Optimisation de la couverture de réassurance pour une compagnie non-vie

Julien Le BORGNE
Oriane Yven
Raphaël KOUASSI
Romain MENIER

Dirigé par :
Romain LAILY (Actuaire)
Wassim YOUSSEF (Actuaire)
Anthony NAHELOU (Actuaire)
Françoise PÈNE (Enseignant Chercheur)
Franck VERMET (Enseignant Chercheur)

En partenariat avec le cabinet de conseil SIA PARTNERS et la compagnie d'assurance AERAS

Bureau d'étude 2016-2017
Master 1 Actuariat
Euro-Institut d'Actuariat

Remerciements

Tout d'abord, nous tenons à remercier Wassim Youssef et Romain Laily pour le temps qu'ils nous ont consacré, les explications qu'ils ont pu nous fournir, et enfin pour leur précieux suivi tout au long de l'année lors de l'élaboration de notre bureau d'étude.

De même, nous remercions Franck Vermet, Anthony Nahelou, ainsi que Françoise Pène pour leur aide, leurs suggestions et leurs conseils avisés durant l'accomplissement de ce bureau d'étude.

Enfin, nos remerciements s'adressent au corps enseignant de l'EURIA, et en particulier à Pierre Ailliot, qui a pu nous éclairer pour la simulation des sinistres grâce à ses connaissances poussées en modélisation.

Table des matières

Table des figures

Introduction

L'EURIA propose à ses étudiants de première année de Master d'Actuariat de travailler sur certains sujets d'actualité proposés par des entreprises partenaires. L'objectif est d'acquérir une première expérience des travaux réalisés par des actuaires. Pour mener à bien ce projet, il est nécessaire de mettre à profit les compétences vues et acquises en $1^{ère}$ et $2^{ème}$ année de la formation. Le thème abordé tout au long de ce rapport sera *la réassurance*.

Un des enjeux majeurs des compagnies d'assurance est la bonne évaluation des risques liés à leurs portefeuilles afin d'être en mesure de choisir la "meilleure" couverture de réassurance, c'est-à-dire celle qui couvrira le plus grand nombre de risques au moindre coût. Dans ce cadre, le cabinet de conseil SIA PARTNERS en partenariat avec la compagnie d'assurance AERAS ont proposé à l'EURIA la problématique suivante : "Optimisation de la couverture de réassurance pour une compagnie non-vie".

On définit de manière simpliste la réassurance comme "l'assurance des assureurs". Il s'agit, plus précisément, d'un contrat liant une société spécialisée (réassurance) et la cédante (assurance) dans lequel cette société s'engage à prendre en charge tout ou une partie des risques des assureurs moyennant une rémunération. C'est-à-dire que le réassureur s'engage à rembourser l'assureur, en cas de réalisation du risque, sur une partie des sommes versées par l'assureur au titre des sinistres. Il perçoit, en contrepartie, une portion des primes originales versées par le ou les assurés. Cette activité est très méconnue du grand public, car le réassureur ne traite qu'avec des assurances (en principe).

Le but de notre démarche est de fournir un outil permettant de trouver la meilleure couverture de réassurance pour une compagnie d'assurance non-vie. Nous raisonnerons en termes de réassurance non proportionnelle.

Dans un premier temps, nous définirons la réassurance : son marché, les types de contrats, la réassurance proportionnelle et la réassurance non proportionnelle, etc. Puis, nous expliciterons les différents mécanismes de couverture de réassurance. Ensuite, nous détaillerons notre démarche pour trouver la loi des sinistres en fréquence et en coût, afin de générer des sinistres ayant les mêmes propriétés statistiques que ceux de la base de données qui nous a été fournie. Grâce aux différents mécanismes de couverture, nous déterminerons le restant à charge pour l'assureur, ainsi que le tarif d'un contrat de couverture. Une dernière étape consistera à appliquer un algorithme d'optimisation, afin d'identifier les paramètres de couverture des différents réassureurs, qui seront optimaux pour l'assureur. Les résultats obtenus seront ensuite discutés.

Chapitre 1

Présentation générale de la réassurance

1.1 Définition

Il existe plusieurs définitions de la réassurance. Par exemple selon P. Blanc en 1960, la réassurance est un "*Contrat intervenant pour réaliser la compensation des écarts, soit par insuffisance du nombre de risques, soit par dépassement anormal des sinistres espérés*". Ainsi, la réassurance est un contrat qui permet à la cédante de transférer une partie des risques qu'elle supporte à un ou plusieurs réassureurs en échange d'une compensation financière.

A quoi sert la réassurance ?

Tout d'abord, la réassurance sert à réduire les risques pris par l'assureur. Il est essentiel pour l'assureur de savoir quel montant il doit posséder s'il sait qu'un sinistre peut engendrer une perte n avec une probabilité p de survenance. C'est la problématique du cycle de production inversé. Le montant des sinistres par année oscille autour d'une certaine moyenne statistique. Afin de réduire sa probabilité de ruine, plusieurs choix s'offrent à l'assureur :

- Inclure des chargements importants dans ses tarifs mais il y a un risque de perte de compétitivité
- Accroitre ses fonds propres mais cette décision sera soumise au vote du conseil d'administration.

Le choix le plus judicieux est de se réassurer. En effet, l'assureur gardera sa compétitivité et sera protégé en cas d'écarts importants entre la sinistralité qu'il a prévu et la sinistralité réelle.

Qui pratique la réassurance ?

Il faut être expert pour exercer ce métier, car il faut connaitre les différents mécanismes financiers, d'assurance et de réassurance. Il faut également bien connaitre le marché concerné par le contrat de réassurance, et la politique commerciale de l'assurance en question. Pour cela, il y a deux types de sociétés qui pratiquent des opérations de réassurance :

- Les compagnies d'assurance
- Les réassureurs professionnels, qui ne pratiquent que de la réassurance et opèrent sur la scène internationale.

1.2 Le marché de la réassurance

La pratique de la réassurance est mondiale. En 2015, le volume mondial de la réassurance est estimé à 230 milliards de dollars, avec la répartition suivante : 28% en vie et 72% en non-vie. Le volume mondial des primes d'assurance monte environ à 4 500 milliards de dollars, avec une répartition vie / non-vie opposée à celle de la réassurance.

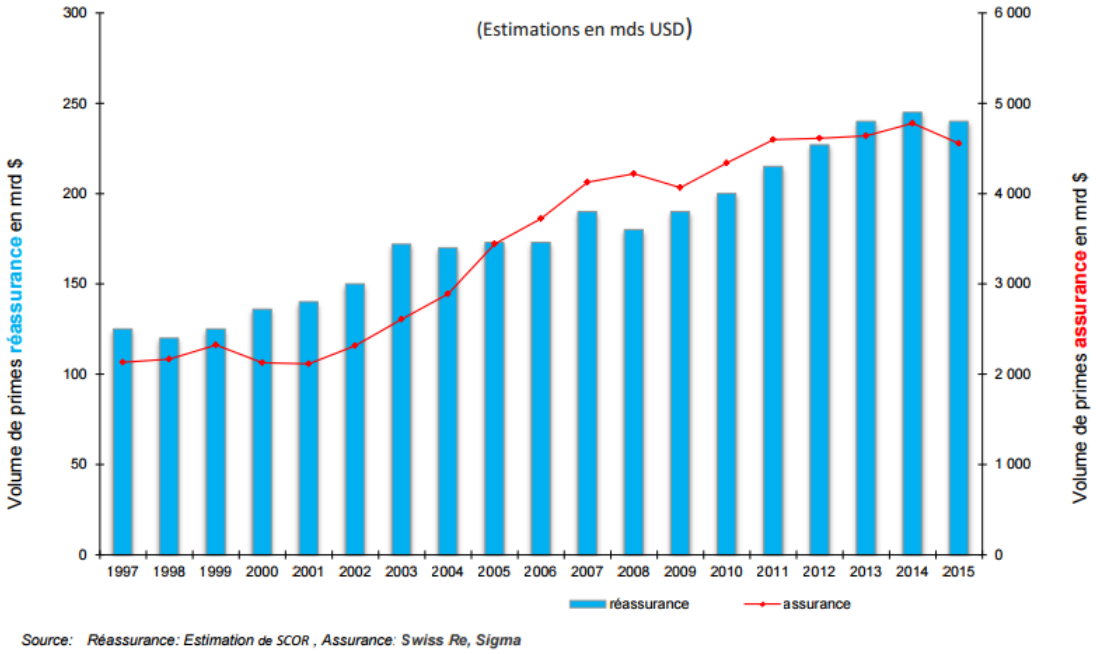

FIGURE 1.1 – Évolution du chiffre d'affaires (Réassurance/Assurance)

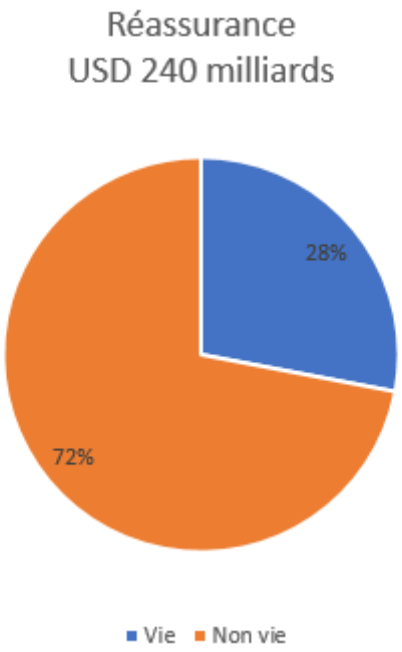

FIGURE 1.2 – Marché mondial de la réassurance

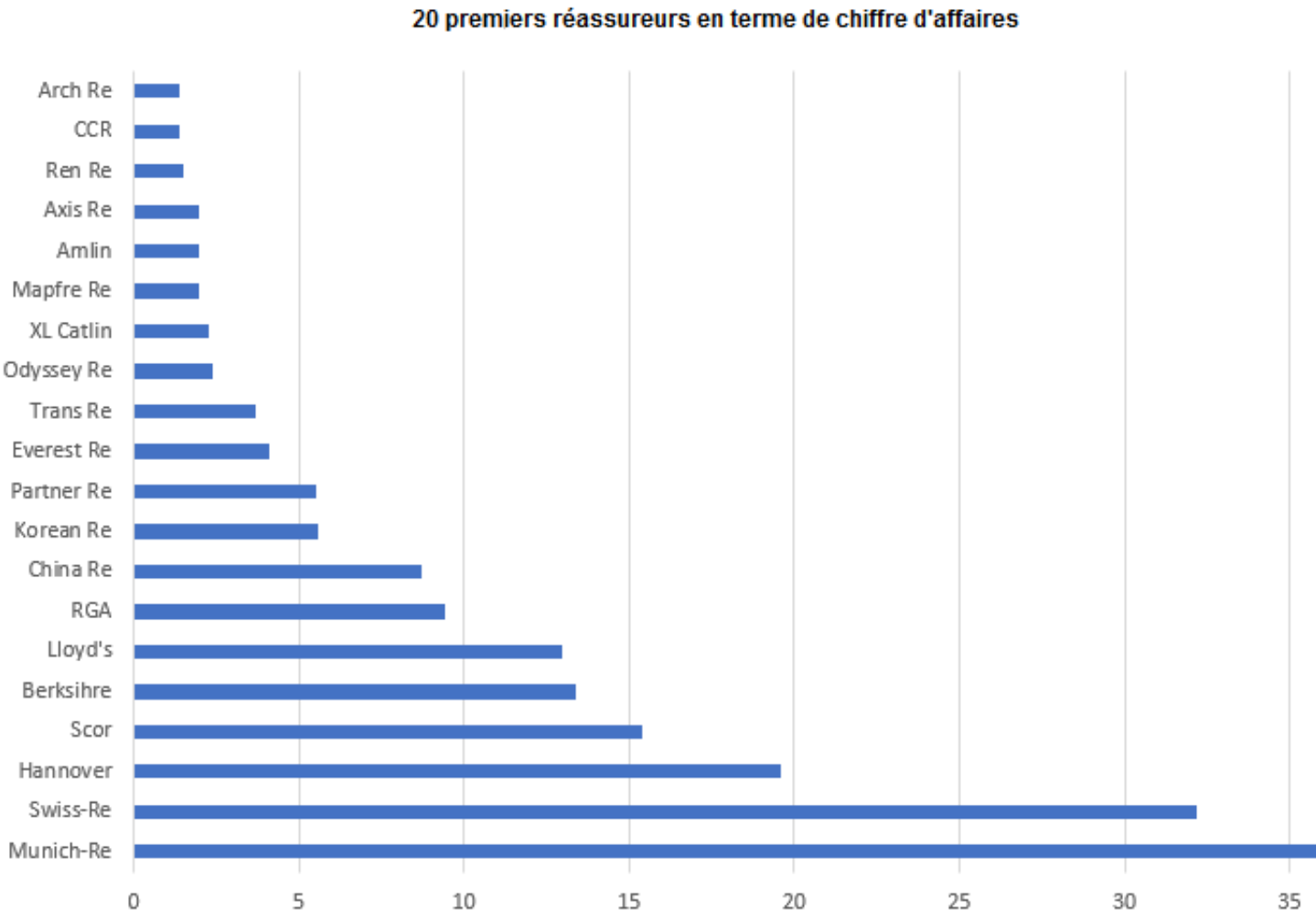

FIGURE 1.3 – Classement des réassureurs mondiaux en 2015

On observe, en 2015, une concentration des réassureurs : les cinq premiers réassureurs représentent à eux seuls 49% du marché de la réassurance et les dix premiers représentent 67% du marché.

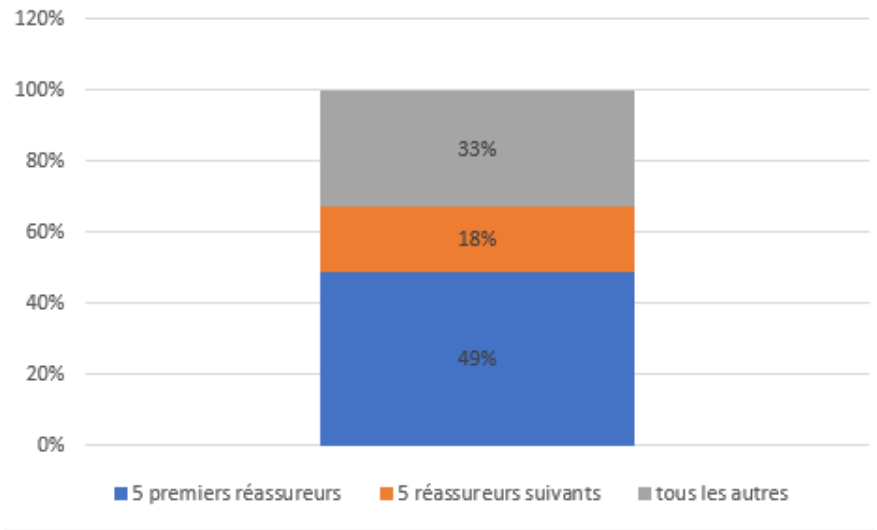

FIGURE 1.4 – Concentration des réassureurs

1.3 Les différents modes de réassurance

En terme juridique, il existe deux modes de réassurance :
- La réassurance obligatoire
- la réassurance facultative.

1.3.1 La réassurance obligatoire

On dit de la réassurance qu'elle est obligatoire lorsqu'assureur et réassureur sont dans l'obligation de céder et d'accepter le risque. Ce risque porte sur l'ensemble d'un portefeuille d'assurance. Le contrat qui lie alors l'assureur et le réassureur est appelé traité de réassurance et chaque année, les deux parties s'accordent pour fixer les conditions du traité. Ce traité peut aussi bien déboucher sur de la réassurance proportionnelle que sur de la réassurance non proportionnelle.

1.3.2 La réassurance facultative

On dit de la réassurance qu'elle est facultative lorsqu'assureur et réassureur peuvent choisir de céder ou d'accepter le risque. Les négociations se font risque par risque. En pratique, l'assureur contacte le réassureur (appelé ici la facultative) pour se couvrir contre un risque. Il fait donc appel à une facultative pour se protéger des risques qui ne rentrent pas dans le cadre du traité d'assurance obligatoire. Par exemple : les risques concernant des sinistres de grande envergure, les risques techniquement complexes... Cette technique de réassurance est relativement rare.

Il existe deux types de réassurance : la réassurance proportionnelle et la réassurance non proportionnelle. Dans la suite, nous expliquons leurs caractéristiques, leurs différences, leurs avantages et leurs inconvénients.

1.4 La réassurance proportionnelle

La réassurance est dite proportionnelle lorsqu'un réassureur prend en charge une proportion du risque de l'assureur. En contrepartie, il reçoit une proportion identique de la prime versée par les assurés. En cas de réalisation du sinistre, il prend en charge le sinistre dans les mêmes proportions que le pourcentage convenu dans le contrat.

1.4.1 Les différents mécanismes de la réassurance proportionnelle

La réassurance proportionnelle contient deux mécanismes de couverture : la *quote-part* et l'*excédent de plein*.

Quote-part

Dans la quote-part, le réassureur prend en charge une proportion constante sur tous les risques du portefeuille. Dans l'exemple suivant, on a pris 70%.

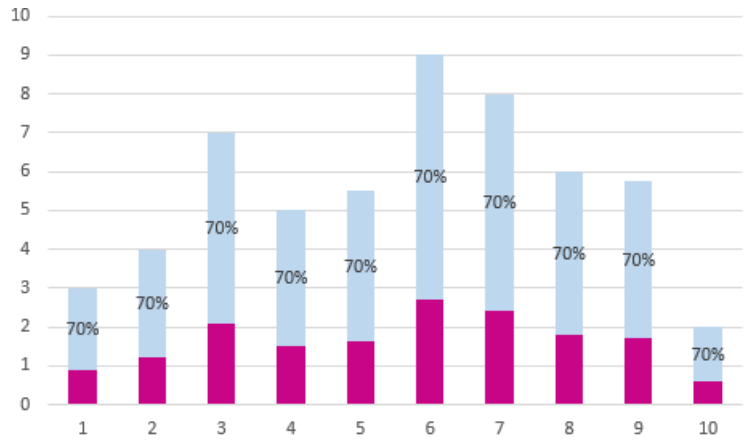

FIGURE 1.5 – Exemple d'une couverture de réassurance quote-part

Posons C_a le capital assuré, C_r le capital réassuré, C_c le capital conservé par la cédante, P_a la prime d'assurance , P_r la prime de réassurance et P_c la prime conservée par la cédante. On a donc :

$$\begin{cases} C_a = C_r + C_c \\ P_a = P_r + P_c \end{cases}$$

Pour mieux comprendre, considérons un portefeuille d'assurance composé des risques suivants :

- risque A : assuré pour un capital de 10 M€ et $P_a = 0.25$ M€
- risque B : assuré pour un capital de 4.8 M€ et $P_a = 0.12$ M€
- risque C : assuré pour un capital de 20 € et $P_a = 0.5$ M€
- risque D : assuré pour un capital de 8 M€ et $P_a = 0.2$ M€
- risque E : assuré pour un capital de 10 M€ et $P_a = 0.25$ M€
- Le total des capitaux assurés (C_a) est de 52.8 M€ et celui des primes (P_a) est de 1.32 M€.

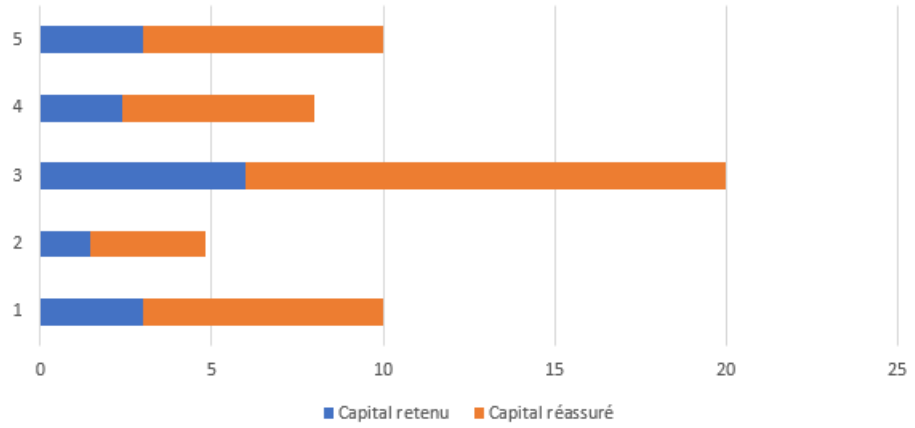

FIGURE 1.6 – Historique du portefeuille observé avec une cession quote-part à 70%

On effectue une cession quote-part à 70%. On pose Q=70%. S désigne le montant de sinistres. On a finalement que :

$$\begin{cases} C_r = Q \cdot (S) \\ C_c = S \cdot (1 - Q) \end{cases}$$

Partage des engagements et des primes :

1. Capitaux assurés : 52.8 M€
2. Capitaux conservés : 15.84 M€
3. Capitaux réassurés : 36.96 M€
4. Prime d'assurance : 1.32 M€
5. Prime conservée : 396 000 €
6. Prime de réassurance : 924 000 €

Excédent de plein

Dans l'excédent de plein, le réassureur prend en charge uniquement la portion des risques dépassant un niveau de capital constant appelé *plein de rétention*. Nous avons pris 3 millions d'euros comme plein de rétention dans l'exemple suivant.

