'ajustement, qui a recours au partitionnement en k-moyennes et à la méthode des moments avec clusters. C'est ce qui a été utilisé dans Fung et coll. (2020) et cela produit des valeurs de départ de paramètres qui sont relativement bonnes.

Notre progiciel comprend une fonction d'initialisation qui applique la méthode des moments avec clusters à toutes les distributions de composantes. Une analyse préliminaire est nécessaire pour déterminer le nombre de clusters (composantes) à utiliser.

Étant donné que la partie positive de toutes les distributions de notre progiciel est unimodale, un point de départ heuristique consiste à examiner l'histogramme empirique des données et à compter le nombre de pics (voir la figure 3).

À titre d'exemple, nous décrivons ci-après la procédure d'initialisation d'un LRMoE à trois composantes. L'utilisateur doit saisir la réponse Y et la covariable X, telles qu'elles sont utilisées dans la fonction d'ajustement. De plus, le troisième argument [3]correspond au nombre de composantes, tandis que le dernier argument *["continuous"]* indique que la réponse Y a à 1 dimension et est continue. Pour l'ensemble de données de démonstration utilisé à la section 3, il faudrait saisir *["discrete" "continuous"]*.

```
# Initialize a 3-component model
init_3 = cmm_init(Y, X, 3, ["continuous"])
```

La fonction cmm_init produira une liste d'initialisation des paramètres. Pour les coefficients de régression logit, nous supposons une estimation non informative de l'influence des covariables, ce qui donne lieu à des coefficients nuls pour toutes les covariables. L'utilisateur est libre d'y intégrer les connaissances antérieures par modification ultérieure. La taille relative de chaque cluster est prise en compte par les termes constants. Dans l'initialisation suivante, la taille de trois clusters latents est proportionnelle à ($e^{-1,41667}$, $e^{-2,8687}$, $e^{0,0}$); autrement dit, la proportion de ces clusters dans tout l'ensemble de données est donnée par (0,3842; 0,0330; 0,5827).

```
> init_3.alpha_init
3 x 30 Array{Float64,2}:
 -1.41667   0.0  0.0  …  0.0
 -2.8687    0.0  0.0  …  0.0
  0.0       0.0  0.0  …  0.0
```

En ce qui concerne les paramètres des distributions de composantes, pour chaque dimension de Y, nous appliquerons la méthode des moments avec clusters pour obtenir une initialisation pour tous les types de fonctions d'experts. Le tableau 7 résume toutes les initialisations. La fonction d'initialisation cmm_init produit également des statistiques descriptives (moyenne, coefficient de variation, asymétrie et coefficient d'aplatissement) qui facilitent le choix de la combinaison de fonctions d'experts à utiliser. L'initialisation avec des valeurs de paramètre extrêmement grandes ou petites pourrait donner lieu à un faux modèle ou à un mauvais ajustement et devrait donc être évitée.

La fonction cmm_init produit deux suggestions de modèles en se basant sur la plus haute log-vraisemblance (ll_best) et sur le meilleur test de Kolmogorov-Smirnov (ks_best). Par exemple, l'initialisation du modèle avec la plus haute log-vraisemblance est donnée par :

```
> init_3.ll_best
1 x 3 Array{ZILogNormalExpert{Float64},2}:
 ZILogNormalExpert{Float64}(0.960674, 6.89627, 1.05169)
```

ZILogNormalExpert{Float64}(0.961532, 6.819891, 1.097471)
ZILogNormalExpert{Float64}(0.958489, 6.80334, 1.09848)

4.3 Résultats d'ajustement et sélection de modèles

À titre illustratif, nous ajustons seulement un certain nombre de LRMoE à tout l'ensemble de données françaises d'assurance automobile. Le tableau 8 résume la log-vraisemblance, le CIA et le CIB ajustés. Dans la plupart des cas, l'ajout de composantes augmentera la log-vraisemblance ajustée (p. ex., considérons les modèles iwl, will et wllil). Les modèles sélectionnés par le CIA et le CIB ont respectivement six et cinq composantes. Dans ce contexte particulier, le critère d'information bayésien pénalise fortement les modèles comportant beaucoup de composantes, car la taille de l'échantillon est grande et l'ajout d'une composante augmente grosso modo le nombre de paramètres de 30 (le nombre de coefficients de régression logit).

Outre le CIA et le CIB, la validation croisée constitue un autre critère de sélection de modèles qui permet d'éviter le surajustement d'un trop grand nombre de composantes latentes. Par exemple, Gui et coll. (2018) ont étudié une validation croisée à 10 blocs pour un mélange de distributions d'Erlang d'ajustement, où la log-vraisemblance moyenne sur les ensembles de test est utilisée comme fonction de score pour sélectionner le nombre optimal de composantes. La validation croisée peut être implémentée avec l'aide du calcul parallèle dans Julia (voir aussi la section 4.5).

4.4 Fonctions de calcul de tarifs et de réserves

Notre progiciel comprend un ensemble de fonctions se rapportant au calcul actuariel des primes, au calcul des réserves, à la gestion du risque, y compris le calcul de la moyenne, de la variance, de la VaR, de l'ECU, de l'espérance limitée $E[(Y \wedge u)]$ et de la prime *stop-loss* $E[(Y - d)_+]$ de la variable de réponse. Ces fonctions commencent par la racine predict_, suivie des quantités d'intérêt appropriées (mean, var, VaR, CTE, limit, excess) et les arguments de fonction correspondants.