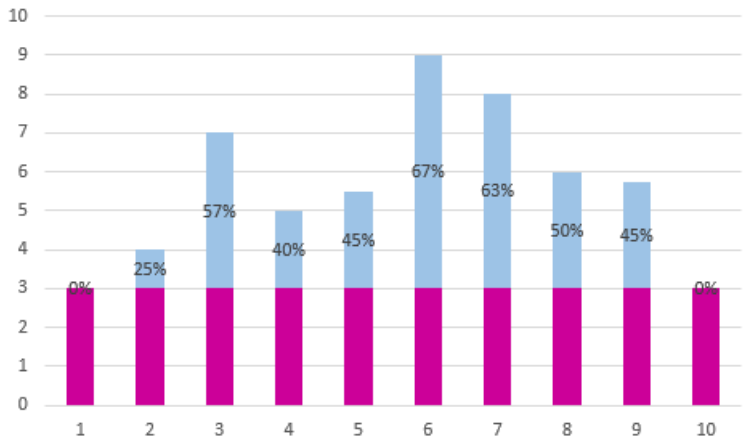

FIGURE 1.7 – Exemple d'une couverture de réassurance par excédent de plein

Reprenons notre portefeuille du 1.1.1. Nous avons appliqué un excédent de plein (plein de rétention de 6 M€)

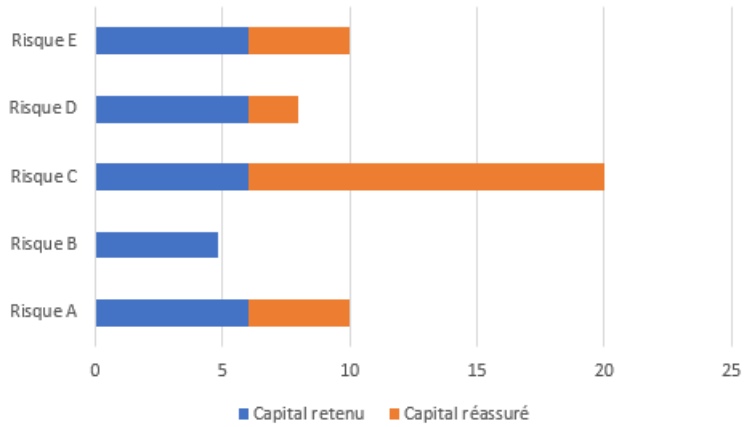

FIGURE 1.8 – Historique du portefeuille observé avec un plein de 6 millions €

Le calcul de la prime de réassurance se fait de la manière suivante, avec N le plein de rétention :

$$\begin{cases} C_c = min(S, N) \\ C_r = max(0, S - N) \\ P_r = C_r \cdot \frac{P_a}{C_a} \end{cases}$$

Les différentes primes de réassurance sont donc :

1. Risque A : $4 \cdot \frac{0.25}{10} = 0.1$ M€
2. Risque B : $0 \cdot \frac{0.12}{4.8} = 0$€
3. Risque C : $14 \cdot \frac{0.5}{20} = 0.35$ M€
4. Risque D : $2 \cdot \frac{0.2}{8} = 0.05$M€
5. Risque E : $4 \cdot \frac{0.25}{10} = 0.1$M€
6. Prime de réassurance : 600 000 €

Partage des engagements et des primes :

1. Capitaux assurés : 52.8 M€
2. Capitaux conservés : 28.8 M€
3. Capitaux réassurés : 24 M€
4. Prime d'assurance : 1.32 M€
5. Prime conservée : 720 000 €
6. Prime de réassurance : 600 000 €

1.4.2 Avantages et inconvénients pour l'assureur

Quote-part

Comme nous l'avons vu au 1.1.1, le calcul de répartition de capital, des primes et des sinistres est très simple : c'est une simple relation de proportionnalité. Nous observons aussi une diminution sensible des engagements de l'assureur (on passe de 52.8M€ à 15.84M€ = 52.8 − 36.96). De plus, ce mécanisme de couverture de réassurance n'est pas spécifique à une branche de l'assurance, on peut l'appliquer sur toutes les branches.

Toutefois, cette méthode ne réduit pas la volatilité du portefeuille, car peu importe le montant du sinistre, l'assureur prendra une partie en charge.

Excédent de plein

Cette méthode est un peu plus bénéfique pour l'assureur, car elle réduit les risques pris par ce dernier. En effet, l'assureur sait d'avance quel montant maximal il doit payer en cas de survenance d'un sinistre. Cependant, comme pour la quote-part, l'assureur doit toujours faire face au risque de cumulations de sinistres (nombre important de sinistres par année).

1.5 La réassurance non proportionnelle

La réassurance est dite non proportionnelle lorsqu'un réassureur reçoit une prime unique annuelle et prend en charge un montant qui ne dépend pas seulement de la survenance ou non d'un sinistre mais aussi de deux paramètres : la **priorité** et la **portée**. La priorité est le seuil à partir duquel le réassureur intervient dans la prise en charge du sinistre, et la portée est une limite à l'engagement du réassureur. La structure de réassurance dépend toujours des besoins de la cédante. Ainsi, on peut qualifier ce type de réassurance "d'excédent de plein plus élaboré", il permet de protéger l'assureur contre les sinistres de type catastrophe et les sinistres à fréquence élevée par exemple.

La réassurance non proportionnelle propose deux types de contrats :
- L'excédent de sinistres (excess of loss) noté XS
- L'excédent de perte annuelle (stop loss)

1.5.1 Fonctionnement du contrat en excédent de sinistres

Le réassureur s'engage à payer, pour tous les sinistres, le montant dépassant la priorité et cela jusqu'à la **portée** qui correspond à l'engagement maximal du réassureur pour un sinistre. La somme de la priorité et de la portée correspond à une valeur appelée **plafond**. La notation pratique est la suivante : ***portée* XS *priorité***. Par exemple, 10M€ XS 10M€ correspond à un traité en excédent de sinistre de portée 10M€ et de priorité 10M€.

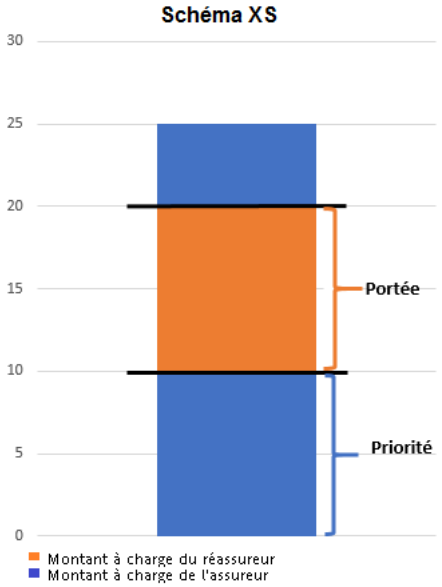

FIGURE 1.9 – Schéma 10M€ XS 10M€

Ainsi, si X est le montant d'un sinistre, le montant à la charge du réassureur sera :

$$\begin{cases} 0 & si \quad X \leq priorité \\ X - priorité & si \quad priorité \leq X \leq plafond \\ Portée & si \quad X \geq plafond \end{cases}$$

Ce montant s'écrit : **min [max(X - priorité, 0), portée]**.

De plus, il existe deux types de traités en excédent de sinistres qui sont l'XS par risque et l'XS par événement. Afin de mieux comprendre la différence entre l'XS par risque et l'XS par évènement, quelques explications et définitions sont nécessaires.
Le risque correspond à la probabilité que survienne un dommage, tandis qu'un sinistre est la matérialisation d'un risque (incendie, tempête,...).
C'est-à-dire le risque peut être :

- Isolé, et dans ce cas : une cause x un risque = 1 sinistre
- Cumulatif, et dans ce cas : une cause x plusieurs risques = 1 évènement
- Agrégatif, et alors : plusieurs causes x plusieurs risques = 1 agrégat

Ainsi, on en déduit que :

- l'XS par risque fournit une couverture de réassurance à chaque fois qu'un risque est sinistré (ex : incendie d'une maison).
- l'XS par évènement intervient lorsque plusieurs risques sont touchés du fait d'une même cause (ex : incendie d'un lotissement).

1.5.2 L'excédent de sinistre par événement

Dans le cas des XS par événement, le dommage est constitué de plusieurs sinistres, nés d'une même cause : catastrophe naturelle, accident industriel, déraillement de train, attentat, etc. Des clauses peuvent être ajoutées au contrat de réassurance. Ces clauses sont généralement restrictives dans le temps et l'espace, pour limiter la charge du réassureur provenant d'un même événement :

- Clause de limitation géographique : la prise en charge sera constituée de plusieurs sinistres issus du même événement et dans un périmètre géographique limité.
- Clause de limitation dans le temps : le traité comporte une durée maximale pendant laquelle les sinistres imputables à une même cause sont pris en compte. Cette période peut généralement varier de 72 à 168 heures en fonction des événements couverts (inondations, accident industriel chimique, etc.).

L'XS par événement est généralement pris en complément d'un XS par risque lorsque l'assureur veut être sûr de ne payer qu'une fois la priorité lorsqu'un événement touche au moins deux ou trois polices de son portefeuille.

1.5.3 L'excédent de perte annuelle ou stop-loss

On utilise un contrat dit « *stop-loss* » lorsque la sinistralité se mesure à l'année. On calcule donc la charge annuelle globale des sinistres en question. Avec cette charge, on calcule le ratio $\frac{S}{P}$ ($\frac{Sinistres}{Primes}$).

C'est ce ratio qui nous donne la base pour calculer le montant pris en charge par le réassureur.

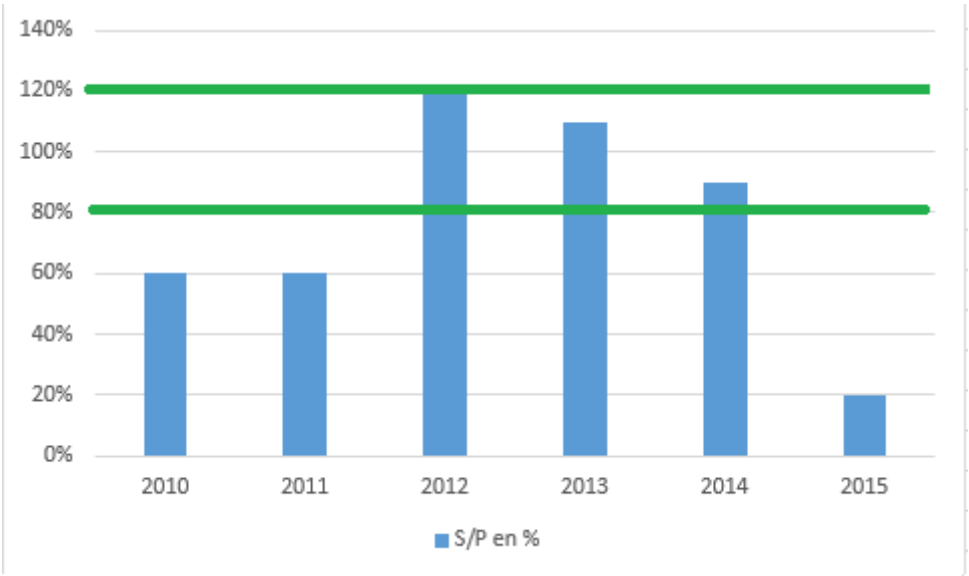

FIGURE 1.10 – Stop-loss

- ***Priorité :*** 80% de la sinistralité (rapportée à l'assiette de primes)

- ***Portée :*** 40% de la sinistralité (rapportée à l'assiette de primes)

1.5.4 Reconstitution de garantie

Dans un contrat en excédent de sinistres, la cédante ne doit pas se trouver à découvert si jamais un second, ou même plusieurs sinistres survenaient dans la même année et absorberaient toute la portée du contrat. Pour éviter cela, les réassureurs ont mis en place la reconstitution de garantie, c'est-à-dire qu'après la survenance d'un sinistre, l'assureur va payer un certain montant pour bénéficier à nouveau de la protection totale du contrat XS qui lui était proposé à l'origine.

En droit coutumier de la réassurance, lorsque rien n'est précisé, le traité est supposé fonctionner avec des reconstitutions illimitées et gratuites. Lorsque le nombre de reconstitutions est précisé, par exemple le réassureur accorde N reconstitutions de garantie, l'assureur pourra bénéficier N+1 fois de la protection que lui procure le contrat XS (la garantie initiale + N reconstitutions). Cette reconstitution peut être gratuite mais se fait souvent au moyen d'une prime additionnelle, définie comme un certain pourcentage de la prime initiale, au prorata des capitaux absorbés (prorata capita) et plus rarement au double prorata de la durée restante du contrat et des capitaux (prorata temporis + prorata capita).

Donc un contrat XS s'accompagne généralement du nombre N de reconstitutions accordées, de la prime de reconstitution (en général égale à 1% de la portée), et la clause de reconstitution (généralement comprise entre 0 et 50% en prorata capita, et entre 0 et 100% en double prorata).

Exemple :

Contrat : Portée XS Priorité
Prime de reconstitution : P € - Clause de reconstitution : c%
Sinistre : Sin € - Survenance : $i^{ème}$ mois de l'année

- ***Prorata capita :***

$$Charge_{Réassureur} = min[max(Sin - priorité, 0), portée]$$

$$Coût_{Reconstitution} = P.\frac{c}{100}.\frac{ChargeRéassureur}{Portee}$$

- ***Double prorata :***

$$Charge_{Réassureur} = min[max(Sin - priorité, 0), portée]$$

$$Coût_{Reconstitution} = P.\frac{c}{100}.\frac{ChargeRéassureur}{Portee}.\frac{12-i}{12}$$

Dans notre bureau d'étude, nous nous plaçons dans le cadre de la réassurance non proportionnelle en excédent de sinistres. De plus, l'assureur bénéficie de plusieurs réassureurs. Précisons que les couvertures des différents réassureurs ne doivent pas se chevaucher, sinon cela signifierait que plusieurs réassureurs pourraient couvrir la même tranche, et donc qu'un assureur se ferait rembourser plusieurs fois cette même tranche.

Chapitre 2

Analyse descriptive du jeu de données

Romain LAILY et Wassim YOUSSEF ont mis à notre disposition un jeu de données. Celui-ci est fictif mais réaliste : il est composé de sinistres concernant une garantie Responsabilité Civile. La base de données contient 5 variables : l'*identifiant* du sinistre, *la date de survenance*, *la date de déclaration*, *la date de réévaluation*, et *le coût* de celui-ci. Nous avons 567 509 observations (lignes). La base de données est composée de 71 406 sinistres différents survenus entre l'année 2000 et l'année 2015. La différence entre le nombre d'observations et le nombre de sinistres provient du fait que la compagnie d'assurance peut réévaluer un sinistre, c'est-à-dire modifier son coût.
Par exemple, une compagnie d'assurance peut estimer que, suite à un sinistre en 2002, elle doit 500€ à son assuré. Mais finalement, après expertise en 2003, elle ne doit plus que 300€. On appelle cela une réévaluation. Les réévaluations peuvent avoir lieu plusieurs fois et plusieurs années après la déclaration du sinistre.

2.1 Quelques statistiques descriptives

Pour avoir une meilleure vision du jeu de données, nous avons établi plusieurs statistiques descriptives du jeu de données. Nous considérons tous les sinistres, même ceux réévalués.

id	surv	decla	date	cout
Min. : 1	Min. :2000	Min. :2000	Min. :2000	Min. : 0
1st Qu.:17337	1st Qu.:2004	1st Qu.:2004	1st Qu.:2005	1st Qu.: 564
Median :34304	Median :2008	Median :2008	Median :2009	Median : 2987
Mean :35005	Mean :2008	Mean :2008	Mean :2009	Mean : 16358
3rd Qu.:53274	3rd Qu.:2012	3rd Qu.:2012	3rd Qu.:2013	3rd Qu.: 3835
Max. :71407	Max. :2015	Max. :2015	Max. :2029	Max. :8790814

FIGURE 2.1 – Résumé du jeu de données

Comme nous pouvons le voir, la variable représentant le coût des sinistres est très **hétérogène** : son écart-type est de 131 082€. La moyenne des coûts de sinistres est de 16 358€ alors que le $3^{ème}$ quartile est de 3 835€. Ainsi 75% des sinistres sont inférieurs à 3 835€ et donc à la moyenne. Cette hétérogénéité aura un impact important sur la manière de simuler des sinistres. En effet, la simulation de nouveaux sinistres constitue la prochaine étape du travail.

Voici un descriptif du nombre de réévaluations par sinistre :

Minimum	Moyenne	Maximum
0	6.94	110

FIGURE 2.2 – Descriptif des réévaluations

Le tableau suivant donne, par année, le coût moyen, le nombre de "vrais" et de "faux" sinistres, le pourcentage de "faux" sinistres. Nous appelons "faux" sinistre un sinistre réévalué à 0€. Cela veut dire que c'est une fausse déclaration de sinistre, ou bien que ce sinistre n'est finalement pas pris en charge.

Annee	Cout moyen	Nb de vrais sinistres	Nb faux sinistres	Pourcentage de faux-sinistres
2000	14923.991	1468	1683	0.5341162
2001	8268.886	1530	1635	0.5165877
2002	8344.054	1826	1560	0.4607206
2003	12724.888	2014	1663	0.4522709
2004	11444.485	2018	1726	0.4610043
2005	10960.474	2387	1831	0.4340920
2006	11447.974	2525	2094	0.4533449
2007	14061.162	2473	2286	0.4803530
2008	9921.263	2489	2063	0.4532074
2009	16275.926	2684	2052	0.4332770
2010	13277.822	3051	2391	0.4393605
2011	13533.663	3096	2486	0.4453601
2012	14481.437	2762	2089	0.4306329
2013	13635.056	3178	2010	0.3874325
2014	10414.372	4527	1636	0.2654551
2015	6725.023	3603	555	0.1334776

FIGURE 2.3 – Tableau descriptif des sinistres

Comme nous pouvons l'observer, il y a chaque année un pourcentage important de "faux" sinistres. Ces sinistres n'intervenant pas dans le processus d'assurance, nous avons décidé de les enlever de la base de sinistres.
Après ce premier traitement, nous avons décidé d'appliquer le processus de réassurance au dernier montant connu de chaque sinistre. Pour cela, nous avons créé une fonction ne gardant que la dernière réévaluation de chaque sinistre.
Ce fut donc la phase de traitement de la base de données.

Après avoir effectué quelques statistiques et traitements, l'étape suivante est de simuler de nouveaux sinistres présentant les mêmes caractéristiques que la base de sinistres retraitée.

2.2 Loi des sinistres

Pour trouver la meilleure couverture de réassurance pour notre compagnie d'assurance, il nous a fallu trouver la distribution que suit notre jeu de données en fréquence et en coût. L'objectif est de simuler plusieurs échantillons suivant les mêmes lois de probabilité que les données fournies. Préalablement, nous avions sélectionné la dernière réévaluation de chaque sinistre.

Dans un premier temps, nous avons observé la fréquence des sinistres par année de survenance, et déterminer empiriquement grâce au graphe suivant la fréquence. Comme nous l'observons sur l'histogramme, il y a eu 1 année où le nombre de sinistres était compris entre 1000 et 1500, 2 années où il était compris entre 1500 et 2000, 5 années où il était compris entre 2000 et 2500, etc... Afin de simuler la loi en fréquence de tous les sinistres, notre approche était

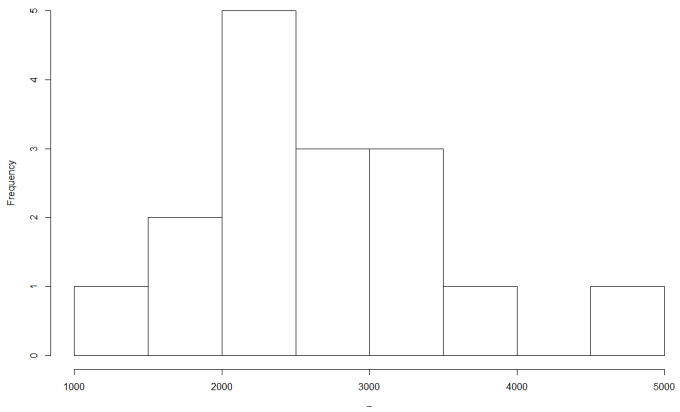

FIGURE 2.4 – Distribution en fréquence des sinistres

empirique. C'est-à-dire que nous tirions, de manière uniforme, un nombre entre 0 et 16 : si ce nombre était inférieur à 1, nous simulions un nombre aléatoire entre 1000 et 1500 sinistres. S'il était strictement compris entre 1 et 3, nous en simulions un nombre aléatoire entre 1500 et 2000 sinistres, ainsi de suite. Cependant, cette méthode n'est pas réaliste, elle ne permet pas d'avoir toutes les valeurs possibles du nombre de sinistres.

Dans un second temps, nous avons donc opté pour une seconde méthode qui consiste à séparer les sinistres attritionnels des sinistres extrêmes et d'étudier leurs lois séparément. Les sinistres attritionnels étant caractérisés par des montants relativement faibles et une forte probabilité d'occurence. Cette méthode semble donc cohérente vis-à-vis de l'hétérogénéité des données.
Nous avons, par la suite, considéré comme attritionnels les sinistres dont le coût ne dépasse pas le quantile à 95% de notre base de sinistres retraitée.

2.2.1 Sinistres attritionnels

Nous avons testé deux méthodes pour trouver la loi des sinistres attritionnels en fréquence et en coût.

Méthode 1 : Simulation d'une loi classique pour la fréquence et utilisation du package MCLUST pour les coûts

Nous décidons de tester trois lois statistiques classiques pour ajuster le modèle sur la fréquence des sinistres attritionnels :

- Une loi de Poisson
- Une loi Binomiale négative
- Une loi Normale.

Chacune des ces lois ont été ajustées avec la fonction *fitdistr* de R. Pour observer les ajustements obtenus, nous superposons les densités des différentes lois avec celle de notre jeu de données (qu'on a obtenu grâce à la fonction *plot(density())* de R).

Loi de Poisson Nous testons H_0 : *La loi des sinistres attritionnels en fréquence est une loi de Poisson* contre H_1 : *La loi des sinistres attritionnels en fréquence n'est pas une loi de Poisson.*
Nous pouvons remarquer qu'une loi de Poisson n'est pas du tout adaptée pour la fréquence des sinistres attritionnels.

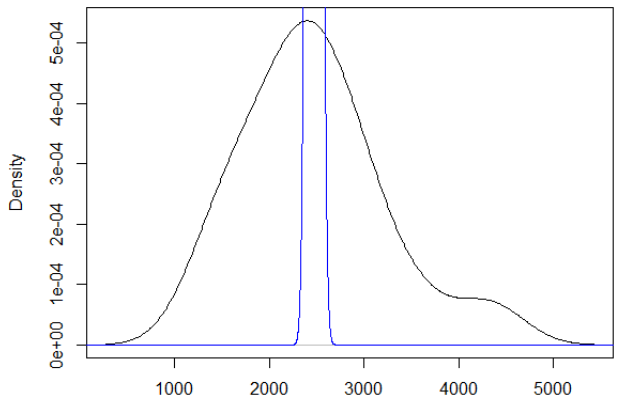

FIGURE 2.5 – La densité d'une loi de Poisson (bleue) superposée à celle du jeu de données(noire)

Comparaison d'une loi Binomiale négative avec une loi Normale Nous testons H_0 : *La loi des sinistres attritionnels en fréquence est une loi Binomiale Négative* contre H_1 : *la loi des sinistres attritionnels en fréquence n'est pas une loi Binomiale négative*. Nous testons également H_0' : *la loi des sinistres attritionnels en fréquence est une Normale* contre H_1' : *La loi des sinistres attritionnels en fréquence n'est pas une loi Normale*.

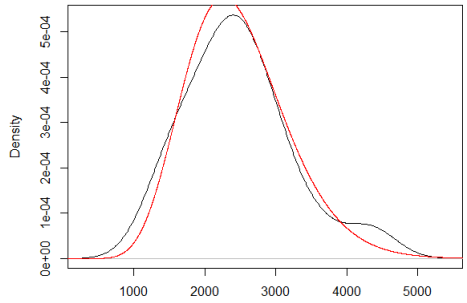

FIGURE 2.6 – La densité d'une loi Binomiale négative (rouge) superposée à celle du jeu de données

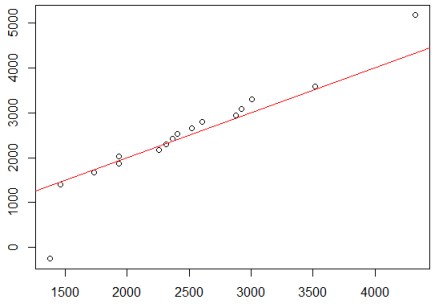

FIGURE 2.7 – QQ-plot de la loi Binomiale négative

La loi Binomiale négative et la loi Normale sont beaucoup plus adaptées qu'une loi de Poisson. Toutefois, la loi Binomiale négative surestime un peu trop le nombre de sinistres par rapport à la loi Normale.

A ce stade de notre étude, nous choisissons donc la loi Normale pour modéliser la fréquence des sinistres attritionnels.

Concernant le montant des sinistres attritionnels, en observant la densité du montant des sinistres (plus précisément le logarithme) du jeu de données (graphique qui suit), nous remarquons qu'elle ressemble à un mélange de lois Normales dépendantes. Nous utilisons un package adapté aux mélanges de lois Normales : le package *MCLUST* de R.

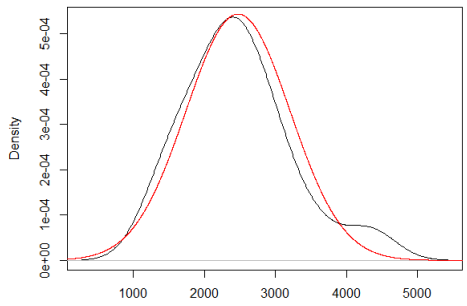

FIGURE 2.8 – La densité d'une loi Normale (rouge) superposée à celle du jeu de données

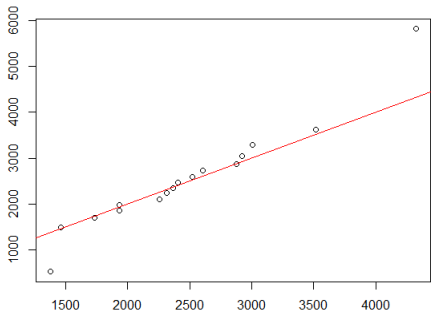

FIGURE 2.9 – QQ-plot de la loi Normale

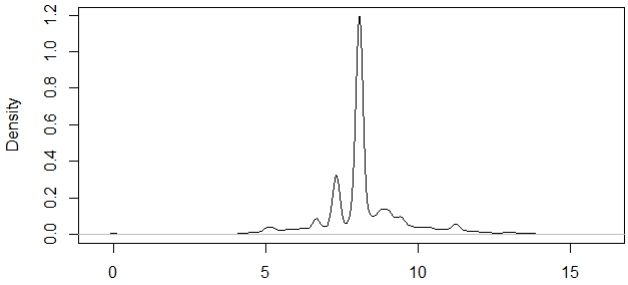

FIGURE 2.10 – La densité du montant des sinistres du jeu de données

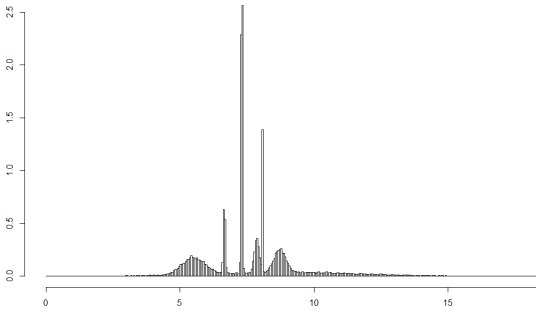

FIGURE 2.11 – La densité simulée avec MCLUST

Problème : La simulation avec MCLUST surestime les montants des sinistres et perturbe la couverture de réassurance. Une simulation individuelle des sinistres attritionnels ne sera pas entièrement efficace. De plus, dans les problématiques actuarielles actuelles, les sinistres attritionnels ne sont pas simulés ligne à ligne.

C'est pourquoi l'approche fréquence∗coût ne sera pas retenue pour modéliser les sinistres attritionnels.

Méthode 2 : Approximation de Wilson-Hilferty

Cette méthode consiste à approcher la charge globale des sinistres attritionnels (notée S_c par la suite) par une transformée d'une variable aléatoire centrée réduite R. On note :

$$\begin{cases} \text{S, la charge globale observée} \\ \mu_S = \mathbb{E}(S),\ l'espérance\ de\ S \\ \mu_Y = \mathbb{E}(Y),\ l'espérance\ de\ Y \\ \sigma_Y^2 = Var(Y),\ la\ variance\ de\ Y \\ \gamma_S, \text{ le coefficient d'asymétrie (skewness, le moment d'ordre 3 de S)} \end{cases}$$

La transformation de Wilson-Hilferty s'écrit :

$$\begin{cases} Y = (S/\mu_S)^h \\ R = {}^{Y-\mu_Y}/_{\sigma_Y} \\ h = 1/3 \end{cases}$$

Il faut ensuite procéder comme suit :

1. Simuler une variable aléatoire $R \sim \mathcal{N}(0,\,1)$,
2. Calculer $\tilde{S} = b_1(R \dot{-} b_2)^3 - b_3$ avec :

$$\begin{cases} b_1 = \gamma_S^2/108 \\ b_2 = \gamma_S/6 - 6/\gamma_S \\ b_3 = 2/\gamma_S \end{cases}$$

3. Calculer $S_{simu} = \mu_S + \sigma_S \dot{\tilde{S}}$

Cette méthode permet d'approcher S_c par S_{simu}.

Validation de l'approximation :
On considère que l'approximation de Wilson-Hilferty est satisfaisante si :
$0 \leqslant \gamma_S \leqslant 1,2$ *(idéalement $\gamma_S < 0,5$)*

Grâce à l'approximation de Wilson-Hilferty, nous avons créé une fonction WH qui renvoie un nombre aléatoire qui représente la charge des sinistres attritionnels cumulés (S_{simu}) pour une année. Nous trouvons $\gamma_S = 0,37$, nous pouvons donc valider notre approximation.

Nous décidons d'effectuer un test de Kolmogorov-Smirnov afin d'avoir un outil supplémentaire nous permettant de valider l'approximation : en effet, une justification basée sur le skewness nous fait nous poser certaines questions sur le choix de b_1, b_2 et b_3.

Test de Kolmogorov-Smirnov Soient $x_1, \ldots, x_n$ n réalisations d'une variable aléatoire X. On se demande s'il est raisonnable de supposer que X suit la loi caractérisée par la fonction de répartition F et on pose les hypothèses de test : H_0 : *X suit la loi F* contre H_1 : *X suit une autre loi*. On mesure l'écart maximum qui existe entre la fonction de répartition empirique de l'échantillon et la fonction de répartition théorique : $d = max \mid F_n(x_i) - F(x_i) \mid$. d est la statistique du test et tend vers 0 sous H_0. Si la p-value du test est inférieure à 5%, on rejette H_0 sinon on l'accepte.

Dans un premier temps nous calculons sur R le montant des sinistres cumulés par années. Puis nous prenons la simulation avec la méthode de Wilson-Hilferty effectuée sur 1 000 000 années. Enfin nous réalisons le test de Kolmogorov-Smirnov. Le test nous donne une p-vaue de 92%. La méthode de Wilson-Hilferty est donc validée pour les sinistres attritionnels.

Toutefois, il est légitime de se demander s'il n'est pas plus simple de prendre une loi Normale plutôt que de passer par l'approximation de Wilson-Hilferty. Nous avons donc simulé 1 000 000 années avec l'approximation de Wilson-Hilferty et 1 000 000 années avec une loi Normale, pour pouvoir les comparer.

Sur le graphique ci-dessous, en bleu nous avons la densité de la simulation de Wilson-Hilferty, en rouge celle d'une loi normale.

Nous pouvons remarquer que la simulation avec Wilson-Hilferty est assez semblable à la loi normale. La différence se fait sur les cas extrêmes : avec la loi normale nous aurons plus d'années avec un coût très faible, donc la simulation avec Wilson-Hilferty peut se montrer plus réaliste qu'une simulation avec la loi normale.

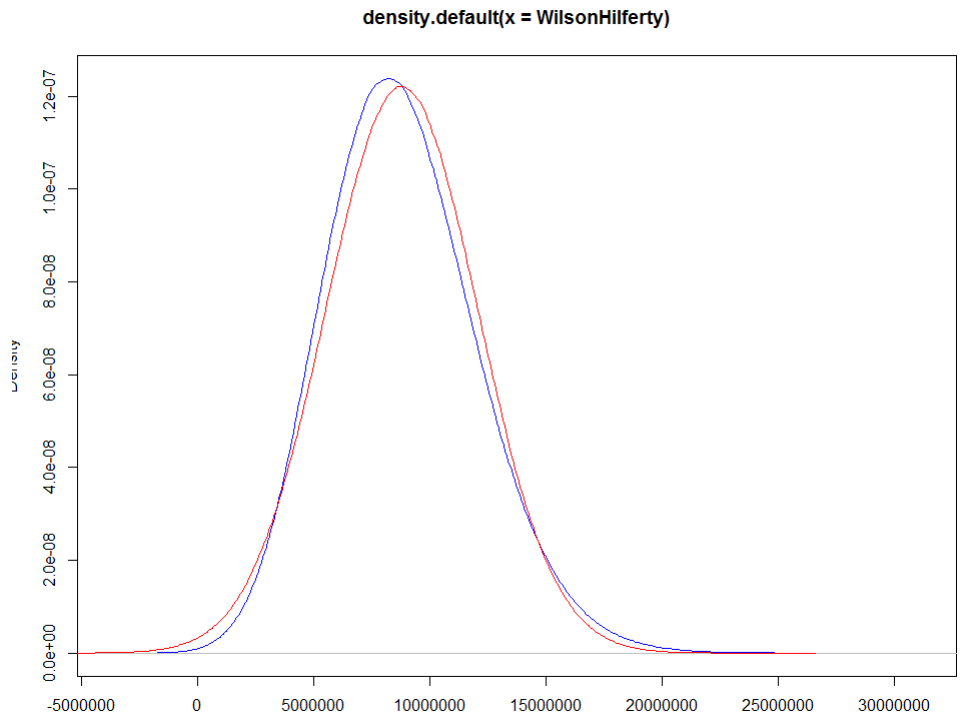

FIGURE 2.12 – Comparaison WH et loi normale

La méthode de Wilson-Hilferty est plus réaliste que la loi Normale, donc nous conservons cette approche pour la simulation des sinistres attritionnels.

2.2.2 Sinistres extrêmes

Puisque le seuil que nous avons choisi pour différencier les sinistres attritionnels des sinistres extrêmes est le quantile à 95% du coût des "vrais" sinistres , nous avons 5% de sinistres extrêmes.

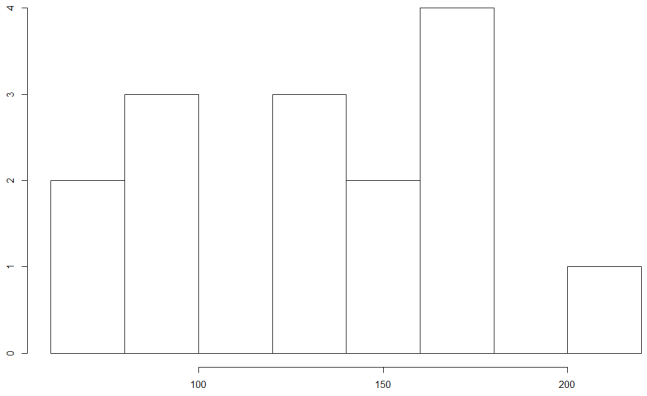

FIGURE 2.13 – Histogramme du nombre de sinistres extrêmes par année

Loi en fréquence

Nous décidons de tester deux lois statistiques classiques, pour ajuster le modèle sur la fréquence des sinistres extrêmes :

- Une loi de Poisson
- Une loi Binomiale négative

Loi de Poisson Nous testons H_0 : *La loi des sinistres extrêmes en fréquence est une loi de Poisson* contre H_1 : *La loi des sinistres extrêmes en fréquence n'est pas une loi de Poisson.*

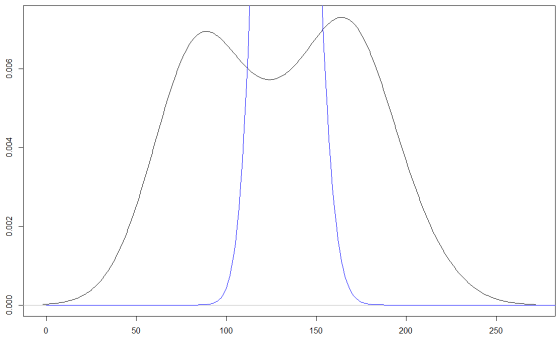

FIGURE 2.14 – La densité d'une loi de Poisson (bleue) superposée à celle du jeu de données (noire

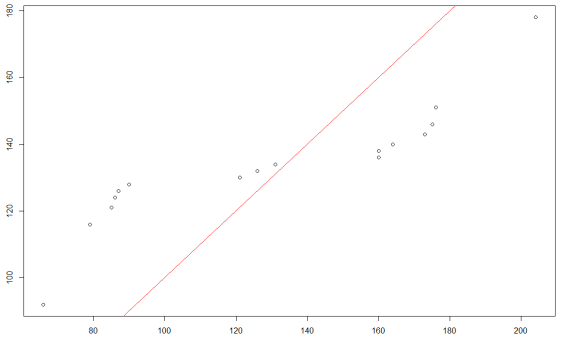

FIGURE 2.15 – QQ-plot de la loi de Poisson

Nous pouvons voir que la loi de Poisson n'est pas adaptée, sa densité ne coïncide pas du tout avec celle des sinistres extrêmes. Tout comme la densité, le QQ-plot n'est pas correct : les points ne sont pas situés sur la droite $y = x$.

Loi Binomiale négative Nous testons H_0 : *La loi des sinistres extrêmes en fréquence est une loi Binomiale négative* contre H_1 : *La loi des sinistres extrêmes en fréquence n'est pas une loi Binomiale négative.*

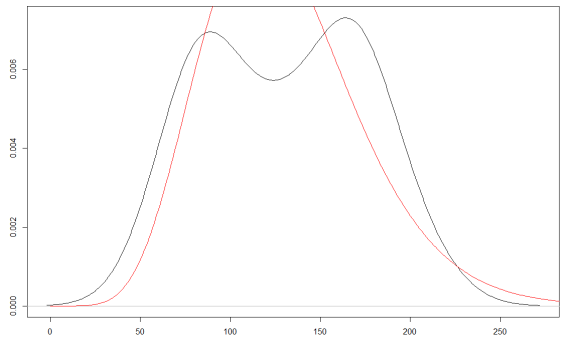

FIGURE 2.16 – La densité d'une loi Binomiale négative (rouge) superposée à celle du jeu de données

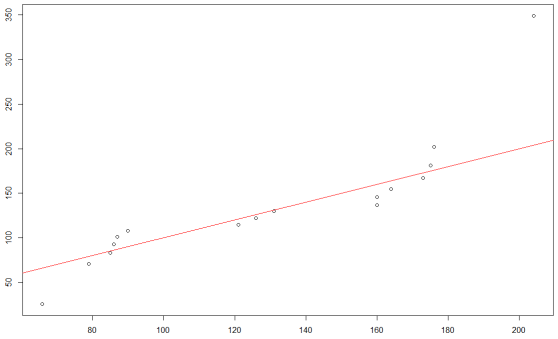

FIGURE 2.17 – QQ-plot de la loi Binomiale négative

Nous voyons que la loi Binomiale négative est mieux ajustée que la loi de Poisson pour la fréquence des sinistres extrêmes. Toutefois elle n'est pas parfaite, mais comme le jeu de données ne contient que 16 années, nous avons considéré que **la loi Binomiale négative est la plus adaptée pour modéliser la fréquence des sinistres extrêmes**.

Cependant, une vérification graphique nous semble un peu limitée pour conclure. Nous décidons donc de faire un test d'adéquation du χ^2.

Test d'adéquation du χ^2

On observe un échantillon de données $y_1, \dots, y_N$ d'une variable aléatoire Y qui prend un nombre fini J de valeurs. On veut tester H_0 : les probabilités que Y prenne les valeurs 1 à J sont respectivement $p_1, \dots, p_J$ avec $\sum_{j=1}^{J} p_j = 1$. On appelle $\hat{p_j}$ la probabilité empirique que Y prenne la valeur j, c'est-à-dire le nombre d'observations qui prennent la valeur j dans l'échantillon divisé par le nombre total d'observations $\mathbf{N}$: $\hat{p_j} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(y_i = j)$.

On peut alors définir la statistique du χ^2 : $T = \sum_{j=1}^{J} \frac{(N\dot{\hat{p}_j} - N\dot{p_j})^2}{N\dot{p_j}}$.

Sous H_0, cette statistique suit asymptotiquement une loi du χ^2 à *J-1* degrés de liberté. On construit alors un test de niveau α en rejetant H_0 lorsque $T \geq F^{-1}_{\chi^2(J-1)}(1-\alpha)$ (avec $F^{-1}_{\chi^2(J-1)}(1-\alpha)$ le quantile d'ordre 1-α de la loi du χ^2à J-1 degrés de libertés). En terme de p-value, on rejette H_0 lorsque p-value $\leq 5\%$.

Nous voulons tester H_0 : *La loi des sinistres extrêmes en fréquence est une loi Binomiale négative* contre H_1 : *La loi des sinistres extrêmes en fréquence n'est pas une loi Binomiale négative.* Grâce à la fonction fitdistr de R, nous ajustons une loi Binomiale négative sur les sinistres extrêmes.

Si $X \sim$ NBin(r,p), alors pour k $\in \mathbb{N}$, $\mathbb{P}(X = k) = \dfrac{\Gamma(r+k)}{k!\Gamma(r)} p^r (1-p)^k$

Ainsi

$$\begin{cases} \mathbb{E}(X) = \dfrac{r(1-p)}{p} \\ Var(X) = \dfrac{r(1-p)}{p^2} \end{cases}$$

En utilisant la méthode des moments, on obtient comme estimateurs de r et p :

$$\begin{cases} \hat{r} = \dfrac{\bar{x_n}^2}{S^2 - \bar{x_n}} \\ \hat{p} = \dfrac{\bar{x_n}}{S^2} \end{cases}$$

, avec $\bar{x_n} = \sum_{i=1}^{n} x_i$, $S^2 = \dfrac{1}{n-1} \sum_{i=1}^{n} (x_i - \mu)^2$, x_i étant la $i^{ème}$ observation et $\mu = \mathbb{E}(X)$.

Méthode des moments

Soit X une variable aléatoire telle que $\mathbb{E}(\mid X \mid^p) < \infty$ pour un certain p $\in \mathbb{N}^*$. $\forall$k $\in \{1 \dots p\}$ on appelle **moment d'ordre k** la quantité $\mathbb{E}(X^k)$ que l'on appelle μ_k.

Soit $(x_1 \dots x_n)$ n observations issues de n variables aléatoires iid de loi P_θ, où θ est inconnu. On suppose :

1. $\exists$ p $\in \mathbb{E}_\theta(\mid X \mid^p) < \infty \quad \forall\theta$
2. $\exists \varphi : \mathbb{R}^k \to \mathbb{R}$ tel que $\theta = \varphi(\mu_1 \dots \mu_k) \quad \forall\theta$

Alors, l'estimateur des moments de θ est donné par

$$\hat{\theta}_n = \varphi(\hat{\mu}_{1,n} \dots \hat{\mu}_{k,n}), \text{ où } \quad \hat{\mu_{j,n}} = \frac{1}{n} \sum_{i=1}^{n} X_i^j \quad \forall j \in \{1 \dots k\}$$

Nous pouvons ainsi estimer les paramètres de la loi Binomiale négative : $\hat{r} = 9.53$ et $\hat{p} = 0.068$. Nous simulons un échantillon de taille 16 puisque le jeu de données ne possède que 16 années. Nous créons ensuite un tableau de contingence avec le nombre de sinistres extrêmes observés par année et le nombre de sinistres simulés par année.

```
        Pearson's Chi-squared test

data:  tableaucontingence
X-squared = 224, df = 210, p-value = 0.2417
```

FIGURE 2.18 – Test du χ^2

Nous obtenons une p-value de 24.17% > 5%, donc nous acceptons H_0 : **La loi en fréquence des sinistres extrêmes est une loi Binomiale négative de paramètres (9.53, 0.068)**.

Loi des coûts

Puisque nous avons affaire à un dépassement de seuil, nous pensons à la théorie des valeurs extrêmes pour simuler la loi des coûts des sinistres extrêmes. Tout d'abord, nous pensons à une GEV (*Generelized Extreme Value* ou loi d'extrêmum généralisée).

Définition d'une GEV

$X \sim GEV(\mu, \sigma, \kappa)$ avec $\mu > 0$ si sa fonction de répartition est donnée par :

$$F(x,\mu,\sigma,\kappa) = \begin{cases} \exp(-(1+\kappa\cdot\frac{x-\mu}{\sigma})^{-1/\kappa}) & pour \kappa \neq 0 \; et \; 1+\kappa\cdot\frac{x-\mu}{\sigma} > 0 \\ \exp(-\exp(-\frac{x-\mu}{\sigma})) & pour \kappa = 0 \; et \; x \in \mathbb{R} \end{cases}$$

Théorème de Fisher Tippett

Soient $X_1, ..., X_n$ n variables aléatoires indépendantes et identiquement distribuées et soit $M_n = \max(X_1, ..., X_n)$. S'il existe une séquence de nombres réels (a_n, b_n) tel que :

$$\begin{cases} a_n > 0 \quad \text{et} \\ \lim\limits_{n\to+\infty} P(\frac{M_n - b_n}{a_n} \leq x) = F(x), \end{cases}$$

Alors F est la fonction de répartition d'une GEV.

Nous pouvons supposer que les hypothèses du théorème de Fisher Tippett sont valides car :
- Les sinistres sont indépendants entre eux car il n'y a aucune raison pour qu'ils soient dépendants les uns des autres
- Une correction est apportée aux sinistres pour certaines années (via la réévaluation). Nous pouvons donc supposer que les sinistres sont identiquement distribués.

Nous allons utiliser les packages *evmix* pour simuler un échantillon de loi GPD et *evd* pour simuler des lois GEV et GPD. Ces packages utilisent la méthode du maximum de vraisemblance, pour déterminer les paramètres des lois. La différence entre ces deux package est que le package *evd* utilise la méthode des maximas par blocs alors que l'autre non.