À titre d'illustration, considérons les trois titulaires de police du tableau 9, où A n'a pas enregistré de sinistre, B en a enregistré un de taille moyenne et C en a enregistré un gros. Voici un exemple de code permettant de calculer différentes quantités d'intérêt :

```
# Mean of claim amount of Policyholders A, B and C.
> predict_mean(X[[1, 33, 96],:], alpha_fit, comp_fit)
[52,859; 52,577; 56,368]
```

Tableau 7 : Exemple d'initialisation des paramètres

Composante		1	2	3
Proportion		0,3842	0,0330	0,5827
Surreprésentation de zéros		0,9615	0,9584	0,9607
Données positives :				
- Moyenne		1686,01	1622,73	1680,57
- Coefficient de variation		2,51	2,50	2,07
- Asymétrie		12,15	13,57	11,61
- Aplatissement		185,67	230,49	183,04
- Initialisation des paramètres				
Gamma	k	0,16	0,16	0,23
	λ	10616,28	10146,28	7231,11
Log-normale	μ	6,82	6,80	6,90
	σ	1,10	1,10	1,05
Gaussienne inverse	μ	1868,01	1622,73	1680,57
	λ	267,76	259,53	390,58
Weibull[1]	k	1,00	1,00	1,00
	θ	1686,01	1622,73	1680,57
Burr[1]	k	351,85	126,52	40,00
	c	0,88	0,90	0,97
	λ	1 218 490	317 504	69 819

[1] Puisque les moments ne peuvent être exprimés en forme close, ces paramètres sont calculés en maximisant la log-vraisemblance en fonction des observations en clusters.

Tableau 8 : Résultats d'ajustement des données françaises pour l'assurance automobile

***g* = 3**	**Log-vraisemblance**	**CIA**	**CIB**	***g* = 4**	**Log-vraisemblance**	**CIA**	**CIB**
lll (69)	-190 010	380 157	380 991	llll (103)	-189 676	379 558	380 684
lib (70)	-190 192	380 524	381 289	lill (102)	-189 884	379 973	381 087
iwl (69)	-190 110	380 358	381 112	llll (102)	-190 008	380 220	381 335
llb (70)	-190 202	380 545	381 310	will (102)	-190 057	380 319	381 445
wll (69)	-190 225	380 588	381 342	bill (103)	-190 109	380 422	381 537
***g* = 5**	**Log-vraisemblance**	**CIA**	**CIB**	***g* = 6**	**Log-vraisemblance**	**CIA**	**CIB**
lllll (135)	-189 323	378 916	**380 392**	llllll (168)	**-189 268**	**378 872**	380 708
lllil (135)	-189 662	379 594	381 070	llblll (169)	-189 441	379 220	381 067
bllil (136)	-189 696	379 663	381 150	liwlll (168)	-189 939	380 214	382 050
bwliw (136)	-189 767	379 805	381 292	llwlll (168)	-190 235	380 805	382 642
wllil (135)	-189 903	380 076	381 551	liwlbl (169)	-190 276	380 890	382 738

Note : Les distributions de composantes sont représentées par la première lettre; par exemple, l pour log-normale, b pour Burr. Toutes les composantes ont une surreprésentation de zéros. Le nombre entre parenthèses représente le nombre total de paramètres. Le critère d'arrêt est une amélioration de la log-vraisemblance < 0.05, ou 500 itérations. Pour chaque *g*, les valeurs de log-vraisemblance sont en ordre décroissant. Les valeurs optimales globales sont en caractères gras.

 # 99% VaR of claim amount of Policyholders A, B and C.

> *predict_VaR(X_obs[[1, 33, 96],:], alpha_fit, comp_fit, 0.99)*

Tableau 9 : Titulaires sélectionnés parmi l'ensemble de données françaises sur l'assurance automobile

	Sinistre	ID	Âge du véhicule	Âge du conducteur	Puissance du véhicule	Marque	Type de carburant	Région
A	0	1	0	46	*g*	JK	Diesel	A
B	302	33	1	64	*g*	JK	Essence ordinaire	IF
C	9 924	96	0	51	*j*	JK	Essence ordinaire	IF

Tableau 10 : Calcul des tarifs pour les titulaires sélectionnés parmi l'ensemble de données françaises sur l'assurance automobile

Principe de calcul des primes des titulaires		A		B		C	
		A priori	A posteriori	A priori	A posteriori	A priori	A posteriori
Au niveau individuel	$E[Y]$	52,86	51,99	52,58	215,42	56,37	200,28
	$E[(Y-1000)_+]$	29,83	29,29	28,51	149,53	30,44	145,76
	$E[Y \wedge 100000]$	39,22	38,46	39,67	162,63	43,15	140,00
Au niveau du portefeuille	VaR(90)	55,89	54,99	55,59	227,83	59,60	211,81
	VaR(95)	57,20	56,28	56,90	233,16	61,00	216,77
	ECU(70)	55,77	54,87	55,48	227,34	59,47	211,36
	ECU(80)	56,59	55,68	56,29	230,69	60,35	214,46
	ECU(90)	58,04	57,10	57,73	236,59	61,89	219,95

Note : Pour ce qui est des principes de calcul des primes au niveau du portefeuille, la répartition est fonction du montant relatif de la prime pure, où le poids pour le titulaire de police *i* est $w_i = E[Y_i] / \sum_j E[Y_j]$. Pour chaque titulaire et chaque principe, le calcul est effectué en fonction de la probabilité a priori (colonne de gauche) et en fonction de la probabilité a posteriori (colonne de droite).

[1198,970; 1195,295; 1204,096]

 # Mean excess of claim amount (d=1000) of Policyholders A, B and C.

> *predict_excess(X_obs[[1, 33, 96],:], alpha_fit, comp_fit, 1000)*

[29,825; 28,512; 30,435]

Le calcul actuariel des primes peut s'effectuer en fonction des distributions individuelles des sinistres ou de la distribution agrégée des sinistres de l'ensemble du portefeuille. Ces deux méthodes peuvent être réalisées au moyen des fonctions intégrées dans notre progiciel.

Au niveau individuel, les fonctions précitées peuvent être appelées directement pour calculer la prime pure $E[Y]$, ainsi que la valeur de la prime lorsqu'il existe une franchise de police ($E[(Y-d)_+]$) ou une limite d'assurance ($E[(Y \wedge u)]$). La première partie du tableau 10 résume le calcul des primes des titulaires A, B et C, selon les principes de calcul au niveau individuel.

Au niveau du portefeuille, la fonction sim_dataset (voir la section 4.6) simulera une valeur de réponse (c.-à-d. le montant du sinistre) pour chaque titulaire de police, qui peut être additionnée aux sinistres agrégés du portefeuille selon un scénario possible. En réalisant plusieurs simulations répétées de la totalité du portefeuille, on obtiendra la distribution empirique des sinistres agrégés. La VaR et l'ECU des sinistres agrégés peuvent être obtenues à partir de l'échantillon simulé et sont utiles pour établir la réserve totale de l'assureur. En outre, la VaR et l'ECU peuvent être réaffectées aux titulaires sous forme d'une prime majorée, selon un régime de pondération qui reflète le degré de risque relatif des titulaires (p. ex., la grandeur relative de leur prime pure). La seconde partie du tableau 10 résume le calcul des primes des titulaires A, B et C selon les principes de calcul au niveau du portefeuille.

4.5 Incertitude des paramètres

En plus d'obtenir des estimations ponctuelles des paramètres du modèle, il est essentiel de calculer leurs intervalles de confiance afin d'identifier les paramètres importants. Étant donné que la log-vraisemblance dans l'équation (4) devient beaucoup plus compliquée lorsque les données sont tronquées ou censurées, le calcul analytique de la variance et des intervalles de confiance des estimations des paramètres ne sera sans doute pas possible et pourrait faire l'objet de problèmes numériques à l'étape de la mise en œuvre.

Nous pouvons utiliser plutôt les méthodes de *bootstrap* de Grün et Leisch (2004). Voici la description d'un simple algorithme de *bootstrap* non paramétrique :

1. Ajuster un LRMoE en utilisant l'ensemble de données original et obtenir une estimation $\hat{\Phi}$.
2. Pour un nombre total fixe d'itérations, disons 200, tirer un échantillon avec remise de l'ensemble de données original, sur lequel on peut ajuster de nouveau un LRMoE avec les mêmes distributions de composantes.
3. Les estimations $\hat{\Phi}$ 1, $\hat{\Phi}$ 2, ..., $\hat{\Phi}$ 200 des échantillons *bootstrap* fourniront une estimation de la variance des paramètres.

L'algorithme qui précède peut être facilement intégré en parallèle dans Julia de la manière qui suit. Nous avons inclus les intervalles de confiance estimés dans l'annexe, calculés à partir de 200 échantillons *bootstrap*.

```
> using Distributed
> @Distributed for i in 1:200
    result = fit_LRMoE(Y, X, alpha_init, model_init)
    @save "result_"*"$(i)"*".JLD2" result # save the result end
```

4.6 Simulation de modèles

Pour le réglage du LRMoE, les distributions de sinistres des titulaires sont des mélanges des mêmes fonctions d'experts mais avec des poids de mélange éventuellement différents. Par conséquent, la distribution des sinistres agrégés, en tant que somme des sinistres individuels, n'admet habituellement pas une forme simple.

Notre progiciel contient un simulateur d'ensembles de données qui facilite l'analyse de la distribution des sinistres agrégés. Étant donné un portefeuille de titulaires X et une spécification de modèle alpha et comp_dist, le simulateur produira un ensemble de réalisation aléatoire de sinistres pour chaque titulaire.

```
> sim_dataset(alpha, X, comp_dist)
```

Avec de multiples appels à sim_dataset, on peut obtenir une approximation des sinistres agrégés. Cela a été appliqué à la section 4.4 pour calculer les primes en fonction des principes au niveau du portefeuille.

4.7 Visualisation des modèles

Après avoir ajusté et choisi un modèle LRMoE adéquat, l'utilisateur peut le visualiser au moyen des fonctions de traçage intégrées du progiciel ou en créant des tracés personnalisés au moyen des fonctions de simulation génériques combinées à des utilitaires de traçage ou d'autres progiciels spécialisés comme **Plots.jl** et **StatsPlots.jl**.

Nous utiliserons ici le modèle llllll à six composantes pour la démonstration.

Probabilités des classes latentes

La régression logit dans le LRMoE affecte chaque titulaire de police à une classe de risque latente en se basant sur les covariables. Étant donné un mode ajusté et un vecteur de covariables, la probabilité des classes latentes peut être calculée et visualisée à l'aide des fonctions intégrées predict_class_prior et plot_class_prob.

Considérons les titulaires au tableau 9. La fonction predict_class_prior produira les probabilités des classes latentes (prob) ainsi que la classe la plus probable (max_prob_idx). L'exemple de code suivant permet de calculer les probabilités des classes latentes, où chaque ligne représente un titulaire de police et chaque colonne représente une classe latente. Si l'on se fie uniquement à l'information sur les covariables, il n'y a pas beaucoup de différences entre ces titulaires de polices et toutes proviennent probablement de la première classe latente.

```
# Predict latent class probabilities, based on covariates and a model
> predict_class_prior(X[[1, 33, 96],:], alpha_fit).prob
```

```
0.400565  0.120057  0.0192732  0.00377685  0.295405  0.160923
0.450407  0.166925  0.0226136  0.00350641  0.204941  0.151607
0.490915  0.139407  0.0325873  0.00430055  0.179139  0.153652
```

```
# Predict latent class probabilities, based on covariates and a model
> predict_class_prior(X[[1, 33, 96],:], alpha_fit).max_prob_idx
```

```
[1, 1, 1]
```

Dans le cas des titulaires dont l'historique des sinistres est connu, il peut être plus instructif de tenir compte des probabilités a posteriori des classes latentes en appelant les fonctions correspondantes des probabilités a posteriori. La probabilité a posteriori de la classe latente j est donnée par $P\{Z_{ij} = 1 | \hat{\Phi}, \boldsymbol{X}, \boldsymbol{Y}\}$ pour les paramètres ajustés $\hat{\Phi}$ qui peuvent être calculés de façon analogue à l'équation (6). Les lecteurs qui souhaitent en savoir plus peuvent se reporter à la section 6.3.2 de Fung et coll. (2019a).