Définition

Soit $X = (X_i)_{i\in\{1...n\}}$ une suite de variables aléatoires iid de loi $\mathbb{P}_\theta$ et $(x_i \dots x_n)$, n observations de X. On définit la **vraisemblance** de l'échantillon par :

$$\begin{aligned} V(., x_1 \dots x_n) \quad & \mathbb{R} \longrightarrow \mathbb{R} \\ & \theta \mapsto \mathbb{P}_\theta(X_1 = x_1, \dots, X_n = x_n) \end{aligned}$$

Par indépendance, $V(\theta, x_1 \dots x_n) = \prod_{i=1}^n \mathbb{P}_\theta(X_1 = x_i)$

L'**estimateur du maximum de vraisemblance** $\hat{\theta_n}$ est la valeur de θ pour laquelle $V(\theta, x_1 \dots x_n)$ est maximale :

$$V(\hat{\theta_n}, x_1 \dots x_n) \geq V(\theta, x_1 \dots x_n) \quad \forall\theta$$

Simulation de GEV

Afin de calibrer une GEV à notre échantillon, nous utilisons la méthode des maximas par blocs. Nous créons une matrice de taille 41×50 qui comprend tous les sinistres extrêmes (41 étant la taille des blocs et 50 le nombre de blocs). Avec la fonction *apply* de R, nous sélectionnons le maximum de chaque bloc. En voici une représentation graphique :

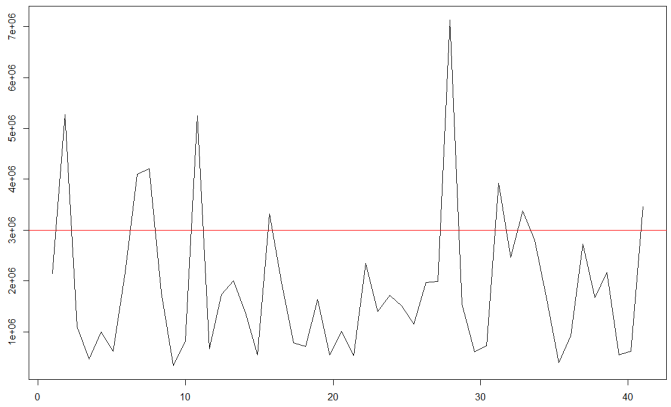

FIGURE 2.19 – Dépassement de seuil

En calibrant une GEV sur ces maximums, nous obtenons :

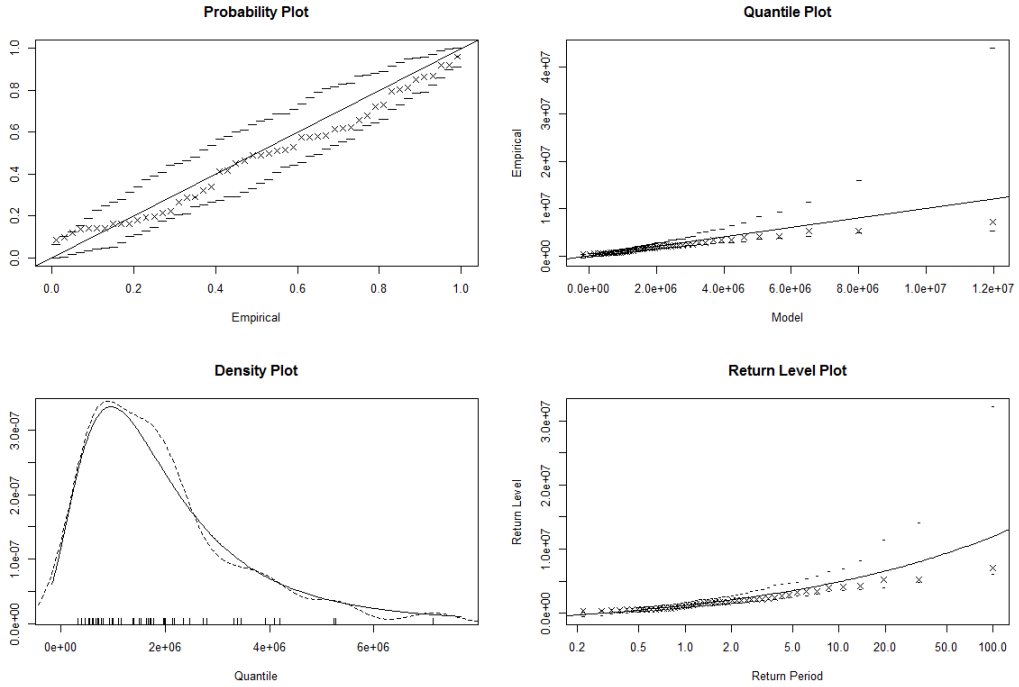

FIGURE 2.20 – Ensemble de graphiques d'une GEV

Le *Probabilty Plot* représente le graphe de la fonction de répartition empirique des données et celle d'une GEV avec les paramètres attribués par le modèle. Il s'agit des couples $(G_n(x_i), G(x_i))$ où G_n est la fonction de répartition empirique des données, G est la fonction de répartition de la GEV calibrée et $x_i,\ i = \{1, \ldots, 50\}$ les différents maximums. Nous voyons que la fonction de répartition empirique des données converge vers la fonction de répartition de la GEV calibrée.

Le *Quantile Plot* fonctionne de la même manière que le *Probability Plot*. La différence majeure est qu'il prend les quantiles des données et ceux de la GEV calibrée au lieu des fonctions de répartition : $(Q_n(x_i), Q(x_i))$ où Q_n est le quantile empirique des données, Q est la fonction quantile de la GEV calibrée et $x_i,\ i = \{1, \ldots, 50\}$ les différents maximums. Nous voyons, ici, que les quantiles des données sont relativement proches de ceux de la GEV.

Le *Density Plot* montre la densité de la fonction de répartition empirique des données et celle d'une GEV avec les paramètres attribués par le modèle, $(f_n(x_i), f(x_i))$ où f_n est la densité empirique des données, f est la densité de la GEV calibrée. Nous pouvons voir que la densité de la GEV coïncide avec celle des données.

Il n'est pas utile d'interpréter le dernier graphique (*Return Level Plot*). En effet, nos sinistres ne sont pas classés par année de survenance, donc le *Return Plot* n'a aucun sens ici.

À la vue des graphiques précédents, on peut supposer que la distribution des sinistres extrêmes est une GEV. Regardons maintenant le résultat de la simulation et comparons l'histogramme des montants des sinistres extrêmes avec celui des montants simulés :

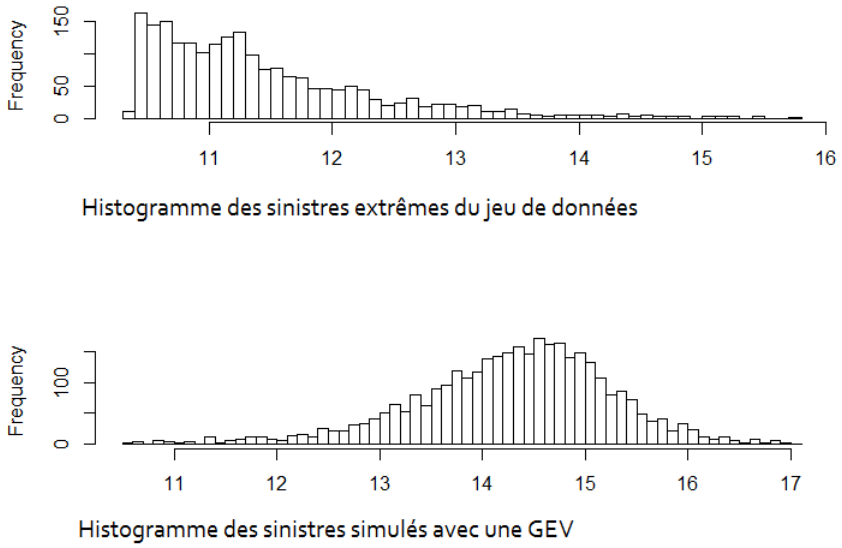

FIGURE 2.21 – Comparaison des histogrammes entre les données simulées et la simulation

Malgré de bons graphiques d'interprétation pour la GEV, lorsque nous observons ces deux histogrammes, on ne peut que conclure que la simulation n'est pas correcte. En effet, la GEV sous-estime, dans un premier temps, les montants des sinistres extrêmes, puis, les surestime énormément. Par la suite, nous testerons l'utilisation de la GPD (*Generelized Pareto Distribution*).

Définition d'une GPD $X \sim GPD(\sigma, \kappa)$ avec $\sigma > 0$ si sa fonction de répartition est donnée par :

$$F(x, \sigma, \kappa) = \begin{cases} 1 - (1 + \frac{\kappa}{\sigma}.x)^{-1/\kappa} & pour \kappa \neq 0, 1 + \kappa \cdot \frac{x}{\sigma} > 0 \ et \ x > 0 \\ 1 - \exp(-\frac{x}{\sigma}) & pour \kappa = 0 \ et \ x > 0 \end{cases}$$

Simulation d'une GPD

1. *Avec le package evmix*

   Afin de calibrer une GPD à nos données, nous avons utilisé la fonction *fgpd*, qui nous permet d'ajuster une loi de Pareto, en lui indiquant le seuil à partir duquel on souhaite appliquer la loi. Les paramètres κ et σ de la loi sont estimés selon la méthode du maximum de vraisemblance. Nous lui avons indiqué d'ajuster la loi sur les coûts dépassant le quantile à 95%, c'est-à-dire 32 669€. Le QQ-plot nous renvoie le résultat suivant :

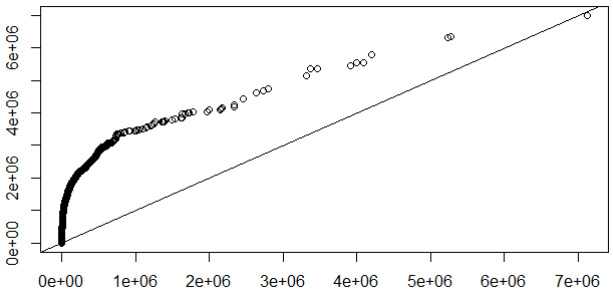

FIGURE 2.22 – QQ-plot obtenu avec une GPD (evmix)

De la même manière que précédemment, comparons les histogrammes des montants extrêmes et des montants simulés :

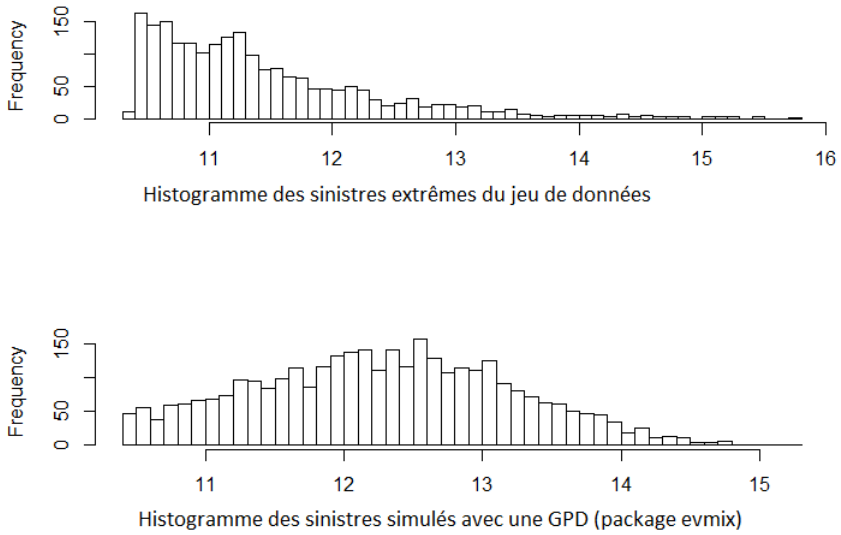

FIGURE 2.23 – Histogrammes des simulations par evmix et des coûts du jeu de données

Au regard des graphes ci-dessus, nous pouvons déjà émettre l'hypothèse que l'ajustement obtenu avec le package *evmix* n'est pas correct.

2. *Avec le package evd*

   Le package *evd* de R nous permet également d'ajuster une GPD. Ainsi, nous calibrons une GPD sur nos données avec ce package. Ci-dessous les graphiques d'interprétation :

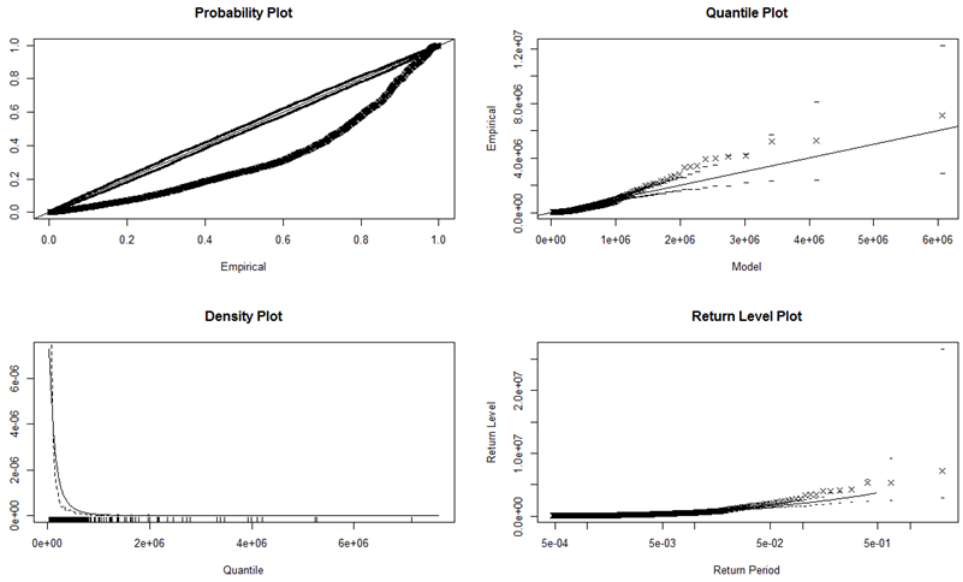

FIGURE 2.24 – Ensemble de graphiques d'une GPD

Le *Probability Plot* n'est pas du tout bien ajusté, la fonction de répartition de la GPD calibrée n'approche pas du tout celle des données.

Le *Quantile Plot* reste correct, les points se trouvent dans un intervalle de confiance assez restreint autour de la $1^{ère}$ bissectrice. Nous pouvons donc dire que les quantiles des données sont relativement proches de ceux de la GPD.

Le *Density Plot* reste correcte également, la densité de la GPD et celle des données sont quasiment superposables.

Comme précédemment, le *Return Level Plot* n'a pas de sens.

Tous ces graphiques ne nous permettent pas d'affirmer que le coût des sinistres extrêmes suit une GPD(σ, κ). Comparons tout de même les histogrammes des montants extrêmes et des montants simulés :

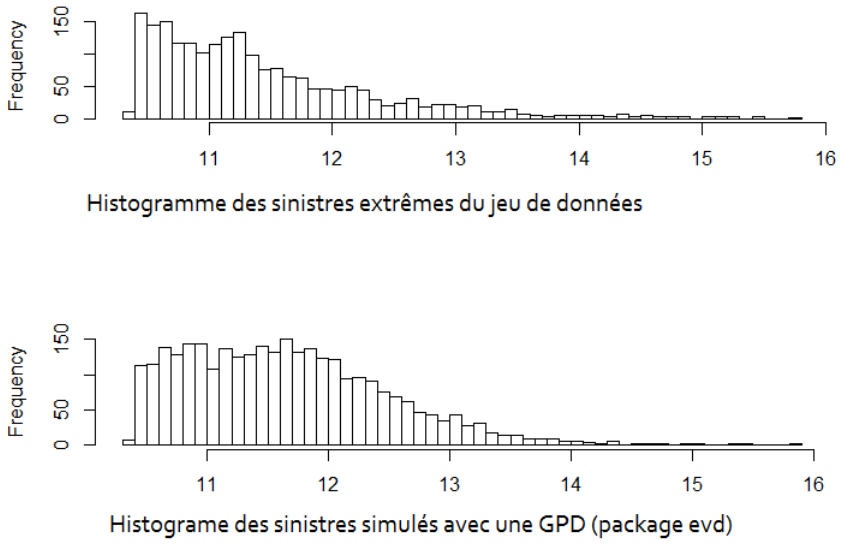

FIGURE 2.25 – Ensemble de graphiques d'une GPD

Nous pouvons voir, que malgré des graphiques douteux, la GPD avec *evd* semble la plus adaptée et la plus cohérente vis-à-vis des données. Le fait que les graphiques ne corroborent pas cette hypothèse vient peut-être du seuil choisi pour la calibrer. Pour vérifier notre choix du seuil, nous traçons un *MRLPLOT* qui nous montre l'erreur de modélisation en fonction du seuil choisi.

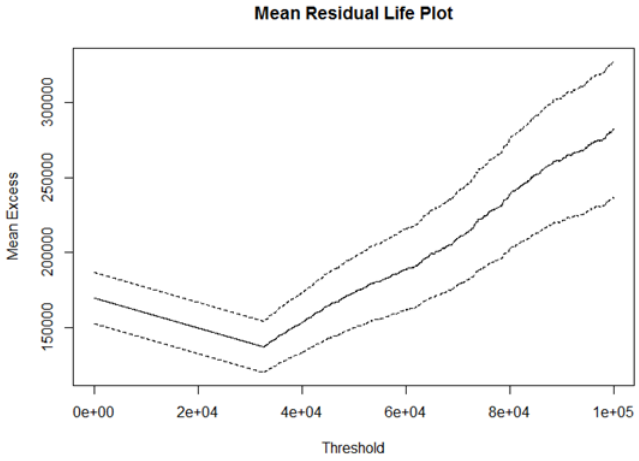

FIGURE 2.26 – Mean Residual Life Plot

Le seuil a effectivement l'air de se trouver au niveau de 32 800€. Nous effectuons un zoom :

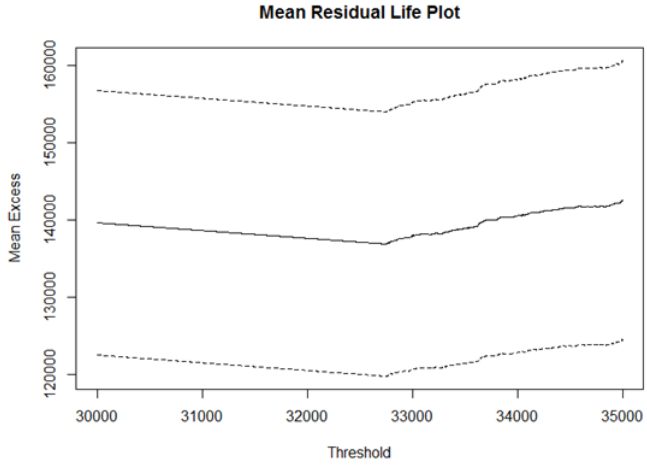

FIGURE 2.27 – Mean Residual Life Plot avec un zoom

Ce graphique montre bien que le seuil que nous avions choisi était le bon : 32 699€.

Nous avons pensé à utiliser encore une fois le test de Kolmogorv-Smirnov mais le principal défaut de celui-ci est de ne pas être très efficace dans les queues de distribution. Nous ne l'appliquons donc pas.

Nous validons la GPD de paramètres $sigma$ **= 136 941.9 et** $kappa$ **= 0.329**.

Toutes ces simulations nous montrent bien que la meilleure manière d'exprimer le montant des sinistres est de passer par l'approximation de Wilson-Hilferty pour les sinistres attritionnels, et une GPD avec *evd* pour les sinistres extrêmes. On appelle :

$$\begin{cases} S_{simu}\text{, le montant total des sinistres,} \\ WH_{attri}\text{, le montant des sinistres attritionnels,} \\ S_{extreme}\text{, le montant des sinistres extrêmes.} \end{cases}$$

Avec ces notations, $S_{simu} = WH_{attri} + S_{extreme}$.

Chapitre 3

Méthodologie de la tarification en assurance

Afin de valider le choix de la loi que suivent les montants des sinistres extrêmes, nous allons faire le rapport entre la tarification faite avec une loi simulée et la tarification faite avec les données en notre possession. Pour comparer les 2 tarifications, nous nous plaçons dans le cas où le réassureur couvre à peu près l'ensemble des sinistres. La couverture de réassurance sur laquelle nous travaillons est une couverture en excédent de sinistres (notée
XS). Tout au long de ce projet, seul ce type de réassurance sera mis en oeuvre. La matrice XS est la suivante :

	Portee	Priorite	Prime Reconstitution	Clause Reconstitution
Réassureur	1e+07	100	1000	0.5

FIGURE 3.1 – Matrice XS avec 1 réassureur qui couvre à peu près l'ensemble des sinistres

Nous simulons des sinistres extrêmes sur 200 années, avec les différentes lois vues précédemment, et nous prenons la moyenne de la prise en charge du réassureur sur les 200 années comme tarif du contrat XS. Nous calculons également le tarif du contrat XS avec les sinistres extrêmes du jeu de données : **pour cela, nous divisons la prise en charge du réassureur sur l'ensemble du jeu de données par 16 (le nombre d'années)**. Nous avons ensuite fait le rapport entre les 2 tarifs pour chaque loi. Nous gardons la loi pour laquelle le rapport est le plus proche de 1 car il s'agira de la loi pour laquelle le tarif XS est le plus proche de celui réalisé avec les données. Voici un tableau récapitulatif des différents rapports en fonction de la loi choisie :

Loi	Rapport
GEV	10.0567
GPD (evmix)	1.140354
GPD (evd)	1.07493

Ce tableau confirme notre choix sur la GPD calibrée avec le package *evd* de paramètres $\sigma = 136\ 941.9$ et $\kappa = 0.329$.

Dans un second temps, nous choisissons un nombre quelconque de réassureurs. Nous sommes en mesure de tarifer un contrat XS, toujours grâce au jeu de données. Par exemple, en prenant 4 réassureurs et la matrice XS :

Réassureurs	Portée	Priorité	Prime de reconstitution	Clause de reconstitution
Réassureur 1	67 000	33 000	670	0.5
Réassureur 2	400 000	100 000	4 000	0.5
Réassureur 3	500 000	500 000	5 000	0.5
Réassureur 4	1 000 000	1 000 000	10 000	0.5

La fonction *TarifXS* que nous avons créée renvoie le tarif du contrat XS pour 1 an, le montant à la charge de l'assureur, le montant des coûts de reconstitution et les montants pris en charge par chaque réassureur. De plus, nous avons appliqué une marge commerciale de 5% pour les réassureurs. Nous avons mis les résultats dans un tableau :

	Montant pris en charge
Tarif XS 1 an	16 843 759
Assureur (coût de reconstitution inclus)	97 809 466
Coût de reconstitution	1 283 334
Réassureur 1	81 269 159
Réassureur 2	108 477 756
Réassureur 3	34 422 677
Réassureur 4	32 497 215

Nous avons également créé une fonction *Montantpris* qui renvoie la simulation d'une année, c'est-à-dire les sinistres extrêmes et les sinistres attritionnels. Nous obtenons les prises en charge de l'assureur et des réassureurs. Toujours avec la même matrice XS, *Montantpris(K)* nous renvoie :

	Montant pris en charge
Assureur (coût de reconstitution inclus)	26 028 693.3
Coût de reconstitution	132 739.5
Réassureur 1	7 622 812.9
Réassureur 2	14 781 939.1
Réassureur 3	3 962 472.7
Réassureur 4	180 668

Maintenant que nous avons calculé le montant pris en charge par chaque acteur, nous pouvons passer à la phase d'optimisation de la couverture.

Chapitre 4

Optimisation

L'optimisation de la couverture de réassurance consiste à rendre le tarif du contrat en excédent de sinistres le plus faible possible.

4.1 Démonstration pour le choix du nombre de réassureurs

Un réassureur ne peut pas forcément supporter l'ensemble de la couverture voulue par l'assureur. C'est pourquoi, il est parfois nécessaire pour un assureur de choisir plusieurs réassureurs. Cependant, nous pouvons nous poser la question quant à l'efficacité de ce découpage de la couverture. En effet, les coûts de reconstitution pourraient alors être supérieurs avec plusieurs réassureurs plutôt qu'avec un unique réassureur. Nous allons montrer que le fait d'avoir 1 ou n réassureurs n'influe pas sur le coût de réassurance, c'est-à-dire le coût de reconstitution.