Considérons de nouveau les titulaires au tableau 9. Le code servant à calculer les probabilités a posteriori est semblable à celui qui précède. Étant donné que les titulaires B et C ont une sinistralité non nulle, une grande partie de la probabilité de la classe latente est transférée aux classes 3 et 6, qui correspondent à la partie centrale et à la queue de la distribution des sinistres (voir aussi le plus gros pic de la figure 3).

Pendant ce temps, le titulaire de police A n'a déclaré aucun sinistre, ce qui entraîne peu de changements aux probabilités des classes latentes, car toutes les composantes sont gonflées à zéro de façon très similaire.

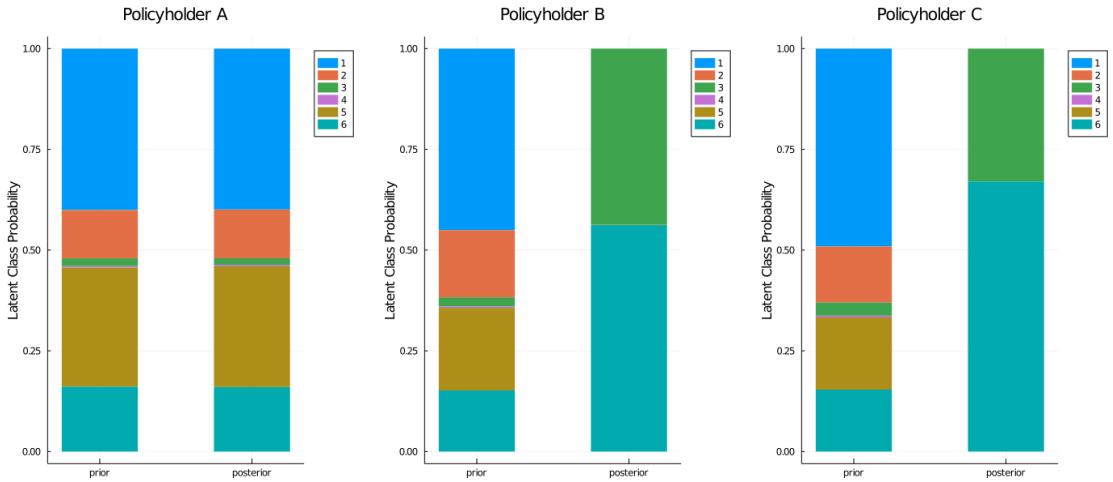
Les probabilités a posteriori sont également utiles pour ajuster les taux de prime. Dans le tableau 10, nous comparons le calcul des primes sur la base des probabilités a priori et des probabilités a posteriori. Il est clair que le titulaire de police A se voit récompenser de son absence de sinistralité par une diminution de sa prime, tandis que B et C connaissent des hausses importantes des taux de prime.

 # Predict latent class probabilities, based on covariates and a model

> *predict_class_posterior(Y[[1, 33, 96],:], X[[1, 33, 96],:],*

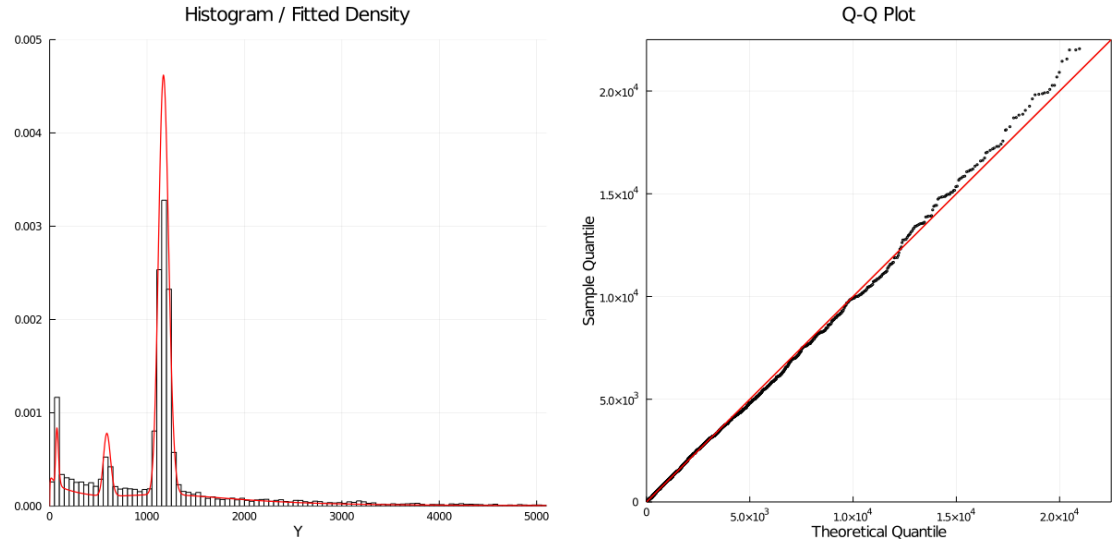
alpha_fit, comp_fit).prob

Figure 2 : Probabilités a priori et a posteriori des classes latentes pour certains titulaires



Note : (Composante 6 (log-normale (6,58; 1,95)) correspond à la queue, tandis que composante 3 (log-normale (6,95; 1,05)) correspond au plus gros pic de l'ensemble de données). Cette figure est seulement disponible en anglais.

Figure 3 : Qualité globale de l'ajustement des sinistres positifs dans l'ensemble de données françaises sur l'assurance automobile



0.399295	0.121097	0.0165664	0.00351454	0.299413	0.160114
9.7166e-207	1.36555e-25	0.437268	4.50508e-13	0.000754288	0.561978
0.0	0.0	0.329656	2.77853e-153	0.000165016	0.670179

Note : (À gauche : tous les points de données; à droite : points de données non extrêmes). Cette figure est seulement disponible en anglais.

```
# Predict latent class probabilities, based on covariates and a model
> predict_class_posterior(Y[[1, 33, 96],:], X[[1, 33, 96],:],alpha_fit, comp_fit).max_prob_idx
[1, 6, 6]
```

Qualité globale de l'ajustement

On peut examiner la qualité globale de l'ajustement en comparant l'histogramme empirique à la courbe de densité ajustée et au diagramme Q-Q des quantiles empiriques et ajustés. Ces tracés peuvent être produits avec la fonction sim_dataset comprise dans notre progiciel, combinée aux fonctions de traçage de base dans Julia. Les graphiques correspondants sont présentés à la figure 3.

Influence des covariables

En apprentissage automatique, on utilise couramment le diagramme de dépendance partielle pour étudier l'influence d'une covariable particulière sur la réponse, en supposant l'indépendance entre les covariables (Friedman 2001). Par exemple, l'effet marginal d'une covariable sur le montant moyen des sinistres peut être obtenu en suivant les étapes suivantes :

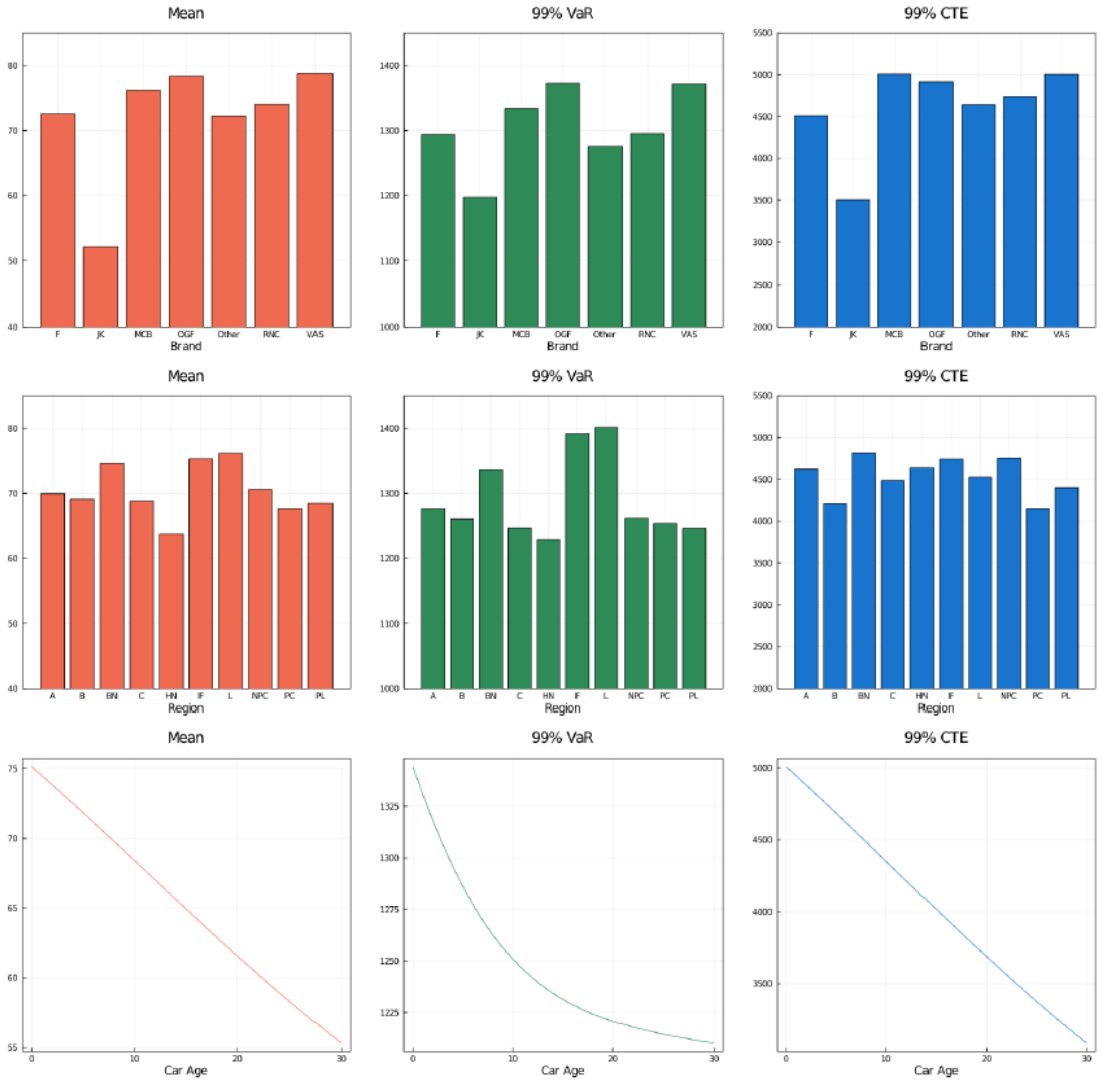
1. Attribuer une valeur particulière à la covariable (disons, marque de véhicule = « F ») pour tous les titulaires, tandis que les autres covariables demeurent inchangées.
2. Utiliser la fonction predict_mean_prior pour calculer les valeurs de réponse moyennes par titulaire de police (voir la section 4.4).
3. Calculer la moyenne globale de toutes les valeurs à l'étape (2), qui fait la moyenne des effets des autres covariables et donne la valeur de réponse moyenne lorsque la marque de véhicule est de type « F ».
4. Répéter les étapes (1) à (3) pour une plage de valeurs de la même covariable d'intérêt; par exemple, substituer « VAS » à « F » comme marque de véhicule.