Nous allons considérer "Réassureur 1" qui va supporter toute la couverture désirée par l'assureur, et n réassureurs qui vont supporter la même couverture, mais découpée en intervalles qui ne se chevauchent pas.
Nous supposons la clause de reconstitution Cl identique pour tous les réassureurs, et la prime de reconstitution est égale à 1% de la portée de chaque contrat.

Soit $n \in \mathbb{N}$, le nombre de réassureurs désiré.

- **Les différents contrats des réassureurs :**

  - Réassureur 1 : $Portée \quad XS \quad Priorité$

  - Réassureur n1 : $\frac{Portée}{n} \quad XS \quad Priorité$

    Réassureur n2 : $\frac{Portée}{n} \quad XS \quad Priorité + \frac{Portée}{n}$

    ...

    Réassureur ni : $\frac{Portée}{n} \quad XS \quad Priorité + (i-1) * \frac{Portée}{n}$

    ...

    Réassureur nn : $\frac{Portée}{n} \quad XS \quad Priorité + (n-1) * \frac{Portée}{n}$

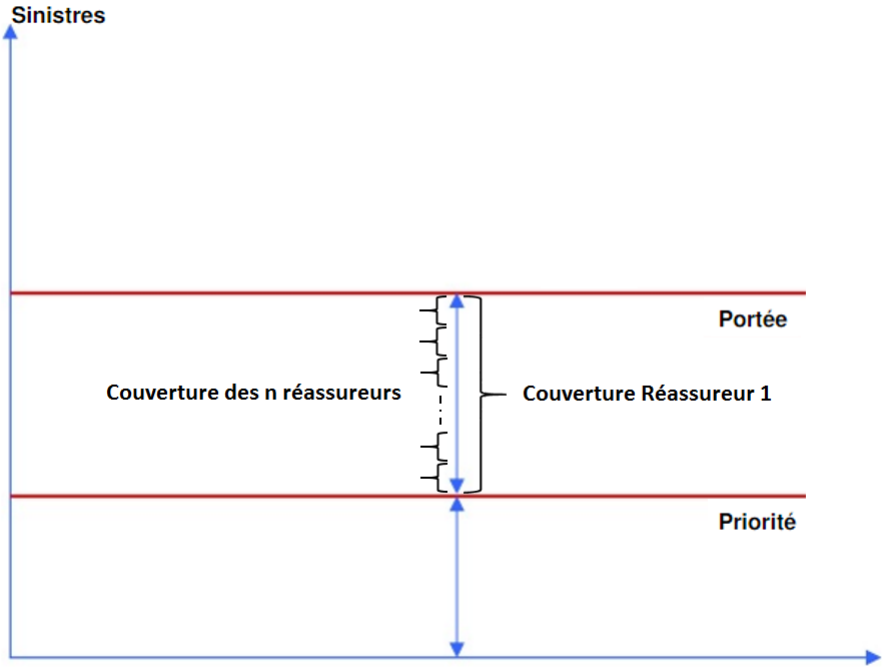

FIGURE 4.1 – Découpe de la couverture avec les n réassureurs

- 1er cas : $S > Priorité + Portée$:

  - Charge réasssureur 1 = $CRéa1$ = Portée

  - $\forall\ i\ \in [\![1, n]\!]$, Charge réassureur ni = $CRéa$ni = $\dfrac{Portée}{n}$

    Cout de reconstitution réassureur 1 = $Rec1$
    Cout de reconstitution réassureur ni = Recni

  - $Rec1 = 0.01 * Portée * Cl * \dfrac{Portée}{Portée} = 0.01 * Portée * Cl$

  - $\forall\ i\ \in [\![1, n]\!]$, $Rec\text{ni} = 0.01 * \dfrac{Portée}{n} * Cl * \dfrac{Portée/n}{Portée/n} = 0.01 * \dfrac{Portée}{n} * Cl$

    $$\boxed{\sum_{i=1}^{n} Rec\text{ni} = 0.01 * Portée * Cl = Rec1}$$

- $\forall\ k\ \in [\![1, n]\!]$, k$^{\text{ème}}$ cas : $Priorité + \dfrac{k * Portée}{n} \geq S > Priorité + \dfrac{(k-1) * Portée}{n}$:

  - Charge réasssureur 1 = $CRéa1 = S - Priorité$

  - $\forall\ j\ \in [\![1, k-1]\!]$, Charge réassureur nj = $CRéa$nj = $\dfrac{Portée}{n}$

    Charge réassureur nk = $CRéa$nk = $S - (Priorité + (k-1) * \dfrac{Portée}{n})$

    $\forall\ j\ \in [\![k+1, n]\!]$, Charge réassureur nj = $CRéa$nj = 0

Cout de reconstitution réassureur 1 = $Rec1$
Cout de reconstitution réassureur ni = Recni

- $Rec1 = 0.01 * Portée * Cl * \dfrac{S - Priorité}{Portée} = 0.01 * Cl * (S - Priorité)$
- $\forall\, j \in [\![1, k-1]\!]$, $Rec\text{nj} = 0.01 * \dfrac{Portée}{n} * Cl * \dfrac{Portée/n}{Portée/n} = 0.01 * \dfrac{Portée}{n} * Cl$

$$Rec\text{nk} = 0.01 * \frac{Portée}{n} * Cl * \frac{S - Priorité - (k-1) * Portée/n}{Portée/n}$$
$$= 0.01 * Cl * (S - Priorité - (k-1) * \frac{Portée}{n})$$

$\forall\, j \in [\![k+1, n]\!]$, $Rec\text{nj} = 0$

$$\sum_{i=1}^{n} Rec\text{ni} = \sum_{j=1}^{k-1} Rec\text{nj} + Rec\text{nk}$$
$$= (\sum_{j=1}^{k-1} 0.01 * \frac{Portée}{n} * Cl) + 0.01 * Cl * (S - Priorité - (k-1) * \frac{Portée}{n})$$
$$= (\sum_{j=1}^{k-1} 0.01 * \frac{Portée}{n} * Cl) + 0.01 * \frac{Portée}{n} * Cl + 0.01 * Cl * (S - Priorité - k * \frac{Portée}{n})$$
$$= k * 0.01 * \frac{Portée}{n} * Cl + 0.01 * Cl * (S - Priorité - k * \frac{Portée}{n})$$
$$= 0.01 * Cl * (S - Priorité) = Rec1$$

- Dernier cas : $Priorité > S$:
  - Charge réasssureur 1 = $CRéa1 = 0$
  - $\forall\, i \in [\![1, n]\!]$, Charge réassureur ni = $CRéa\text{ni} = 0$
  - $Rec1 = 0$
  - $\forall\, i \in [\![1, n]\!]$, $Rec\text{ni} = 0$

$$\sum_{i=1}^{n} Rec\text{ni} = 0 = Rec1$$

$$\boldsymbol{Donc,\ \forall S\ et\ \forall n,\ Rec1 = \sum_{i=1}^{n} Rec\text{ni}}$$

4.2 Optimisation de la couverture de réassurance non proportionnelle

Un assureur, au cours d'une année, définit le montant maximum de pertes N qu'il peut subir avec une probabilité de p. Dans les problématiques actuarielles actuelles en réassurance, la sinistralité attritionnelle n'est jamais simulée ligne à ligne. Seule la sinistralité atypique sera importante. En effet, les contrats de réassurance ne sont appliqués que sur ces sinistres extrêmes (seuil de 32 699€ dans notre cas). Ainsi, l'assureur prendra obligatoirement en charge les sinistres attritionnels.

Soit $p = 95\%$ la probabilité choisie par l'assureur pour l'optimisation de son contrat. Nous devons trouver le montant pour lequel p% des sinistres attritionnels cumulés soient inférieurs à ce dernier, c'est à dire trouver le quantile à p%. Les sinistres attritionnels cumulés sont obtenus avec la fonction WH (méthode de Wilson-Hilferty). À l'aide d'une fonction nommée *funCalculQuant*, le quantile trouvé vaut 14 416 084€. Nous comprenons donc que, avec une probabilité de 95%, l'assureur devra prendre en charge ce montant, puisque les sinistres attritionnels sont entièrement à la charge de ce dernier. Ainsi, l'assureur doit être prêt à perdre 14 416 084€ + un certain montant qu'il choisira pour la prise en charge des sinistres extrêmes.

La société de réassurance propose à la compagnie d'assurance de fixer, au choix, la priorité ou la portée pour l'optimisation du contrat.

4.2.1 Travail préliminaire

Premièrement, nous avons créé une matrice comprenant 500 années de simulations de sinistres extrêmes (une colonne représente une année). Il est possible d'augmenter le nombre d'années simulées pour obtenir encore plus de précision, mais l'optimisation sera plus longue. La matrice comprenant toutes ces simulations est *SIMU222*.

Nous avons aussi créé un vecteur contenant 500 années de simulations des sinistres attritionnels cumulés avec Wilson-Hilferty. Le vecteur comprenant toutes ces simulations est *SIMUATTRI222*.

Nous avons utilisé 2 méthodes d'optimisation :
— Méthode avec la fonction *Optimize* de R
— Méthode avec un algorithme de dichotomie que nous avons créé.

4.2.2 Optimisation avec *Optimize*

En fixant la priorité

L'assureur définit le montant maximum N (25M€ dans notre code) de prise en charge avec une probabilité p, et choisit également une priorité. Ainsi, l'assureur accepte de prendre en charge $n = N - QuantAttri$ € pour les sinistres extrêmes (environ 10.5M€ dans notre code). À partir de cette priorité, l'objectif est de déterminer la portée minimale avec laquelle la prise en charge de l'assureur est inférieure à N avec une probabilité p.

Cependant il faut que la priorité choisie par l'assureur ne soit pas trop grande. En effet, si la priorité est trop élevée, alors même si le contrat comprend une portée infinie, les conditions ne pourront pas être respectées.
C'est pourquoi nous avons créé une fonction *RecherchePriorite* qui prend comme argument une priorité donnée, et si cette priorité satisfait les conditions désirées avec une portée infinie, alors la fonction va renvoyer cette même priorité, sinon elle renverra 0. *Optimize* va alors maximiser cette fonction.

Cependant, nous avons remarqué qu'*Optimize* ne fonctionne pas toujours. Tout d'abord, il faut sélectionner un intervalle de recherche pour la priorité. Donc si la priorité maximale n'est pas dans l'intervalle que nous avons indiqué à *Optimize*, alors elle renverra la borne supérieure de l'intervalle de recherche. Ainsi la première solution qui nous vient à l'esprit serait de mettre un intervalle très grand. Mais là encore, la fonction *Optimize* a ses limites. En effet, si la borne supérieure de l'intervalle de recherche est trop éloignée de la valeur que nous désirons trouver (la priorité maximale ici), alors *Optimize* ne va pas fonctionner. Pour pallier ce problème, nous avons mis en place un système de contrôle. Pour être sur que la fonction *Optimize* ait bien fonctionné, il faut vérifier que la borne supérieure de l'intervalle de recherche est différente de la valeur *...$maximum* trouvée, et que la valeur *...$maximum* soit égale à la valeur *...$objective*. Si ces 2 conditions sont respectées, alors cela signifie que la fonction *Optimize* a bien fonctionné et nous a renvoyé la bonne valeur.
C'est pourquoi nous avons créé un programme qui va augmenter l'intervalle de recherche petit à petit jusqu'à ce que les conditions soient respectées. Nous avons cependant gardé la fonction *RecherchePriorite* car il est possible d'anticiper dans quel intervalle se situera la priorité maximale, et utiliser directement cette fonction sans passer par l'algorithme d'agrandissement de l'intervalle fait gagner un temps précieux. Par exemple avec les conditions précédentes nous trouvons une priorité maximale de 64 488€. L'optimisation de la fonction *RecherchePriorite* met 1 minute 06 à trouver la bonne valeur alors que l'algorithme met 9 minutes 36.

Une fois ce premier travail réalisé, l'assureur peut alors choisir une priorité inférieure à la priorité maximale venant d'être calculée. La fonction *OptiPortee* prend comme argument une portée donnée. Si cette portée satisfait les conditions désirées avec la priorité désirée par l'assureur, alors la fonction va renvoyer cette même portée, sinon elle renverra 10^{15}. *Optimize* va alors minimiser cette fonction, car **minimiser la portée revient à réduire le tarif du contrat**. On vérifie que les 2 conditions de succès d'*Optimize* soient bien respectées pour être sur que la valeur trouvée est correcte (comme ici on minimise la fonction, la 2^{e} condition est nécessaire et suffisante pour le succès d'*Optimize*) :

— borne sup de l'intervalle de recherche $\neq$ valeur *...$minimum*
— valeur *...$maximum* = valeur *...$objective*.

Si la 2^{e} condition est respectée, alors Optimize a bien fonctionné, sinon il faut augmenter l'intervalle de recherche. C'est pourquoi nous avons créé un algorithme agrandissant l'intervalle de recherche petit à petit pour l'optimisation de la portée. Cependant le temps d'exécution de l'op-

timisation est beaucoup plus important avec cet algorithme : on passe de 1 minute 12 à environ 16 minutes. C'est pourquoi l'anticipation de l'intervalle de recherche est une technique clé pour la rapidité de l'optimisation avec *Optimize*. Nous obtenons comme portée optimale 1 184 194 €.

En fixant la portée

L'assureur définit le montant maximum N (25M€ dans notre code) de prise en charge avec une probabilité p, et choisit également une portée. Ainsi l'assureur accepte de prendre en charge $n = N - QuantAttri$ € pour les sinistres extrêmes (environ 10M5€ environ dans notre code). L'objectif est de déterminer la priorité maximale avec laquelle la prise en charge de l'assureur est inférieure à N avec une probabilité p (avec la portée fournie par l'assureur).

Comme lorsqu'on fixe la priorité, son choix n'est pas complètement libre. En effet, pour une portée relativement faible, le montant à charge de l'assureur a un grand risque d'être supérieur à la perte maximale N désirée. Afin de s'assurer qu'il est possible que l'assureur puisse avoir une perte au maximum égale à N avec une probabilité p, il faut lui imposer une portée minimale. En considérant le cas extrême où la priorité est nulle, c'est-à-dire que tous les sinistres extrêmes vont être pris en charge par l'assureur, nous avons déterminé la portée minimale qui va permettre de remplir les conditions imposées par l'assureur (grâce à la fonction *RechercheePortee*, semblable aux fonctions précédentes d'optimisation). Nous n'avons pas rencontré de problème avec *Optimize* pour l'optimisation de cette fonction malgré un grand nombre de valeurs testées. Il n'est donc pas nécessaire de faire un algorithme qui agrandit l'intervalle petit à petit. Le temps d'exécution de l'optimisation de cette fonction est de 1 minute 18. La portée minimale à choisir avec les conditions que nous avions fixées est de 561 801 €.

Par la suite, l'assureur doit choisir une portée supérieure à la portée minimale qui vient d'être calculée. À partir de cette portée, l'objectif est de déterminer la priorité maximale avec laquelle la prise en charge de l'assureur est inférieure à N avec une probabilité p, car **maximiser la priorité revient à réduire le tarif du contrat**.

Nous avons programmé une fonction nommée *OptiPriorite* qui prend comme argument une priorité donnée et qui va nous révéler si cette priorité donnée satisfait les conditions.
Si la priorité satisfait les conditions, alors la fonction va renvoyer la priorité donnée en argument, sinon elle va renvoyer 0. Le but sera alors de maximiser cette fonction.
Cependant il va encore falloir faire attention à un éventuel échec d'optimisation d'*Optimize*. Il va falloir vérifier que les 2 conditions de succès soient bien réalisées :

— borne sup de l'intervalle de recherche ≠ valeur *...$minimum*
— valeur *...$maximum* = valeur *...$objective*

Si les 2 conditions sont réalisées, nous avons trouvé la priorité optimale, sinon, cela veut dire qu'*Optimize* n'a pas fonctionné et il va alors falloir augmenter l'intervalle de recherche. C'est pourquoi nous avons créé un algorithme qui agrandit l'intervalle de recherche petit à petit pour l'optimisation de la fonction *OptiPortee*. Les 2 temps d'exécution sont 1 minute 06 et 6 minutes 36. Dans notre code R, l'assureur choisit une portée égale à la portée minimale trouvée + 10^6, soit 1 561 801 €. La priorité optimale que nous avons trouvé est 44 844 €.

4.2.3 Optimisation avec dichotomie

Suite aux problèmes rencontrés avec la fonction *Optimize*, nous avons décidé d'établir une nouvelle méthode d'optimisation qui ne rencontrerait pas le problème de l'intervalle de recherche. Il s'agit de la *dichotomie*.

La dichotomie est une méthode permettant d'approcher la solution d'une équation de type $f(x) = 0$. Précisément, nous supposons que la fonction f est continue sur un intervalle $[a, b]$, avec $f(a) < 0$ et $f(b) > 0$. On sait donc qu'il existe au moins un réel $c \in [a, b]$ tel que $f(c) = 0$. Le principe est le suivant :

- Si $f\left(\frac{a+b}{2}\right) < 0$ alors $c \in [\frac{a+b}{2}, b]$
- Si $f\left(\frac{a+b}{2}\right) > 0$ alors $c \in [a, \frac{a+b}{2}]$

Concrètement l'algorithme consiste à construire les suites (a_n) et (b_n) comme suit :

- $a_0 = a$ et $b_0 = b$
- Si $f\left(\frac{a_n+b_n}{2}\right) < 0$, alors $a_{n+1} = \frac{a_n+b_n}{2}$ et $b_{n+1} = b_n$
- Sinon, $a_{n+1} = a_n$ et $b_{n+1} = \frac{an+bn}{2}$

Les suites (a_n) et (b_n) converge donc vers c la solution de $f(x) = 0$. L'intervalle $[a_n, b_n]$ est de longueur $\dfrac{b_n - a_n}{2^n}$.

En fixant la priorité

Comme pour l'optimisation avec *Optimize*, nous devons rechercher la priorité maximale que l'assureur peut choisir en respectant les conditions de prise en charge qu'il désire. La fonction *DichoRechPrio* va nous renvoyer cette valeur, en utilisant un algorithme de dichotomie. Avec cette méthode, nous ne rencontrons plus le problème d'intervalle de recherche. Nous trouvons comme priorité maximale possible 64 887 €. Le temps d'exécution de cette fonction est de 1 minute 18.

L'assureur peut alors choisir une priorité, qui est inférieure à la priorité maximale. Nous avons créé la fonction *DichoOptiPortee* qui va renvoyer la portée optimale pour la priorité choisie. Nous obtenons 1 184 219 €. Son temps d'exécution est de 1 minute 18.

En fixant la portée

Premièrement nous devons rechercher la portée minimale que l'assureur peut choisir avec une priorité nulle. La fonction *DichoRechPortee* nous permet de connaitre cette valeur. Avec les conditions que nous avions fixées, nous obtenons comme portée minimale 562 296 €. Son temps d'exécution est de 1 minute 18.

L'assureur peut maintenant choisir une portée supérieure à la portée minimale. Dans notre code R, l'assureur choisit une portée égale à la portée minimale trouvée + 10^6, soit 1 562 296 €. La fonction *DichoOptiPriorite* nous renvoie la priorité optimale qui respecte les conditions imposées par l'assureur. Dans notre code, nous obtenons une priorité optimale égale à 44 834 €. Le temps d'exécution de cette fonction est de 1 minute 12.

4.2.4 Comparaison des 2 méthodes

En fixant la priorité

L'assureur a fixé la priorité à 33 000 € dans l'exemple de notre code. Il accepte de prendre en charge 25M€ avec une probabilité de 95%.

Optimize		**Dichotomie**
64 488 €	Priorité maximale	64 887 €
1,1 min ou 9,6 min	Temps d'exécution RecherchePrioriteMax	1,3 min
33 000 €	Priorité choisie	33 000 €
1 184 194 €	Portée optimale	1 184 219 €
1,2 min ou 15,8 min	Temps d'exécution OptiPortee	1.3 min

Nous pouvons voir que les portées optimales obtenues sont très semblables, et que les priorités maximales calculées sont quasiment égales, donc les résultats sont cohérents. Cependant il y a une différence au niveau du temps d'exécution. En effet, pour l'optimisation avec *Optimize*, si nous anticipons le bon intervalle, alors le temps d'exécution est légèrement inférieur au temps d'exécution de la dichotomie. Mais si nous n'anticipons pas l'intervalle de recherche, alors *Optimize* met beaucoup plus de temps à converger vers la portée optimale.

En fixant la portée

L'assureur a fixé la portée à des valeurs différentes en fonction de la méthode choisie, car les portées minimales obtenues étaient légèrement différentes. Il accepte de prendre en charge 25M€ avec une probabilité de 95%.

Optimize		**Dichotomie**
561 801 €	Portée minimale	562 296 €
1,3 min	Temps d'exécution RecherchePorteeMin	1,3 min
1 561 801 €	Portée choisie	1 562 296 €
44 844 €	Priorité optimale	44 834 €
1,1 min ou 6,6 min	Temps d'exécution OptiPriorite	1,2 min

Nous pouvons de nouveau remarquer que les résultats sont très semblables, et même les temps d'exécution sont à peu près semblables si on compare les temps entre Dichotomie et *Optimize* avec anticipation d'intervalle. Néanmoins si on n'anticipe pas l'intervalle de recherche, le temps d'exécution est 6 fois plus long.

Ainsi nous pouvons conclure la comparaison des 2 méthodes en affirmant que l'optimisation par dichotomie est la meilleure dans le cadre de notre bureau d'étude.

4.2.5 Optimisation du choix de l'assureur et Application numérique

Afin de rendre à la fois l'optimisation du tarif du contrat plus satisfaisante et le choix de l'assureur plus pertinant, nous allons prendre en compte une optimisation du choix de l'assureur. Dans le cas où l'assureur choisit sa priorité, dans la limite de la priorité maximale, cela signifie que nous allons tester avec plusieurs choix de priorité (tout en restant dans la limite de priorité maximale). Appliquons tout de suite un nouvel exemple. :

Premier cas : Priorité à choisir par l'assureur → Portée optimale :

Soit un assureur ayant un profil de risque similaire à celui de notre jeu de données. L'assureur est prêt à perdre 35M€ avec une probabilité de 95%. Sous ces conditions, la limite du choix de la priorité est d'environ 149 033€.
Il possède déjà un contrat XS qui remplit ces conditions, ce contrat a une priorité de 50 000€ et une portée de 1.5M€. Voici le contrat actuel :

Portée	**Priorité**	**Prime de reconstitution**	**Clause de reconstitution**
1 500 000	50 000	15 000	0.5

Avec ce contrat, le réassureur prend en charge le montant des sinistres compris entre 50 000€ et 1 550 000€, tandis que l'assureur prend en charge le reste. L'assureur perd 25 089 858 au maximum avec une probabilité de 95% : les conditions sont respectées et il perd en moyenne 16 771 738 . Pour ce contrat, le tarif proposé par le réassureur est de 14 178 319€pour un an.