La procédure ci-dessus peut être généralisée à des covariables continues et à d'autres quantités d'intérêt, comme le quantile de la variable de réponse. Le tracé de l'influence des covariables illustre de façon graphique comment les caractéristiques d'un titulaire de police sont associées à une mesure particulière de la variable réponse. Par exemple, le tracé de la moyenne (ou VaR/ECU) de la variable de réponse en fonction de la marque du véhicule reflète la relation entre le degré de risque global (ou risque de queue de distribution) d'un titulaire de police et la marque du véhicule.

La figure 4 montre l'influence des covariables marque, région et âge du véhicule, qui peut être interprétée comme suit. En ce qui concerne la marque du véhicule, les modèles MCB (Mercedes, Chrysler ou BMW), OGF (Opel, General Motors ou Ford) et VAS (Volkswagen,

Audi, Skoda ou Seat) sont généralement plus risqués. De plus, comparativement aux autres régions, les polices émises dans les régions IF (Île-de-France) et L (Limousin) peuvent être considérées comme étant plus risquées. De plus, on associe généralement les véhicules âgés à des risques moindres.

Figure 4 : Influence des covariables Marque, Région et Âge du véhicule



*Cette figure est seulement disponible en anglais.

5. Sommaire et perspectives

Nous avons présenté ici un nouveau progiciel Julia, **LRMoE**, qui est utile à la modélisation actuarielle des sinistres. Dans cette première version, le progiciel répond au besoin le plus fondamental, à savoir ajuster un modèle LRMoE, en plus d'aider l'utilisateur à visualiser le modèle ajusté et à calculer la prime d'assurance.

Nous avons prévu plusieurs projets, notamment :

- Outils de sélection de modèles : Avec la version actuelle, la sélection des modèles s'effectue au moyen du CIA, du CIB ou de la validation croisée, ce qui oblige l'utilisateur à choisir et à exécuter une sélection de modèles. Certaines procédures automatisées de sélection de modèles pourraient être intégrées au progiciel (p. ex., pénalité de type SCAD dans Fan et Li 2001, et Yin et Lin 2016).
- Outils de sélection de caractéristiques : Les ensembles de données contiennent habituellement un grand nombre de covariables, mais ils ne sont pas tous importants en tant que prédicteurs de la réponse. Même si les tracés de l'influence des covariables peuvent donner une idée de leur importance relative, il est peut-être plus utile de quantifier leur influence et d'offrir une fonction permettant de choisir automatiquement les covariables les plus influentes.

En outre, il est crucial de comparer les performances d'ajustement et de prédiction du LRMoE que nous proposons et celles des modèles de régression classiques. Plusieurs ouvrages ont déjà montré que le modèle de comptage d'Erlang LRMoE donne de bien meilleurs résultats que le modèle linéaire généralisé binomial négatif (voir le tableau 7, section 6.1) de Fung et coll. 2019a), et que le LRMoE gamma transformé donne de bien meilleurs résultats que le LRMoE de diverses sévérités (voir la section 5.3.1 (figure 2 et tableau 2) de Fung et coll. 2020). Néanmoins, il est toujours souhaitable d'effectuer des comparaisons approfondies entre le LRMoE sous diverses fonctions expertes et un large éventail de modèles classiques (y compris, par exemple, le modèle additif généralisé), et c'est ce que nous faisons dans un nouvel ouvrage en cours de préparation.

6. Annexe

L'annexe contient les estimations des paramètres du modèle llllll présenté à la section 4, ainsi que leurs intervalles de confiance à 95 %. Les paramètres significatifs sont indiqués en caractères gras.