L'objectif est de trouver le tarif le plus faible en faisant varier la priorité. Il a été choisi de faire varier la priorité avec un pas de 14 600 (une dizaine de contrats sera simulé). Pour chaque priorité considérée, la portée optimale correspondante sera à déterminer, ainsi que le tarif du contrat.

Comparons graphiquement les résultats obtenus.

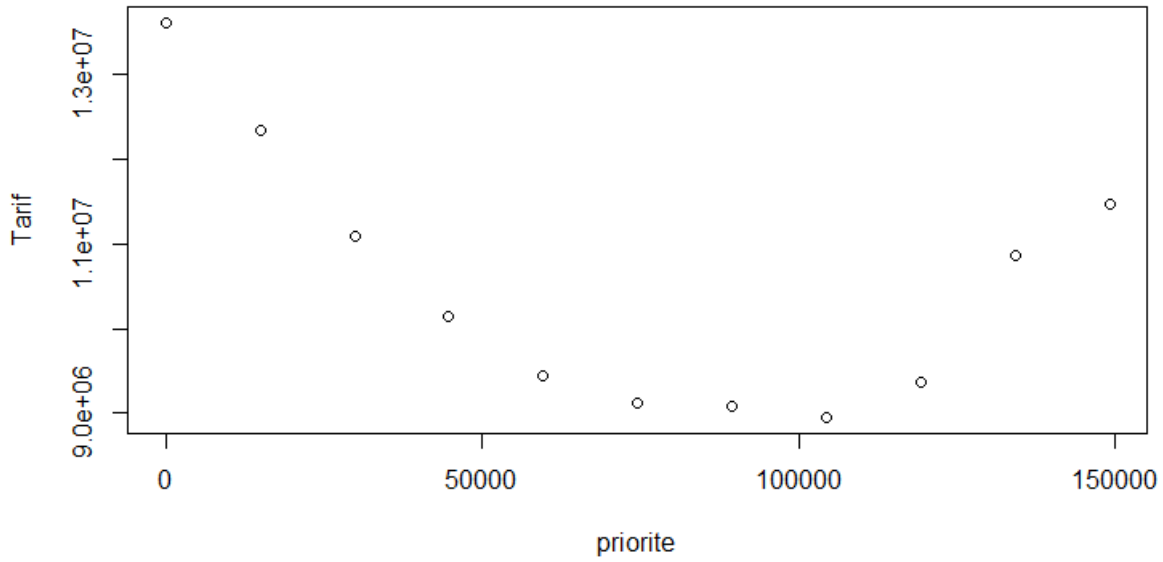

FIGURE 4.2 – Evolution du tarif en fonction de la priorité choisie

Le graphe ci-dessus représente le tarif du contrat en fonction de la priorité. L'assureur a donc intérêt de choisir la priorité qui minimise le tarif qu'il devra verser pour se couvrir. On constate que la priorité optimale est situé entre 100 000 et 115 000€. La valeur trouvée est exactement de 104 323.1€.

La portée optimale calculée avec DichoOptiPortee est de 828 856€. Finalement, le contrat optimal est :

Portée	**Priorité**	**Prime de reconstitution**	**Clause de reconstitution**
828 856	104 323.1	8 288.56	0.5

Avec ce contrat optimisé, le réassureur prend en charge le montant des sinistres compris entre 104 323.1€ et 933 179.1€, tandis que l'assureur prend en charge le reste. L'assureur perd 34 667 731€ au maximum avec une probabilité de 95% : les conditions sont respectées et il perd en moyenne 23 682 025€. Pour ce contrat, le tarif proposé par le réassureur est de 8 956 321.77€ pour un an.

Contrat non optimisé		**Contrat optimisé**
1 500 000	Portée	828 856
50 000	Priorité	104 323.1
14 178 319	Tarif du contrat	8 956 321.77
16 771 738	Prise en charge moyenne de l'assureur	23 682 025
25 089 858	Prise en charge maximale (à 95%)	34 667 731

Le contrat optimisé nous donne un tarif plus satisfaisant tout en respectant les conditions (il y a une différence de 5 221 997.23€).

Deuxième cas : Portée à choisir par l'assureur → Priorité optimale :

Soit un assureur ayant un profil de risque similaire à celui de notre jeu de données. L'assureur est prêt à perdre 35M€ avec une probabilité de 95%. Sous ces conditions, le plancher de portée est de 226 776.6€.
Il possède déjà un contrat XS qui remplit ces conditions, ce contrat a une priorité de 50 000€ et une portée de 1.5M€. Voici le contrat actuel :

Portée	Priorité	Prime de reconstitution	Clause de reconstitution
1 500 000	50 000	15 000	0.5

Avec ce contrat, le réassureur prend en charge le montant des sinistres compris entre 50 000€ et 1 550 000€, tandis que l'assureur prend en charge le reste. L'assureur perd 25 089 858€au maximum avec une probabilité de 95% : les conditions sont respectées et il perd en moyenne 16 771 738€. Pour ce contrat, le tarif proposé par le réassureur est de 14 178 319€ pour un an.

L'objectif est de trouver le tarif le plus faible en faisant varier la portée. Il a été choisi de faire varier la portée avec un pas de $25 * 10^5$(une dizaine de contrats sera simulé). Avec un pas supérieur à celui-ci, le tarif devient constant à partir d'un certain montant de portée. Pour chaque portée considérée, la priorité optimale correspondante sera à déterminer, ainsi que le tarif du contrat.

Comparons graphiquement les résultats obtenus.

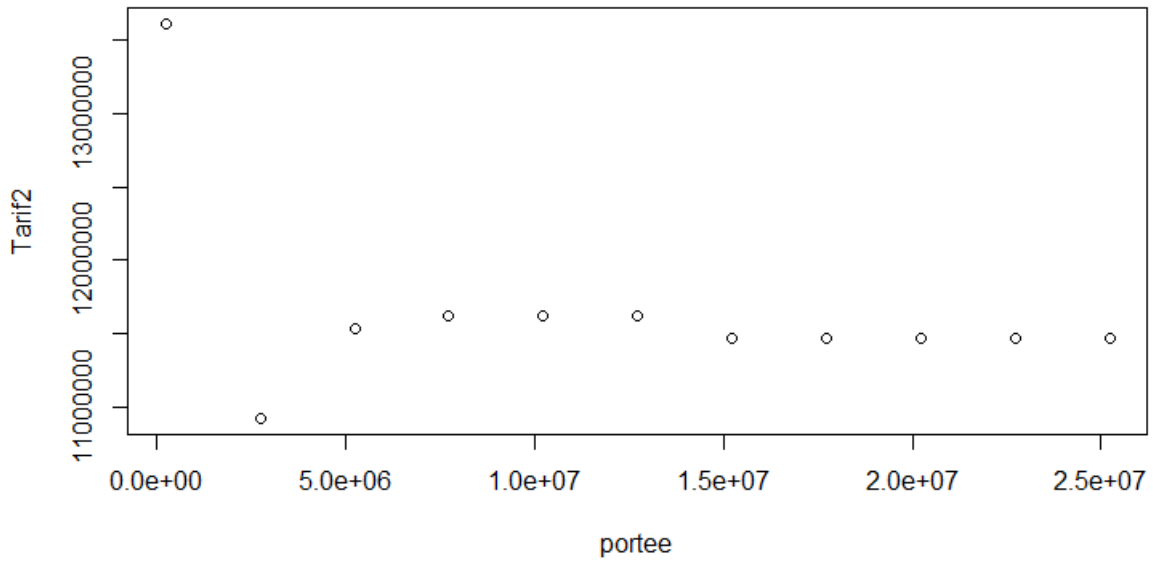

FIGURE 4.3 – Evolution du tarif en fonction de la portée choisie

Le graphe ci-dessus représente le tarif du contrat en fonction de la portée. L'assureur a donc intérêt de choisir la portée qui minimise le tarif qu'il devra verser pour se couvrir. La portée optimale trouvée est exactement de 2 726 776.64€.

La priorité optimale calculée avec DichoOptiPriorite est de 134 809.08€. Finalement, le contrat optimal est :

Portée	Priorité	Prime de reconstitution	Clause de reconstitution
2 726 776.64	134 809.08	27 267.77	0.5

Avec ce contrat optimisé, le réassureur prend en charge le montant des sinistres compris entre 134 809.08€ et 2 861 585.72€, tandis que l'assureur prend en charge le reste. L'assureur perd 34 999 939€ au maximum avec une probabilité de 95% : les conditions sont respectées et il perd en moyenne 23 552 301€. Pour ce contrat, le tarif proposé par le réassureur est de 10 930 457.85€ pour un an.

Contrat non optimisé		Contrat optimisé
1 500 000	Portée	2 726 776.64
50 000	Priorité	134 809.08
14 178 319	Tarif du contrat	10 930 457.85
16 771 738	Prise en charge moyenne de l'assureur	23 552 301
25 089 858	Prise en charge maximale (à 95%)	34 999 939

Le contrat optimisé nous donne un tarif plus satisfaisant tout en respectant les conditions (il y a une différence de 3 247 861.15€).

Conclusion

Grâce à une démarche scientifique, nous avons montré que l'approximation de Wilson-Hilferty était une approche satisfaisante pour simuler la charge cumulée des sinistres attritionnels, de manière fidèle au jeu de données. Aussi, nous avons validé le choix d'une approche fréquence∗sévérité pour les sinistres extrêmes. Ceux-ci sont donc modéliser par une loi binomiale négative pour la fréquence et une GPD pour les coûts.
Par la suite, nous avons exposé la méthode pour tarifer un contrat XS, et déterminé une technique d'optimisation, convenable en terme de rapidité et de résultat.
Si un assureur accepte de perdre N€ avec une probabilité p, l'objectif de notre bureau d'étude est de lui fournir une couverture qui répond à cette contrainte, tout en étant la moins chère possible. Les exemples ci-dessus illustrent ainsi la possibilité d'allier minimisation du tarif et respect des conditions de l'assureur : notre objectif est donc réalisé.

Toutefois une question subsiste : pourquoi lors de l'optimisation du prix de couverture, la somme du tarif et de la prise en charge moyenne est plus importante pour le contrat optimisé que pour le contrat de base ? Pour aller plus loin, une approche judicieuse serait alors de minimiser la somme du tarif du contrat et de l'espérance de prise en charge, tout en respectant les conditions, plutôt que de seulement minimiser le tarif du contrat. Quelques mois de travail supplémentaires nous auraient permis de développer cette approche.

Bibliographie

[1] "Les bases de la réassurance", Gen Re, septembre 2015.

[2] Un cours concis sur les techniques de réassurance, https://fr.scribd.com/doc/32142159/Reassurance.

[3] Conférence de presse : "Le marché de la réassurance en 2015".

[4] Mémoire d'actuariat de Xavier AGENOS, "Appetit pour le risque et gestion stratégique d'une société d'assurance non-vie" http://www.ressources-actuarielles.net.

[5] Pierre Ailliot, cours Théorie des valeurs extrêmes, Euria, 2016.

[6] Brice Franke, Dépendance linéaire, Euria 2016.

[7] Nicolas Pétrélis, cours Inférence statistique, Université de Nantes, 2014.

Annexe

Estimation des paramètres des lois de probabilité

Sinistres attritionnels en fréquence

X désigne la fréquence annuelle des sinistres attritionnels.

- Loi de Poisson :
  $X \sim$ Poisson(λ) avec $\lambda = 2472.375$

- Loi Binomiale négative :
  $X \sim$ NBin(r,p) avec r = 11.908 et p = 0.48%

- Loi Normale :
  $X \sim \mathbb{N}(\mu,\sigma^2)$ avec $\mu = 2473.375$ et σ=733.337, σ étant l'écart-type

Sinistres attritionnels en coût

Avec un mélange de lois Normales et le package MCLUST

Espérance	Variance
7.858	9.409
5.494	0.432
6.645	0.033
7.284	0.018
7.338	0.006
8.068	0.008
7.861	0.085
8.725	0.216
9.073	2.501

Sinistres attritionnels par la méthode de Wilson-Hilferty

$\mu_S = 8792483$
$\sigma_S^2 = 1.066 \cdot 10^{13}$
$\gamma_S = 0.372$

$\mu_Y = 0.985$
$\sigma_Y^2 = 0.016$

$b_1 = 0.001$

b_2 = -16.081
b_3 = 5.381

Sinistres extrêmes en fréquence

X désigne la fréquence annuelle des sinistres extrêmes.

- Loi de Poisson :
  $X \sim$ Poisson(λ) avec λ = 130.188

- Loi Binomiale négative :
  $X \sim$ NBin(r,p) avec r = 9.576 et p = 6.85%

Sinistres extrêmes en coût

X désigne le coût des sinistres extrêmes.

Avec la GEV

$X \sim$ GEV(μ, σ,κ) avec $\mu = 1.237 \cdot 10^6$
$\sigma = 1.135 \cdot 10^6$
$\kappa = 2.831 \cdot 10^{-1}$