$\hat{\alpha}$	Classe 1	Classe 2	Classe 3
Terme constant	**0,733** (0,538; 0,916)	**-1,038** (-1,221; -0,675)	**-0,496** (-0,74; -0,243)
Âge du véhicule	**0,039** (0,031; 0,043)	**-0,015** (-0,039; -0,011)	-0,001 (-0,013; 0,003)
Âge du conducteur	**0,009** (0,006; 0,01)	-0,001 (-0,006; 0,002)	-0,002 (-0,007; 0)
Puissance : e	0,047 (-0,012; 0,192)	-0,099 (-0,23; 0,129)	**0,201** (0,134; 0,368)
Puissance : f	-0,050 (-0,1; 0,071)	-0,007 (-0,123; 0,253)	**0,195** (0,098; 0,375)
Puissance : g	**-0,109** (-0,2; -0,021)	**-0,231** (-0,421; -0,061)	-0,031 (-0,166; 0,098)
Puissance : h	0,040 (-0,083; 0,176)	-0,228 (-0,437; 0,008)	0,091 (-0,077; 0,284)
Puissance : i	0,088 (-0,021; 0,27)	-0,240 (-0,488; 0,039)	**0,267** (0,102; 0,548)
Puissance : j	0,091 (-0,022; 0,259)	**-0,449** (-0,791; -0,17)	**0,297** (0,08; 0,472)
Puissance : k	0,053 (-0,101; 0,315)	-0,001 (-0,217; 0,344)	**0,305** (0,089; 0,654)
Puissance : l	-0,124 (-0,402; 0,119)	**-0,506** (-0,895; -0,077)	-0,016 (-0,467; 0,361)
Puissance : m	-0,071 (-0,503; 0,32)	**-0,546** (-0,863; -0,088)	-0,062 (-0,621; 0,436)
Puissance : n	**-0,577** (-1,083; -0,155)	0,078 (-0,657; 0,747)	-0,032 (-0,655; 0,501)
Puissance : o	-0,228 (-0,577; 0,355)	-0,466 (-0,973; 0,099)	0,192 (-0,603; 0,8)
Marque : JK	**-0,121** (-0,403; -0,07)	**1,023** (0,663; 1,16)	**-1,493** (-1,813; -1,269)
Marque : MCB	**-0,381** (-0,502; -0,173)	**-0,404** (-0,786; -0,103)	-0,231 (-0,443; 0,039)
Marque : OGF	-0,138 (-0,26; 0,024)	**-0,299** (-0,539; -0,04)	0,066 (-0,115; 0,248)
Marque : Divers	-0,181 (-0,391; 0,03)	**-0,422** (-0,778; -0,066)	-0,259 (-0,514; 0,006)
Marque : RNC	**-0,189** (-0,306; -0,031)	-0,043 (-0,237; 0,201)	**-0,178** (-0,335; -0,003)
Marque : VAS	**-0,220** (-0,342; -0,036)	-0,219 (-0,45; 0,127)	-0,020 (-0,167; 0,176)
Type de carburant : Essence ordinaire	**-0,170** (-0,251; -0,148)	**-0,097** (-0,259; -0,018)	**-0,240** (-0,338; -0,174)
Région : BN	0,066 (-0,075; 0,284)	0,067 (-0,216; 0,396)	**0,248** (0,008; 0,531)
Région : B	**0,466** (0,39; 0,683)	**-0,317** (-0,583; -0,055)	**0,341** (0,194; 0,587)
Région : C	**0,111** (0,028; 0,262)	0,048 (-0,08; 0,257)	-0,040 (-0,159; 0,179)
Région : HN	**-0,464** (-0,808; -0,332)	0,106 (-0,372; 0,307)	**-0,745** (-1,233; -0,396)
Région : IF	**0,174** (0,126; 0,357)	**0,517** (0,426; 0,79)	**0,494** (0,361; 0,747)
Région : L	**0,514** (0,315; 0,855)	0,242 (-0,118; 0,78)	**0,763** (0,363; 1,185)
Région : NPC	-0,102 (-0,25; 0,018)	0,120 (-0,123; 0,306)	-0,176 (-0,421; 0,05)
Région : PL	**0,194** (0,078; 0,335)	0,086 (-0,082; 0,362)	0,036 (-0,188; 0,245)
Région : PC	**0,449** (0,343; 0,712)	0,144 (-0,078; 0,583)	**0,332** (0,18; 0,638)
δ	**0,970** (0,969; 0,970)	**0,981** (0,980; 0,984)	**0,836** (0,822; 0,840)
$\hat{\mu}$	**7,065** (7,064; 7,067)	**6,379** (6,370; 6,387)	**6,950** (6,892; 7,031)
$\hat{\sigma}$	**0,044** (0,042; 0,045)	**0,061** (0,053; 0,068)	**1,046** (0,958; 1,109)

$\hat{\alpha}$	Classe 4	Classe 5	Classe 6
Terme constant	**-1,362** (-1,845; -0,902)	-0,077 (-0,095; 0,327)	0
Âge du véhicule	-0,010 (-0,029; 0,004)	**-0,012** (-0,029; -0,011)	0
Âge du conducteur	**-0,015** (-0,022; -0,011)	**-0,002** (-0,005; -0,001)	0
Puissance : e	**0,275** (0,035; 0,561)	-0,026 (-0,091; 0,091)	0
Puissance : f	**0,248** (0,031; 0,529)	**-0,082** (-0,217; -0,03)	0
Puissance : g	0,108 (-0,165; 0,343)	-0,029 (-0,117; 0,034)	0
Puissance : h	0,098 (-0,328; 0,463)	-0,014 (-0,21; 0,079)	0
Puissance : i	-0,009 (-0,413; 0,369)	**-0,313** (-0,619; -0,257)	0
Puissance : j	0,135 (-0,335; 0,584)	**-0,207** (-0,425; -0,109)	0
Puissance : k	0,483 (0; 1,003)	**-0,296** (-0,523; -0,151)	0
Puissance : l	0,518 (-0,146; 1,14)	-0,154 (-0,478; 0,091)	0
Puissance : m	0,134 (-0,786; 0,796)	-0,133 (-0,504; 0,156)	0
Puissance : n	0,560 (-0,528; 1,085)	-0,311 (-0,769; 0,043)	0
Puissance : o	0,559 (-0,453; 1,502)	**-0,528** (-0,966; -0,319)	0
Marque : JK	**-1,794** (-2,387; -1,432)	**0,797** (0,728; 1,025)	0
Marque : MCB	0,016 (-0,391; 0,357)	**0,254** (0,134; 0,453)	0
Marque : OGF	0,093 (-0,234; 0,465)	**-0,128** (-0,347; -0,01)	0
Marque : Divers	0,100 (-0,401; 0,528)	0,087 (-0,085; 0,339)	0
Marque : RNC	-0,287 (-0,562; 0,034)	0,005 (-0,18; 0,091)	0
Marque : VAS	-0,304 (-0,636; 0,108)	-0,051 (-0,236; 0,066)	0
Type de carburant : Essence ordinaire	-0,035 (-0,207; 0,079)	**0,080** (0,019; 0,148)	0
Région : BN	0,321 (-0,248; 0,756)	**-0,395** (-0,641; -0,271)	0
Région : B	**0,434** (0,121; 0,778)	**-0,609** (-0,908; -0,625)	0
Région : C	**0,318** (0,094; 0,595)	**-0,379** (-0,587; -0,355)	0
Région : HN	-0,530 (-1,258; 0,012)	**0,468** (0,408; 0,746)	0
Région : IF	0,260 (-0,048; 0,579)	**-0,347** (-0,445; -0,23)	0
Région : L	-0,287 (-1,042; 0,43)	**-0,714** (-0,99; -0,527)	0
Région : NPC	**-0,570** (-1,064; -0,182)	**-0,185** (-0,376; -0,092)	0
Région : PL	0,388 (-0,009; 0,682)	**-0,462** (-0,66; -0,389)	0
Région : PC	**0,670** (0,293; 1,02)	**-0,525** (-0,755; -0,455)	0
δ	**0,905** (0,896; 0,911)	**0,986** (0,983; 0,992)	**0,968** (0,962; 0,974)
$\hat{\mu}$	**4,350** (4,334; 4,366)	**7,341** (7,172; 7,420)	**6,580** (6,443; 6,688)
$\hat{\sigma}$	**0,183** (0,165; 0,200)	**0,415** (0,076; 0,478)	**1,948** (1,835; 2,023)

Ouvrages de référence

Blostein, M. et T. Miljkovic. « On modeling left-truncated loss data using mixtures of distributions », *Insurance: Mathematics and Economics*, **vol. 85**, 2019, pp. 35–46, ISSN 0167-6687.