Avec la GPD et le package evd

$X \sim$ GPD(σ, κ) avec
σ = 136941.9
κ = 0.329

Avec la GPD et le package evmix

$X \sim$ GPD(σ, κ) avec
σ = 310744.7
κ = 0.145

Codes R

```
################################################################################################
#TRAVAIL EFFECTUE SUR LA BASE DE DONNEES INITIALE
################################################################################################

Z=readRDS("C:/Users/.../base_sinistres.rds")
Z
View(Z)

library(MASS)
library(evd)
library(evmix)
library(mclust)
library(e1071)

################################

#Fonction qui sélectionne les sinistres survenus dans l'année souhaitée
#(on prend leur vraie valeur donc leur dernière réévaluation)

donnees_reev1=function(h,date){
  h=h[h$surv==date,]
  n=length(h$cout)
  a=NULL  #vecteur contenant les sinistres reevaluees
  i=1
  while (i<n){
    if ((h$id[i])!=(h$id[i+1])){a=rbind(a,h[i,])}
    i=i+1
  }
  return(a)
}

t1=Sys.time()

View(donnees_reev1(Z,2000))

t2=Sys.time()
t2-t1  #1.7 sec

################################

#Nombre et Moyenne du coût des sinistres + Pourcentage de faux sinistres
#sur la i-ème année :

cout=function(Z,date){
  Y=donnees_reev1(Z,date)
  N=length(Y[,1])
  Y=Y[Y$cout!=0,]
```

```
  n=length(Y[,1])
  C=0
  for (i in 1:n){
    C=C+Y[i,5]
  }
  C=C/n
  P=(N-n)/N
  V=cbind(C,N)
  V=cbind(V,n)
  V=cbind(V,N-n)
  V=cbind(V,P)
  V=as.data.frame(V)
  colnames(V)[1] <- "Cout moyen"
  colnames(V)[2] <- "Nb de sinistres"
  colnames(V)[3] <- "Nb de vrais sinistres"
  colnames(V)[4] <- "Nb faux sinistres"
  colnames(V)[5] <- "Pourcentage de faux-sinistres"
  return(V)
}

t1=Sys.time()
cout(Z,2000)
t2=Sys.time()
t2-t1  #1.8 sec


t1=Sys.time()

Un=NULL
for (i in 0:15){
  Un=rbind(Un,cout(Z,2000+i))
}
Annee=NULL

for (i in 1:16){Annee[i]=1999+i}
Deux=cbind(Annee,Un)
Deuxx=rbind(mean(Deux[,2]),mean(Deux[,6])*100)  #Pourcentage de faux sinistres
#et du coût par année sur l'ensemble du jeu de données
Deuxx=as.data.frame(Deuxx)
rownames(Deuxx)[1]<- "Nombre moyen de sinistres par année"
rownames(Deuxx)[2]<- "Pourcentage de faux sinistres moyen par année"
View(Deuxx)

t2=Sys.time()
t2-t1  #48 sec

################################

#Nombre moyen du nombre de fois qu'un sinistre est réévalué
```

```
nbréévaluation=function(Z){
  n=length(Z[,1])
  r=0
  N=n
  for (i in 1:(n-1)){
    if ((Z$id[i])==(Z$id[i+1])){
      r=r+1
      N=N-1
    }
  }
  if ((Z$id[n-1])==(Z$id[n])){
    r=r+1
    N=N-1
  }
  return (r/N)
}

t1=Sys.time()

nbréévaluation(Z)

t2=Sys.time()
t2-t1  #10 sec

################################

#Table sans les mêmes sinistres réévalués dans la même année

Reevalan=function(d){
  Y=0
  n=length(d[,1])
  for (i in 1:n-1){
    Y[i]=abs(d[i,1]-d[i+1,1]) + abs(d[i,4]-d[i+1,4])
  }
  Y[n]=1
  G=cbind(Y,d)
  G=G[G[,1]!=0,]
  G=G[,-1]
  return(G)
}

t1=Sys.time()

V=Reevalan(Z)
View(V)

t2=Sys.time()
t2-t1  #8.1 min

################################
```

```
#Fonction qui renvoie la dernière réévaluation de chaque sinistre

Sansreeval=function(d){
  n=length(d[,1])
  Y=NULL
  for (i in 1:(n-1)){
    if (d[i,1] != d[i+1,1]){
      Y=rbind(Y,d[i,])
    }
  }
  return(Y)
}

t1=Sys.time()

U=Sansreeval(Z)
View(U)

t2=Sys.time()
t2-t1  #5.2 min

################################

#Fonction qui renvoie les sinistres dont la dernière
#réévaluation n'est pas nulle

SansCoutNul=function(Z){
  UU=Sansreeval(Z)
  UU=UU[UU$cout!=0,]
  return(UU)
}

t1=Sys.time()
M=SansCoutNul(Z)
t2=Sys.time()
t2-t1  #5.9 min


#Sans devoir repasser par la fonction SansReeval :

UU=U[U$cout!=0,]
View(UU)

################################

#Fonction qui renvoie le cout réel du sinistre pour chaque réévaluation

cout3=function(Z){
  n=length(Z[,1])
```

```
  c=Z$cout
  for (i in 2:n){
    if (Z$id[i-1] == Z$id[i]){
      c[i]=Z$cout[i]-Z$cout[i-1]
    }
  }
  Z=Z[,-5]
  Z=cbind(Z,c)
  Z=as.data.frame((Z))
  colnames(Z)[5] <- "cout"
  return(Z)
}

t1=Sys.time()

View(cout3(Z))

t2=Sys.time()
t2-t1  #22 sec

################################

#MONTANT + NB DE SINISTRES PRIS EN CHARGE POUR CHAQUE REASSUREUR
#ET L'ASSUREUR POUR L'ANNEE N

PaiementAnnee=function(h,date,K){
  d=h[h$date==date,]
  d=Reevalan(d)
  n=length(K[,1])
  k=length(d[,1])
  A=matrix(rep(0,k),ncol=k)
  T=matrix(rep(0,k),ncol=k)
  R=matrix(rep(0,n*k),nrow=k)
  O=rep(0,k)
  for (j in 1:n){  #Si la clause de reconsitution est définie comme nulle
#dans la matrice K, alors elle est automatiquement égale à 1% de la portée
    if (K[j,3]==0){K[j,3]=0.01*K[j,1]}
  }
  for (i in (1:k)){
    T[i]=d$cout[i]
    for (j in (1:n)){
      R[i,j]=min((max(0,T[i]-K[j,2])),K[j,1])
      O[i]=O[i]+K[j,3]*K[j,4]*R[i,j]/K[j,1]
    }
    A[i]=T[i]-sum(R[i,])+O[i]
  }
  montant=cbind(t(A),O)
  montant=cbind(montant,R)
  montant=as.data.frame(montant)
  montantfinal=colSums(montant)
```

```
  montantfinal=as.data.frame(montantfinal)
  colnames(montantfinal)[1] <- "Montant pris en charge"
  rownames(montantfinal)[1] <- "Assureur (coût Recons inclus)"
  rownames(montantfinal)[2] <- "Coût Reconstitutions"
  for (i in 3:length(montant)){
    rownames(montantfinal)[i] <- paste("Réassureur", i-2)
  }
  return(montantfinal)
}

##Construction de la matrice K (matrice XS)

"K[i,]=(C1,C2,CR,Prime)
C1 : portée, C2 : priorité, CR : Clause de reconstitution (0 à 50%),
Prime : Prime de reconstitution (si égale à 0 alors Prime = 1%*portée )"

#La matrice K ne doit pas avoir des contrats XS qui se "recoupent",
#ie qui couvrent plusieurs fois le même intervalle

K=matrix(c(700,300,0,0.5,4000,1000,0,0.5,5000,5000,0,0.5,10000,10000,0,0.5),
ncol=4,byrow=TRUE)
K=as.data.frame(K)
colnames(K)[1] <- "Portee"
colnames(K)[2] <- "Priorite"
colnames(K)[3] <- "Prime Reconstitution"
colnames(K)[4] <- "Clause Reconstitution"
for (i in 1:length(K[,1])){
  rownames(K)[i] <- paste("Réassureur", i)
}

View(K)

View(PaiementAnnee(Z,2000,K))

################################


#MONTANT TOTAL PRIS EN CHARGE PAR L'ASSUREUR ET REASSUREURS SUR LES 16 ANNEES

PaiementTotal=function(d,K){

  #Possibilité de lancer la fonction sans utiliser d=SansCoutNul(d)
  #qui est longue à exécuter
  d=UU
  #d=SansCoutNul(d) si on veut prendre la fonction
  #(mettre un dièse au-dessus et enlever celui de cette ligne)
  n=length(K[,1])
  k=length(d[,1])
  A=matrix(rep(0,k),ncol=k)
  T=matrix(rep(0,k),ncol=k)
```

```
  R=matrix(rep(0,n*k),nrow=k)
  O=rep(0,k)
  for (j in 1:n){  #Si la clause de reconsitution est définie comme nulle
#dans la matrice K, alors elle est automatiquement égale à 1% de la portée
    if (K[j,3]==0){K[j,3]=0.01*K[j,1]}
  }
  for (i in (1:k)){
    T[i]=d$cout[i]
    for (j in (1:n)){
      R[i,j]=min((max(0,T[i]-K[j,2])),K[j,1])
      O[i]=O[i]+K[j,3]*K[j,4]*R[i,j]/K[j,1]
    }
    A[i]=T[i]-sum(R[i,])+O[i]
  }
  montant=cbind(t(A),O,R)
  montant=as.data.frame(montant)
  montantfinal=colSums(montant)
  montantfinal=as.data.frame(montantfinal)
  colnames(montantfinal)[1] <- "Montant pris en charge"
  rownames(montantfinal)[1] <- "Assureur (coût Recons inclus)"
  rownames(montantfinal)[2] <- "Coût Reconstitutions"
  for (i in 3:length(montant)){
    rownames(montantfinal)[i] <- paste("Réassureur", i-2)
  }
  return(montantfinal)
}

t1=Sys.time()
PaiementTotal(Z,K)
t2=Sys.time()
t2-t1  #15 sec

################################



hist(log(UU$cout),breaks=300)


#########################################################################################
#SIMULATION DES SINISTRES
#########################################################################################


"K[i,]=(C1,C2,CR,Prime)
C1 : portée, C2 : priorité, CR : Clause de reconstitution (0 à 50%),
Prime : Prime de reconstitution (si égale à 0 alors Prime = 1%*portée )"

#La matrice K ne doit pas avoir des contrats XS qui se "recoupent",
#ie qui couvrent plusieurs fois le même intervalle
```

```
K=matrix(c(67301,32699,673,0.30,400000,100000,4000,
0.35,500000,500000,5000,0.45,1000000,1000000,10000,0.50),ncol=4,byrow=TRUE)
K=as.data.frame(K)
colnames(K)[1] <- "Portee"
colnames(K)[2] <- "Priorite"
colnames(K)[3] <- "Prime Reconstitution"
colnames(K)[4] <- "Clause Reconstitution"
for (i in 1:length(K[,1])){
  rownames(K)[i] <- paste("Réassureur", i)}
View(K)


################################

#Calcul du seuil à 5% pour différencier
#les sinistres extrêmes des sinistres attritionnels

q=quantile(UU$cout,0.95)  #32 699.17e


#############################################################################################
#SINISTRES ATTRITIONNELS
#############################################################################################



#1ERE METHODE : ESTIMATION DES LOIS EN FREQUENCE ET EN COUT DE MANIERE ANALYTIQUE
#############################################################################################


##ESTIMATION DE LA LOI POUR LES COUTS DES SINISTRES ATTRITIONNELS

hist(log(UU$cout),breaks=300)

#Simulation Sinistres avec Mclust
library(mclust)

t1=Sys.time()
UUU=Mclust(log(UU$cout))   #UU est la matrice obtenue avec la fonction SansCoutNul
t2=Sys.time()
t2-t1  #36 sec

summary(UUU)
plot(UUU)  #Ecrire 0 dans "Selection :" puis valider pour quitter la sélection
UUU$parameters

n=10^5
a=UUU$parameters$pro[1]
b=UUU$parameters$pro[2]
```

```
c=UUU$parameters$pro[3]
d=UUU$parameters$pro[4]
e=UUU$parameters$pro[5]
f=UUU$parameters$pro[6]
g=UUU$parameters$pro[7]
h=UUU$parameters$pro[8]
hh=UUU$parameters$pro[9]
j=runif(n,0,1)
MM=rep(0,n)

while (length(MM[MM<=0])>0){
  for (i in 1:n){
    if (MM[i]<=0){
      if (j[i]<=a){ MM[i]=rnorm(1,UUU$parameters$mean[1],
      UUU$parameters$variance$sigmasq[1]) }
      else if (j[i]<=a+b){ MM[i]=rnorm(1,UUU$parameters$mean[2],
      UUU$parameters$variance$sigmasq[2]) }
      else if (j[i]<=a+b+c){ MM[i]=rnorm(1,UUU$parameters$mean[3],
      UUU$parameters$variance$sigmasq[3]) }
      else if (j[i]<=a+b+c+d){ MM[i]=rnorm(1,UUU$parameters$mean[4],
      UUU$parameters$variance$sigmasq[4]) }
      else if (j[i]<=a+b+c+d+e){ MM[i]=rnorm(1,UUU$parameters$mean[5],
      UUU$parameters$variance$sigmasq[5]) }
      else if (j[i]<=a+b+c+d+e+f){ MM[i]=rnorm(1,UUU$parameters$mean[6],
      UUU$parameters$variance$sigmasq[6]) }
      else if (j[i]<=a+b+c+d+e+f+g){ MM[i]=rnorm(1,UUU$parameters$mean[7],
      UUU$parameters$variance$sigmasq[7]) }
      else if (j[i]<=a+b+c+d+e+f+g+h){ MM[i]=rnorm(1,UUU$parameters$mean[8],
      UUU$parameters$variance$sigmasq[8]) }
      else { MM[i]=rnorm(1,UUU$parameters$mean[9],
      UUU$parameters$variance$sigmasq[9]) }
    }
  }
}


hist(MM,breaks=1000,freq=FALSE,xlim=c(0,20))

################################


##ESTIMATION DE LA LOI POUR LA FREQUENCE DES SINISTRES ATTRITIONNELS


#Fréquence des sinistres par année

freq=NULL
for (i in 2000:2015){
  V=UU[UU[,2]==i & UU$cout<q,] #Sélectionne les sinistres de l'année i
```

```
  freq=rbind(freq,c(length(V[,5])))
}

freq=cbind(seq(2000,2015, by=1),freq)


plot(freq[,1], freq[,2],'l') #Evolution du nombre de sinistres attritionnels
#au fil des années
hist(freq[,2])


################################


#Simulation empirique :
#Simule un nombre de sinistres pour une année.
#INCONVENIENTS : Ne prend pas toutes les valeurs "possibles"

sim_freq=function(){
  u=runif(1,0,16)
  if(u<2){n=floor(runif(1,1000,1500))}
  else if (u<5){n=floor(runif(1,1500,2000))}
  else if (u<9){n=floor(runif(1,2000,2500))}
  else if (u<13){n=floor(runif(1,2500,3000))}
  else if (u<14){n=floor(runif(1,3000,3500))}
  else if (u<15){n=floor(runif(1,3500,4000))}
  else if (u<16){n=floor(runif(1,4000,4500))}
  return(n)
}

sim_freq()


############ Est-ce que la fréquence des sinistres suit une loi de poisson ?

fit=fitdistr(freq[,2],"poisson")
fit$estimate[1]

hist(freq[,2], freq=FALSE)
plot(density(freq[,2]))
tab=seq(0,6000, by=1)
lines(tab, dpois(tab, fit$estimate[1]), col='blue') #Pas bien ajusté du tout

qqplot(freq[,2], rpois(10000, fit$estimate[1])) #Pas ajusté
abline(0,1,col='red')

############ Est-ce que la fréquence des sinistres suit une loi négative binomiale ?

fit=fitdistr(freq[,2],"negative binomial")
```

```
hist(freq[,2], freq=FALSE)
plot(density(freq[,2]))
tab=seq(0,6000, by=1)
lines(tab, dnbinom(tab, size=fit$estimate[1],
 mu=fit$estimate[2]), col='red') ## C'est mieux

qqplot(freq[,2], rnbinom(10000, size=fit$estimate[1], mu=fit$estimate[2]))
abline(0,1,col='red')  #C'est pas mal


############ Est-ce que la fréquence des sinistres suit une loi normale ?

fit2=fitdistr(freq[,2], "normal")

hist(freq[,2], freq=FALSE)
plot(density(freq[,2]))
tab=seq(0,6000, by=1)
lines(tab, dnorm(tab, mean=fit2$estimate[1], sd=fit2$estimate[2]), col='red')
# C'est bien

qqplot(freq[,2], rnorm(10000, mean=fit2$estimate[1], sd=fit2$estimate[2]))
abline(0,1,col='red')  #C'est pas mal aussi

################################



#Comme les résultats les plus satisfaisants proviennent de la loi binomiale négative
#et de la loi normale, nous allons comparer les 2.

par(mfrow=c(2,2))

plot(density(freq[,2])) #Loi Binomiale Négative
lines(tab, dnbinom(tab, size=fit$estimate[1], mu=fit$estimate[2]), col='red')
qqplot(freq[,2], rnbinom(10000, size=fit$estimate[1], mu=fit$estimate[2]))
abline(0,1,col='red')

plot(density(freq[,2])) #Loi Normale
lines(tab, dnorm(tab, mean=fit2$estimate[1], sd=fit2$estimate[2]), col='red')
qqplot(freq[,2], rnorm(10000, mean=fit2$estimate[1], sd=fit2$estimate[2]))
abline(0,1,col='red')

par(mfrow=c(1,1))

#Nous allons donc choisir la loi Normale pour simuler la fréquence des sinistres
# attritionnels
meannorm=fit2$estimate[1]
sdnorm=fit2$estimate[2]

##############################################################################################
```

```
#2EME METHODE POUR LA SIMULATION DES SINISTRES ATTRITIONNELS :

#Calcul du ratio S/P

t1=Sys.time()

S=rep(0,16)
for (j in 1:length(UU$cout)){
  for (i in 1:16){
    if (UU$surv[j]==(1999+i) & UU$cout[j]<q){
      S[i]=S[i]+UU$cout[j]}
  }
}

t2=Sys.time()
t2-t1  #17 sec

P=sum(UU$cout[UU$cout>q])/16

rat=S/P
rat=rat*100
meanrrat=mean(rat) #39.8%

plot((S/P),type='l')

"Les ratios trouvés ne correspondent pas avec les ratios utilisés en
#responsabilité civile
#(60-70% selon les rapports annuels de l'ACAM et de la FFSA)"

################################################################################

#3EME METHODE POUR LA SIMULATION DES SINISTRES ATTRITIONNELS :

#Méthode Wilson - Hilferty

#Pour la charge cumulée par année S
muS=mean(S)
sigma2S=var(S)
skewnessS=skewness(S)

#Transformation de Wilson-Hilferty
h=1/3
Y=(S/muS)^h
muY=mean(Y)
sigma2Y=var(Y)

R=(Y-muY)/sqrt(sigma2Y)
mean(R)  #Egale à 0 environ
```

```
var(R)   #Egale à 1

b1=(skewnessS^2)/108
b2=skewnessS/6-6/skewnessS
b3=2/skewnessS

r=rnorm(1)
Stilde=b1*(r-b2)^3-b3
Ssimu=muS+sqrt(sigma2S)*Stilde

Ssimu
skewnessS #=0.37<0.5 : Approximation de WH validée

r=rnorm(10^7)
Stilde=b1*(r-b2)^3-b3
Ssimu=muS+sqrt(sigma2S)*Stilde

mean(Ssimu)
hist(Ssimu)
###############Test de Kolmogorv-Smirnov WH
WilsonHilferty=Ssimu
plot(density(WilsonHilferty),col="red")
q=quantile(UU$cout,0.95)
S=rep(0,16)
for (j in 1:length(UU$cout)){
  for (i in 1:16){
    if (UU$surv[j]==(1999+i) & UU$cout[j]<q){
      S[i]=S[i]+UU$cout[j]}
  }
}
S   #montant des sinistres cumulés par années

ks.test(WilsonHilferty,S, alternative = "two.sided")

########################################
#Comparaison avec une loi normale

hist(Ssimu,freq=FALSE,col="lightgreen")
abline(v = quantile(Ssimu,0.975), col="red", lwd=3, lty=2)
abline(v = quantile(Ssimu,0.025), col="blue", lwd=3, lty=2)

loinormale=rnorm(10^7,muS,sqrt(sigma2S))
hist(loinormale,freq=FALSE ,col="lightblue")
abline(v = quantile(loinormale,0.975), col="red", lwd=3, lty=2)
abline(v = quantile(loinormale,0.025), col="blue", lwd=3, lty=2)

LIWH=quantile(Ssimu, 0.975)-quantile(Ssimu, 0.025)
LILN=quantile(loinormale, 0.975)-quantile(loinormale, 0.025)
LIWH/LILN  #Comparaison de la longueur des intervalles qui comportent 95%
#des valeurs de chaque simulation
```

```
plot(density(Ssimu),col='red')
lines(density(loinormale))

quantile(Ssimu, 0.01) - quantile(loinormale, 0.01)  #Différence de presque
# 1 million d'euros
10^6/muS  #1 million
#d'euros représente une variation de 11% par rapport
# à la valeur moyenne donc ce n'est pas négligeable

quantile(Ssimu, 0.95) - quantile(loinormale, 0.95)  #Différence de 320 000 euros
(quantile(Ssimu, 0.95) - quantile(loinormale, 0.95))/quantile(loinormale, 0.95)
#Différence de 2.25% entre les 2 quantiles

#Nous pouvons remarquer que la simulation avec Wilson-Hilferty est assez semblable
#à une loi normale.
La différence se fait sur les cas extrêmes : Avec la loi normale nous aurons plus
#d'années avec un coût très faible,
la simulation avec Wilson-Hilferty peut se montrer alors plus réaliste que
#la simulation avec une loi normale.

########################################

WH=function(){
  r=rnorm(1)
  Stilde=b1*(r-b2)^3-b3
  Ssimu=muS+sqrt(sigma2S)*Stilde
  return(Ssimu)
}

WH()

#Cette fonction nous renvoie une valeur aléatoire
# pour la charge cumulée des sinistres attritionnels pour une année


#########################################################################################
#SINISTRES EXTREMES
#########################################################################################


#Sélection des sinistres extrêmes,
#ie ceux dont le cout final est sup ou égale à 32 699.17e (quantile à 5%)

ExtSini=UU[UU$cout>=q,]


################################
```

```
##ESTIMATION DE LA LOI POUR LA FREQUENCE DES SINISTRES EXTREMES


############### Nombre de sinistres extrêmes survenus par année en 2000 et 2015

freqExt=NULL
for (i in 2000:2015){
  V=ExtSini[ExtSini[,2]==i,]  #Sélectionne les sinistres extrêmes de l'année i
  freqExt=rbind(freqExt,c(length(V[,5])))
}

freqExt=cbind(seq(2000,2015, by=1),freqExt)
freqExt

sum(freqExt[,2])
sum(freqExt[,2])/length(UU$cout)  #On retrouve bien les 5% de sinistres extrêmes

plot(freqExt[,1], freqExt[,2],'l') #Evolution du nombre de sinistres extrêmes
# au fil des années
hist(freqExt[,2])

############## Proportion de sinistres extrêmes pour chaque année de survenance


Pourcent_Ext=NULL
for (i in 1:16){
  Pourcent_Ext=rbind(Pourcent_Ext, c( freqExt[i,2]/freq[i,2])) # vecteur contenant
la proportion de sinistres extrêmes pour chaque année
}

Pourcent_Ext=cbind(seq(2000,2015, by=1), Pourcent_Ext)

Resultats_Pourcent_Ext=c(min(Pourcent_Ext[,2]),max(Pourcent_Ext[,2]),
mean(Pourcent_Ext[,2]))
Resultats_Pourcent_Ext  # min=2.39%, max=6.47%, moy=5.05%

Pourcent_Ext
plot(Pourcent_Ext)
hist(Pourcent_Ext[,2],breaks=10)


############ Est-ce que la fréquence des sinistres extrêmes suit une loi de poisson ?

fit=fitdistr(freqExt[,2],"poisson")
fit$estimate[1]

hist(freqExt[,2], freq=FALSE)
plot(density(freqExt[,2]))
tab=seq(0,600, by=1)
```

```
lines(tab, dpois(tab, fit$estimate[1]), col='blue') #pas bien ajusté du tout

qqplot(freqExt[,2], rpois(10000, fit$estimate[1])) #pas ajusté
abline(0,1,col='red')


############ Est-ce que la fréquence des sinistres extrêmes suit une loi
négative binomiale ?

fit=fitdistr(freqExt[,2], "negative binomial")

hist(freqExt[,2], freq=FALSE)
plot(density(freqExt[,2]))
tab=seq(0,600, by=1)
lines(tab, dnbinom(tab, size=fit$estimate[1], mu=fit$estimate[2]), col='red')
## C'est mieux

qqplot(freqExt[,2], rnbinom(10000, size=fit$estimate[1], mu=fit$estimate[2]))
abline(0,1,col='red')  #Nous choisissons cette loi
pour la modélisation de la fréquence des sinistres extrêmes

sizebnegative=fit$estimate[1]
mubnegative=fit$estimate[2]

################################

#Validation de la simulation en fréquence

library(MASS)
A=freqExt[,2]
fit=fitdistr(A, "negative binomial")
r=mean(A)^2/(var(A)-mean(A))  #Méthode des moments
p=mean(A)/var(A)
r
p

#Test d'adéquation du khi deux
A=freqExt[,2];length(A)
nsim=length(A)
ysim=rnbinom(nsim,r,p)
table(A , ysim)->tableaucontingence
tableaucontingence
k=chisq.test(tableaucontingence)

k
"La pvalue est égale à 24% donc on accepte l'hypothèse que la fréquence des
sinistres extrêmes suit une loi binomiale
négative de paramètres r=9.53 et p=0.068"

################################
```

```
#ESTIMATION DE LA LOI POUR LE COUT DES SINISTRES EXTREMES

#TENTATIVE PREMIERE GEV AVEC LE PACKAGE {evd}

J=UU$cout #on utilise UU car c'est la dernière réévaluation de chaque sinistre,
sans les sinistres qui ne sont pas pris en charge
M=quantile(J,0.95)
M
J=J[J>M] #on a choisi le seuil (priorité) grâce au quantile à 95% :
distinction sinistres graves et pas graves
min(J) #Plus petite valeur des sinistres considérés comme graves
n=length(J)
nn=floor(n/50) #Nombre de blocs
xx=matrix(J,nrow=nn,ncol=50) #création de 50 blocs de taille 41
m=apply(xx,2,max) #on prend les max de chaque bloc
t=seq(1,nn,length.out=50) #création de l'abscisse
plot(t,m,type='l') #tracé des montants en fonction du "numéro" du sinistre
(classé de 1 à n)
abline(a=3*10^6,b=0,col='red')

fit=fgev(m,std.err = FALSE)
par(mfrow=c(2,2))
plot(fit) #les graphiques confirment l'hypothèse d'une GEV
#Probability plot : La fonction de répartition de la GEV converge
 vers la fonction de répartition de la loi de probabilité que suivent les données
#Quantile plot : La fonction quantile de la GEV est dans l'intervalle de confiance
de la fonction quantile observée sur les données
#Density plot : La densité  approchée coïncide à peu près avec celle observée
#Return Level Plot : Ce graphique n'a aucun sens car on ne raisonne pas en fonction
du temps, mais en fonction d'un "numéro" que l'on a attribué au sinistre
#Conclusion: Ce modèle nous parait réaliste!
fit #K>0 (shape) DA= Frechet, loi à queue lourde
locgev=fit$estimate[1]
scalegev=fit$estimate[2]
shapegev=fit$estimate[3] #K=0.2831407
par(mfrow=c(1,1))


################################


#TENTATIVE PREMIERE GPD AVEC LE PACKAGE {evd}

J=UU$cout
q=quantile(J,0.95) #Seuil à 95%
Jq=J[J>q]
nq=length(Jq)
hist(Jq,breaks=10^2,xlim=c(0,5*10^6)) #Confirme bien l'intuition d'une GPD
tq=seq(1,nq,length.out=nq)
```

```
plot(tq,Jq,type='l')
fit=fpot(J,threshold=q,npp=length(J),std.err = FALSE)
fit
fit$estimate[2]#K=0.3287491
par(mfrow=c(2,2))
plot(fit)  #Les résultats du qqplot sont incohérents par rapport aux résultats
des simulations obtenus
par(mfrow=c(1,1))

scalegpd=fit$estimate[1]
shapegpd=fit$estimate[2]

################################


#TENTATIVE SECONDE GPD AVEC LE PACKAGE {evmix}

fit1=fgpd(J,u=q,phiu=NULL,std.err=FALSE)  #GPD
fit1

n=41641
Sinis2=rgpd(n,sigmau=fit1$sigmau,xi=fit1$xi)

qqplot(J,Sinis2)
abline(0,1)

sigmaugpd=fit1$sigmau
xigpd=fit1$xi