Dempster, A.P., Laird, N.M. et D.B. Rubin. « Maximum likelihood from incomplete data via the EM algorithm » *Journal of the Royal Statistical Society: Series B (Methodological)*, **vol. 39**, 1977, n° 1, pp. 1–22.

Dutang, C. et A. Charpentier. *CASdatasets : Insurance datasets*, version 1.0-10 du progiciel R, 2019.

Fan, J. et R. Li. « Variable selection via nonconcave penalized likelihood and its oracle properties », *Journal of the American Statistical Association*, **vol. 96**, 2001, n° 456, pp. 1348–1360.

Friedman, J.H. « Greedy function approximation: A gradient boosting machine », *Annals of Statistics*, **vol. 29**, n° 5, 2001, pp. 1189–1232.

Fung, T.C., Badescu, A.L. et X.S. Lin. « A class of mixture of experts models for general insurance: Application to correlated claim frequencies », *ASTIN Bulletin*, **vol. 49**, n° 3, 2019a, pp. 647-688.

Fung, T.C., Badescu, A.L. et X.S. Lin. « A class of mixture of experts models for general insurance: Theoretical developments », *Insurance: Mathematics and Economics*, **vol. 89**, 2019b, pp. 111–127.

Fung, T.C., Badescu, A.L. et X.S. Lin. « A new class of severity regression models with an application to IBNR prediction », *North American Actuarial Journal*, **vol. 25**, n° 2, 2020, pp. 1–26.

Fung, T.C., Badescu, A.L. et X.S. Lin. « Fitting censored and truncated regression data using the mixture of experts models », 2021. Sur SSRN : https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3740061.

Grün, B. et F. Leisch. *Bootstrapping Finite Mixture Models*, 2004, présenté au COMPSTAT 2004 Symposium, https://statmath.wu.ac.at/AASC/mixtures/Grun+Leisch-2004.pdf.

Grün, B. et F. Leisch. « FlexMix version 2: Finite mixtures with concomitant variables and varying and constant parameters », *Journal of Statistical Software*, **vol. 28**, n° 4, 2008, pp. 1–35.

Gui, W., Huang, R. et X.S. Lin. « Fitting the Erlang mixture model to data via a GEM-CMM algorithm », *Journal of Computational and Applied Mathematics*, **vol. 343**, 2018, pp. 189–205.

Jiang, W. et M.A. Tanner. « On the identifiability of mixtures-of-experts », *Neural Networks*, **vol. 12**, n° 9, 1999, pp. 1253–1258.

Jordan, M.I. et R.A. Jacobs. « Hierarchical mixtures of experts and the EM algorithm », *Neural Computation*, **vol. 6**, n° 2, 1994, pp. 181–214.

Lee, D., Li, W.K. et T.S.T Wong. « Modeling insurance claims via a mixture exponential model combined with peaks-over-threshold approach », *Insurance: Mathematics and Economics*, **vol. 51**, n° 3, 2012, pp. 538–550.

Leisch, F. « FlexMix: A general framework for finite mixture models and latent class regression in R », *Journal of Statistical Software*, **vol. 11**, n° 8, 2004, pp. 1–18.

McLachlan, G. et D. Peel. *Finite Mixture Models*, John Wiley & Sons, 2004.

Meng, X.L. et D.B. Rubin. « Maximum likelihood estimation via the ECM algorithm: A general framework », *Biometrika*, vol. **80**, n° 2, 1993, pp. 267–278, ISSN 0006-3444.

Miljkovic, T. et B. Grün. « Modeling loss data using mixtures of distributions », *Insurance: Mathematics and Economics*, **vol. 70**, 2016, pp. 387–396, ISSN 0167-6687.

Scollnik, D.P. et C. Sun. « Modeling with Weibull-Pareto models », *North American Actuarial Journal*, **vol. 16**, n° 2, 2012, pp. 260–272.

Tseung, S.C., Badescu, A.L., Fung, T.C. et X.S. Lin. « LRMoE: An R package for flexible actuarial loss modelling using mixture of experts regression model », 2021. Sur SSRN : https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3740215

Yin, C. et X.S. Lin. « Efficient estimation of Erlang mixtures using iSCAD penalty with insurance application », *ASTIN Bulletin*, **vol. 46**, n° 3, 2016, pp. 779–799.

Institut canadien des actuaires
360, rue Albert, bureau 1740
Ottawa, ON K1R 7X7
613-236-8196
siege.social@cia-ica.ca

cia-ica.ca
voiraudeladurisque.ca

L'Institut canadien des actuaires (ICA) est l'organisme de qualification et de gouvernance de la profession actuarielle au Canada. Nous élaborons et maintenons des normes rigoureuses, partageons notre expertise en gestion du risque et faisons progresser la science actuarielle pour le bien-être financier de la société. Nos plus de 6 000 membres utilisent leurs connaissances en mathématiques, en statistiques, en analyses de données et en affaires dans le but de prodiguer des services et des conseils de la plus haute qualité pour aider à assurer la sécurité financière de toute la population canadienne.