################################

"Nous allons faire des tests de simulation de sinistres pour sélectionner
quelle simulation en coût est la plus fidèle à notre jeu de données."


#############################################################################################
#SIMULATION DES SINISTRES SUR UNE ANNEE
#############################################################################################

#1ERE METHODE : ON UTILISE MCLUST POUR SIMULER LES SINISTRES ATTRITIONNELS


simu1=function(){
  n=-1
  while (n<0){n=floor(0.95*rnorm(1,meannorm,sdnorm))} #Nombre de sinistres
  attritionnels dans une année
  nn=rnbinom(1,size=sizebnegative,mu=mubnegative) #nombre de sinistres
  extrêmes dans une année
  sinistre=rep(0,n+nn)
  for (i in 1:n){
```

```
    sinistre[i]=q+1 #q=32 699.17e représente le quantile à 95% des sinistres
pris en charge
    while(sinistre[i]>q){ #Simulation du coût des sinistres attritionnels
      ATTRI=floor(runif(1,1,length(MM))) #MM est un vecteur composé de 100 000
sinistres simulés avec les résultats de Mclust
      sinistre[i]=exp(MM[ATTRI]) #On choisit au hasard un sinistre dans MM, mais
qui est inférieur au seuil des sinistres extrêmes
    }
  }
  for (i in 1:nn){  #Renforce le caractère aléatoire des sinistres extrêmes,
ils ne représenteront pas forcément 5% des sinistres annuels
    sinistre[i+n]=0
    while(sinistre[i+n]<q){  #Choisir le type de simulation désirée en
rajoutant/retirant les dièses
      #sinistre[i+n]=rgev(1,locgev,scalegev,shapegev)  #Test GEV
      #sinistre[i+n]=rgpd(1,sigmau=sigmaugpd,xi=xigpd)  #Test GPD package {evmix}
      sinistre[i+n]=rgpd(1,sigmau=scalegpd,xi=shapegpd) #Test GPD package {evd}
    }
  }
  return (sinistre)
}


SIMU=simu1()
hist(log(SIMU),breaks=200)

SIMU2=NULL
for (i in 1:25){
  SIMU=simu1()
  SIMU2=c(SIMU2,SIMU)
}

par(mfrow=c(2,1))
hist(log(UU$cout),breaks=300)
hist(log(SIMU2),breaks=300) #La simulation est correcte
par(mfrow=c(1,1))

length(SIMU2[SIMU2>q])/length(SIMU2) #On obtient les 5% de sinistres extrêmes

##############################################################################



#2EME METHODE : ON UTILISE LA METHODE DE WILSON - HILFERTY POUR SIMULER LES SINISTRES
ATTRITIONNELS
#Donc on ne simule plus les sinistres attritionnels individuellement, on simule
seulement les sinistres extrêmes
#On aura juste à rajouter une valeur S simulée par la méthode WH à la charge de
l'assureur pour obtenir ce qu'il paie par année
```

```
simu2=function(){
  nn=rnbinom(1,size=sizebnegative,mu=mubnegative) #nombre de sinistres extrêmes
dans une année
  sinistre=rep(0,nn)
  for (i in 1:nn){
    while(sinistre[i]<q){  #Choisir le type de simulation désirée
en rajoutant/retirant les dièses
      #sinistre[i]=rgev(1,locgev,scalegev,shapegev)  #Test GEV
      #sinistre[i]=rgpd(1,sigmau=sigmaugpd,xi=xigpd)  #Test GPD package {evmix}
      sinistre[i]=rgpd(1,sigmau=scalegpd,xi=shapegpd) #Test GPD package {evd}
    }
  }
  return (sinistre)
}


SIMU=simu2()
hist(log(SIMU),breaks=10)


SIMU22=NULL
for (i in 1:50){
  SIMU=simu2()
  SIMU22=c(SIMU22,SIMU)
}

par(mfrow=c(2,1))
hist(log(UU$cout[UU$cout>q]),breaks=60)
hist(log(SIMU22),breaks=60) #La simulation est correcte
par(mfrow=c(1,1))


################################


#Recherche du seuil pour la GPD grace au package "evd"

evd::mrlplot(UU$cout[UU$cout>q],tlim=c(0,100000))
evd::mrlplot(UU$cout[UU$cout>q],tlim=c(30000,35000))

#Le seuil correspond bien au seuil q que nous avions trouvé(quantile à 5%)=32 699.17e
#Validation du seuil q pour la GPD


##############################################################################
#TARIFICATION
##############################################################################

#MONTANT TOTAL PRIS EN CHARGE PAR L'ASSUREUR ET LES REASSUREURS POUR UN VECTEUR
```

```
DE SINISTRESET UN CONTRAT XS DONNES

#Matrice K avec contrats XS
PaiementSimu=function(d,K){
#K[i,]=(C1,C2,Prime,CR) C1 : portée, C2 : priorité,
CR: Clause de reconstitution (0 à 50%),
  n=length(K[,1])
#Prime : Prime de reconstitution (définie, si égale à 0 alors Prime = 1%*portée )
  k=length(d)
  A=matrix(rep(0,k),ncol=k)
  T=matrix(rep(0,k),ncol=k)
  R=matrix(rep(0,n*k),nrow=k)
  O=rep(0,k)
  for (j in 1:n){
    if (K[j,3]==0){K[j,3]=0.01*K[j,1]}
  }
  for (i in (1:k)){
    for (j in (1:n)){
      R[i,j]=min((max(0,d[i]-K[j,2])),K[j,1])
      O[i]=O[i]+K[j,3]*K[j,4]*R[i,j]/K[j,1]
    }
    A[i]=d[i]-sum(R[i,])+O[i]
  }
  montant=cbind(t(A),O,R)
  montant=as.data.frame(montant)
  montantfinal=colSums(montant)
  montantfinal=as.data.frame(montantfinal)
  montantfinal[1,1]=montantfinal[1,1]  #+WH() Si on veut ajouter les sinistres
attritionnels
  colnames(montantfinal)[1] <- "Montant pris en charge"
  rownames(montantfinal)[1] <- "Assureur (coût Recons inclus)"
  rownames(montantfinal)[2] <- "Coût Reconstitutions"
  for (i in 3:length(montant)){
    rownames(montantfinal)[i] <- paste("Réassureur", i-2)
  }
  return(montantfinal)
}

#K=matrix(c(10000000,100,1000,0.50),ncol=4,byrow=TRUE)

#Si vous désirez tester une couverture
SIMU=simu2()

PaiementSimu(SIMU,K)

################################


#Fonction test qui va valider notre simulation de sinistres extrêmes
#L'objectif est que la tarification d'un contrat XS avec un grand nombre de simulations
```

```
#le contrat XS avec notre base de données Z

TarifXSTest=function(K){
  N=200
  for (i in 1:N){
    SIM=simu2()
    SIMU=c(SIMU,SIM)
  }
  Montant=PaiementSimu(SIMU,K)
  Paiement=(colSums(Montant)-Montant[1,1]-Montant[2,1])/N
  Paiement=rbind(Paiement,Montant)
  rownames(Paiement)[1] <- "TARIF XS 1 an"
  return(Paiement)
}

K=matrix(c(10000000,100,1000,0.50),ncol=4,byrow=TRUE)
"On prend une matrice K avec une priorité très faible et une portée très grande
pour qu'à peu près l'ensemble des sinistres soient totalement pris en charge.
Nous devons observer que la tarification d'un contrat XS à l'aide de notre
jeu de données Z revient au même que de tarifer le contrat XS avec une simulation
sur un grand nombre d'années."

t1=Sys.time()

TarifXSTest(K)

t2=Sys.time()
t2-t1  #26 sec

PaiementTotal(UU[UU$cout>q],K)
PaiementTotal(UU[UU$cout>q],K)[3,1]/16  #Tarif du contrat XS selon notre jeu de données

t1=Sys.time()
TarifXSTest(K)[1,1]/(PaiementTotal(UU[UU$cout>q],K)[3,1]/16)
t2=Sys.time()
t2-t1  #27 sec
#Le rapport est bien environ égal à 1, donc nous validons notre simulation de sinistres


################################


#Fonction qui va nous donner le tarif d'un contrat XS K à l'aide de notre jeu de données

TarifXS=function(K){
  d=UU$cout[UU$cout>q]
  n=length(K[,1]) #Impossible d'appeler la fonction PaiementSimu car des sinistres
attritionnels vont être générés
  k=length(d)     #La fonction PaiementTotal est longue à exécuter à cause
de la prise en compte des réévaluations
```

```
  A=matrix(rep(0,k),ncol=k)     #Donc nous adaptons le code de PaiementSimu pour
qu'il soit le plus adapté possible
  T=matrix(rep(0,k),ncol=k)
  R=matrix(rep(0,n*k),nrow=k)
  O=rep(0,k)
  for (j in 1:n){
    if (K[j,3]==0){K[j,3]=0.01*K[j,1]}
  }
  for (i in (1:k)){
    T[i]=d[i]
    for (j in (1:n)){
      R[i,j]=min((max(0,T[i]-K[j,2])),K[j,1])
      O[i]=O[i]+K[j,3]*K[j,4]*R[i,j]/K[j,1]
    }
    A[i]=T[i]-sum(R[i,])+O[i]
  }
  montant=cbind(t(A),O,R)
  montant=as.data.frame(montant)
  montantfinal=colSums(montant)
  montantfinal=as.data.frame(montantfinal)
  colnames(montantfinal)[1] <- "Montant pris en charge"
  rownames(montantfinal)[1] <- "Assureur (coût Recons inclus)"
  rownames(montantfinal)[2] <- "Coût Reconstitutions"
  for (i in 3:length(montant)){
    rownames(montantfinal)[i] <- paste("Réassureur", i-2)
  }
  Montant=montantfinal
  Paiement=(colSums(Montant)-Montant[1,1]-Montant[2,1])/16*1.05
#*1.05 = marge commerciale de 5%
  Paiement=rbind(Paiement,Montant)
  rownames(Paiement)[1] <- "TARIF XS 1 an"
  return(Paiement)
}

TarifXS(K)

################################


#Fonction qui calcule ce que l'assureur et les réassureurs auront à leur charge avec
un contrat XS donné lors d'une année simulée

MontantPris=function(K){
  SIMU=simu2()
  Montant=PaiementSimu(SIMU,K)
  Montant[1,1]=Montant[1,1]+WH()
  return(Montant)
}

MontantPris(K)
```

```
#########################################################################################
#OPTIMISATION OPTIMIZE
#########################################################################################

#Calcul du seuil à p% des sinistres attritionnels

t1=Sys.time()

p=0.95
Calculquant=0
for (i in 1:10^5){
  Calculquant[i]=WH()
}

funCalculQuant=function(){  #On simule 10^5 années pour les sinistres attritionnels
  Calculquant=0
  for (i in 1:10^5){
    Calculquant[i]=WH()
  }
  return(Calculquant)
}

t2=Sys.time()
t2-t1  #19 sec

hist(Calculquant)

quantAttri=quantile(Calculquant,p)
quantAttri  #Seuil à p% des simulations des sinistres attritionnels avec WH

"Ainsi, si l'assureur veut payer un montant N maximum par an avec une probabilité p,
N doit absolument être supérieur au quantile à p % du coût des sinistres attritionnels."

################################

#CREATION DE LA MATRICE DES SINISTRES EXTREMES ET DU VECTEUR DES SINISTRES
ATTRITIONNELS CUMULES

NN=500   #On simule 500 années (On peut augmenter le nombre pour plus de précision,
mais l'optimisation sera plus longue)

################################

#Simulation d'une matrice contenant une simulation de NN années des sinistres extrêmes

t1=Sys.time()

SIMU222=matrix(rep(0,NN*400),ncol=NN)
for (i in 1:NN){
```

```
  SIMU=simu2()
  nn=length(SIMU)
  for (j in 1:(400-nn)){
    SIMU[j+nn]=0
  }
  SIMU222[,i]=SIMU
}

t2=Sys.time()
t2-t1  #3 min

sum(SIMU222[400,])  #On somme la dernière ligne en vérifiant que c'est bien égale à 0
(aucune année à 400 sinistres ou plus)

mean(apply(SIMU222,2,sum))  #Moyenne des coûts des sinistres extrêmes par an
quantile(apply(SIMU222,2,sum),0.95)  #Quantile à 95% de la simulation des sinistres
extrêmes
hist(log(SIMU222))

################################


#Simulation d'une matrice contenant une simulation de NN années des sinistres
attritionnels

SIMUATTRI222=rep(0,NN)
for (i in 1:NN){
  SIMUATTRI222[i]=WH()
}

mean(SIMUATTRI222)  #Moyenne des coûts des sinistres attritionnels par an
quantile(SIMUATTRI222,0.95)  #Quantile à 95% de la simulation des sinistres
attritionnels

################################

#Optimisation de la couverture de réassurance et de son tarif :
#L'assureur est prêt à payer N euros avec une probabilité p par an

################################

#ON FIXE LA PRIORITE ET ON TROUVE LA PORTEE OPTIMALE POUR LE CONTRAT


p=0.95  #Probabilité avec laquelle l'assureur est prêt à payer N euros par an

t1=Sys.time()
quantAttri=quantile(funCalculQuant(),p)  #Quantile à p% des sinistres attritionnels
t2=Sys.time()
t2-t1  #19 sec
```

```
N=25*10^6  #Montant que l'assureur est prêt à supporter au total
n=N-quantAttri  #Montant que l'assureur est prêt à supporter pour les
sinistres extrêmes


################################

#Recherche de la priorité maximale avec laquelle les conditions sont respectées
(avec une portée infinie)

RecherchePriorite=function(priorite){
  pp=(1-p)*NN
  j=0
  portee=10^18  #Portée infinie
  K=matrix(c(portee,priorite,0.01*portee,0.5),ncol=4)
  for (k in 1:NN){
    E=PaiementSimu(SIMU222[,k],K)[1,1]+SIMUATTRI222[k]
    if (E>N){j=j+1}
  }
  if (pp<j){priorite=0}
  return(priorite)
}

t1=Sys.time()
ResuRecherchePriorite=optimize(RecherchePriorite,c(0,10^5),maximum = TRUE)
t2=Sys.time()
t2-t1  #1.1 min
ResuRecherchePriorite   #Priorité maximale que l'assureur peut choisir
#(augmenter l'intervalle si la priorité trouvée est égale à la borne supérieure)

"ATTENTION : SUR CERTAINS ORDINATEURS OPTIMIZE NE FONCTIONNE PAS TOUJOURS !
Pour que le résultat soit correct, il faut que $objective soit égal à $maximum,
sinon cela veut dire que optimize n'a pas fonctionné.
Méthode pour éviter le problème :
Ne pas prendre une borne supérieure trop grande sinon optimize bug :
prendre (0,10^5) pour le premier optimize.
Si la priorité obtenue est égale à 10^4, alors on change l'intervalle ==> (0,2*10^5),
et on relance optimize.
Si la priorité obtenue est égale à 2*10^4,
alors on change l'intervalle ==> (0,4*10^5), et on relance optimize, etc...
On augmente au fur et à mesure jusqu'à ce que la valeur trouvée ne soit pas la borne
supérieure de l'intervalle.

On garde tout de même ce programme car on peut anticiper dans quel intervalle se
situera la portée en fonction de la priorité choisie,
et donc gagner du temps."

################################
```

```
#Programme qui augmente l'intervalle automatiquement (si on n'anticipe pas l'intervalle
probable)

t1=Sys.time()

sup=10^4
ProgResuRechPrio=optimize(RecherchePriorite,c(0,sup),maximum = TRUE)
a=ProgResuRechPrio$maximum
b=ProgResuRechPrio$objective
while ((abs(b-a)>1) | (abs(b-sup)<1)) {
  sup=min(2*sup,sup+20000)
  ProgResuRechPrio=optimize(RecherchePriorite,c(0,sup),maximum = TRUE)
  a=ProgResuRechPrio$maximum
  b=ProgResuRechPrio$objective
}

ProgResuRechPrio  #Priorité maximale que l'assureur peut choisir

t2=Sys.time()
t2-t1  #9.6 min

################################

#Optimisation de la portée

priorite=33000  #Priorité choisie par l'assureur (inférieure à la priorité maximale)

OptiPortee=function(portee){
  pp=(1-p)*NN
  j=0
  K=matrix(c(portee,priorite,0.01*portee,0.5),ncol=4)
  for (k in 1:NN){
    E=PaiementSimu(SIMU222[,k],K)[1,1]+SIMUATTRI222[k]
    if (E>N){j=j+1}
  }
  if (pp<j){portee=10^15}
  return(portee)
}

t1=Sys.time()
resuPortee=optimize(OptiPortee,c(0,10^7),maximum = FALSE)  #Optimisation de la portée
t2=Sys.time()
t2-t1  #1.2 min
resuPortee  #On vérifie que les 2 résultats soient égaux et différents de la borne sup
de l'intervalle de recherche
#Sinon on augmente l'intervalle


################################
```

```
#Programme qui augmente l'intervalle automatiquement (si on n'anticipe pas
l'intervalle probable)

t1=Sys.time()

sup=10^5
ProgResuPortee=optimize(OptiPortee,c(0,sup),maximum = FALSE)
a=ProgResuPortee$minimum
b=ProgResuPortee$objective
while (abs(b-a)>1){
  sup=min(2*sup,sup+200000)
  ProgResuPortee=optimize(OptiPortee,c(0,sup),maximum = FALSE)
  a=ProgResuPortee$minimum
  b=ProgResuPortee$objective
}

ProgResuPortee  #Portée optimisée
t2=Sys.time()
t2-t1  #15.8 min

################################


#Présentation des résultats

porteeOpti=resuPortee$minimum  #Portée optimisée obtenue avec anticipation de
l'intervalle
porteeOpti=ProgResuPortee$minimum  #Portée optimisée obtenue
sans anticipation de l'intervalle


KresuPortee=matrix(c(porteeOpti,priorite,0.01*porteeOpti,0.5),ncol=4)

resuPorteeFinal=as.data.frame(c(N,p,priorite,porteeOpti,TarifXS(KresuPortee)[1,1]))
colnames(resuPorteeFinal)[1] = "Valeur"
rownames(resuPorteeFinal)[1] = "Montant que l'assureur est prêt à supporter"
rownames(resuPorteeFinal)[2] = "Probabilité avec laquelle il est prêt à payer N euros"
rownames(resuPorteeFinal)[3] = "Priorité du contrat XS correspondant"
rownames(resuPorteeFinal)[4] = "Portée du contrat XS correspondant"
rownames(resuPorteeFinal)[5] = "Tarif du contrat XS correspondant"
resuPorteeFinal

View(resuPorteeFinal)

################################


#Test de la couverture sur 500 années simulées (nouvelle simulation)
```

```
t1=Sys.time()

ll=1:500
for (j in 1:500){
  m=MontantPris(KresuPortee)
  ll[j]=m[1,1]
}

t2=Sys.time()
t2-t1  #1.7 min

N  #Montant que l'assureur accepte de prendre en charge avec une probabilité p
quantile(ll,p)  #Quantile à p% de la nouvelle simulation
quantile(ll,p-0.01)  #Quantile à (p-1)% de la nouvelle simulation
(Parfois la nouvelle simulation est très légèrement au dessus de N)
quantile(ll,p-0.02)  #C'est pourquoi on observe les quantiles inférieurs

################################

#ON FIXE LA PORTEE ET ON TROUVE LA PRIORITE OPTIMALE POUR LE CONTRAT

p=0.95

quantAttri=quantile(funCalculQuant(),p)  #Quantile à p% des sinistres attritionnels

N=25*10^6  #Montant que l'assureur est prêt à supporter au total
n=N-quantAttri  #Montant que l'assureur est prêt à supporter pour les sinistres
extrêmes

################################

#Recherche de la portée minimale avec laquelle les conditions sont respectées
(avec une priorité nulle)

RecherchePortee=function(portee){
  pp=(1-p)*NN
  j=0
  K=matrix(c(portee,0,0.01*portee,0.5),ncol=4)
  for (k in 1:NN){
    E=PaiementSimu(SIMU222[,k],K)[1,1]+SIMUATTRI222[k]
    if (E>N){j=j+1}
  }
  if (pp<j){portee=10^15}
  return(portee)
}

t1=Sys.time()
ResuRecherchePortee=optimize(RecherchePortee,c(10^3,10^8),maximum = FALSE)
t2=Sys.time()
```

```
t2-t1  #1.3 min
ResuRecherchePortee   #Portée minimale que l'assureur peut choisir

################################


#Trouver la priorité optimale pour une portée donnée

ResuRecherchePortee$minimum   #La portée doit absolument être supérieure à ce montant
portee=ResuRecherchePortee$minimum+10^6  #La portée fixée par l'assureur

OptiPriorite=function(priorite){
  pp=(1-p)*NN
  j=0
  K=matrix(c(portee,priorite,0.01*portee,0.5),ncol=4)
  for (k in 1:NN){
    E=PaiementSimu(SIMU222[,k],K)[1,1]+SIMUATTRI222[k]
    if (E>N){j=j+1}
  }
  if (pp<j){priorite=0}
  return(priorite)
}

t1=Sys.time()
resuPriorite=optimize(OptiPriorite,c(0,10^5),maximum = TRUE)  #Optimisation de la
priorité
t2=Sys.time()
t2-t1  #1.1 min
resuPriorite  #On vérifie que les 2 résultats soient égaux et différents de la borne
sup de l'intervalle de recherche
#Sinon on augmente l'intervalle

################################


#Programme qui augmente l'intervalle automatiquement (si on n'anticipe pas l'intervalle
probable)

t1=Sys.time()

sup=10^4
ProgResuPrio=optimize(OptiPriorite,c(0,sup),maximum = TRUE)
a=ProgResuPrio$maximum
b=ProgResuPrio$objective
while ((abs(b-a)<1) & (abs(b-sup)<1)) {
  sup=min(2*sup,sup+20000)
  ProgResuPrio=optimize(OptiPriorite,c(0,sup),maximum = TRUE)
  a=ProgResuPrio$maximum
  b=ProgResuPrio$objective
}
```

```
ProgResuPrio  #Priorité optimisée
t2=Sys.time()
t2-t1  #6.6 min

################################


#Présentation des résultats

prioriteOpti=resuPriorite$maximum  #Prioritée optimisée obtenue avec anticipation
de l'intervalle
prioriteOpti=ProgResuPortee$maximum  #Prioritée optimisée obtenue sans anticipation
de l'intervalle


KresuPriorite=matrix(c(portee,prioriteOpti,0.01*portee,0.5),ncol=4)

resuPrioriteFinal=as.data.frame(c(N,p,prioriteOpti,portee,TarifXS(KresuPriorite)[1,1]))
colnames(resuPrioriteFinal)[1] = "Valeur"
rownames(resuPrioriteFinal)[1] = "Montant que l'assureur est prêt à supporter"
rownames(resuPrioriteFinal)[2] = "Probabilité avec laquelle il est prêt à payer N euros"
rownames(resuPrioriteFinal)[3] = "Priorité du contrat XS correspondant"
rownames(resuPrioriteFinal)[4] = "Portée du contrat XS correspondant"
rownames(resuPrioriteFinal)[5] = "Tarif du contrat XS correspondant"
resuPrioriteFinal

View(resuPrioriteFinal)

################################


#Test de la couverture sur 500 années simulées (nouvelle simulation)

t1=Sys.time()

ll=1:500
for (j in 1:500){
  m=MontantPris(KresuPriorite)
  ll[j]=m[1,1]
}

t2=Sys.time()
t2-t1  #1.7 min

N  #Montant que l'assureur accepte de prendre en charge avec une probabilité p
quantile(ll,p)  #Quantile à p% de la nouvelle simulation
quantile(ll,p-0.01)  #Quantile à (p-1)% de la nouvelle simulation
(Parfois la nouvelle simulation est très légèrement au dessus de N)
```

```
quantile(ll,p-0.02)  #C'est pourquoi on observe les quantiles inférieurs

#########################################################################################
#OPTIMISATION DICHOTOMIE
#########################################################################################

#Calcul du seuil à p % des sinistres attritionnels

t1=Sys.time()

p=0.95
Calculquant=0
for (i in 1:10^5){
  Calculquant[i]=WH()
}

funCalculQuant=function(){  #On simule 10^5 années pour les sinistres attritionnels
  Calculquant=0
  for (i in 1:10^5){
    Calculquant[i]=WH()
  }
  return(Calculquant)
}

t2=Sys.time()
t2-t1  #19 sec

hist(Calculquant)

quantAttri=quantile(Calculquant,p)
quantAttri  #Seuil à p % des simulations des sinistres attritionnels avec WH

"Ainsi, si l'assureur veut payer un montant N maximum par an avec une probabilité p,
N doit absolument être supérieur au quantile à p % du coût des sinistres attritionnels."

################################


#CREATION DE LA MATRICE DES SINISTRES EXTREMES ET DU VECTEUR DES SINISTRES
ATTRITIONNELS CUMULES

NN=500   #On simule 500 années (On peut augmenter le nombre pour plus de précision,
mais l'optimisation sera plus longue)

################################


#Simulation d'une matrice contenant une simulation de NN années des sinistres extrêmes

t1=Sys.time()
```

```
SIMU222=matrix(rep(0,NN*400),ncol=NN)
for (i in 1:NN){
  SIMU=simu2()
  nn=length(SIMU)
  for (j in 1:(400-nn)){
    SIMU[j+nn]=0
  }
  SIMU222[,i]=SIMU
}

t2=Sys.time()
t2-t1  #3 min

sum(SIMU222[400,])  #On somme la dernière ligne en vérifiant que c'est bien égale à 0
(aucune année à 400 sinistres ou plus)

mean(apply(SIMU222,2,sum))  #Moyenne des coûts des sinistres extrêmes par an
quantile(apply(SIMU222,2,sum),0.95)  #Quantile à 95% de la simulation des sinistres
extrêmes

################################


#Simulation d'une matrice contenant une simulation de NN années des
sinistres attritionnels

SIMUATTRI222=rep(0,NN)
for (i in 1:NN){
  SIMUATTRI222[i]=WH()
}

mean(SIMUATTRI222)  #Moyenne des coûts des sinistres attritionnels par an
quantile(SIMUATTRI222,0.95)  #Quantile à 95% de la simulation des sinistres
attritionnels
hist(log(SIMU222))

################################

#Optimisation de la couverture de réassurance et de son tarif :
#L'assureur est prêt à payer N euros avec une probabilité p par an

################################

#ON FIXE LA PRIORITE ET ON TROUVE LA PORTEE OPTIMALE POUR LE CONTRAT


p=0.95  #Probabilité avec laquelle l'assureur est prêt à payer N euros par an

t1=Sys.time()
```

```
quantAttri=quantile(funCalculQuant(),p)  #Quantile à p% des sinistres attritionnels
t2=Sys.time()
t2-t1  #19 sec

N=25*10^6  #Montant que l'assureur est prêt à supporter au total
n=N-quantAttri  #Montant que l'assureur est prêt à supporter pour les sinistres
extrêmes


################################


#Recherche de la priorité maximale avec laquelle les conditions sont respectées
(avec une portée infinie)

DichoRechPrio=function(){
  b=max(SIMU222)
  a=0
  portee=10^15
  while((b-a)>1){
    cc=(a+b)/2
    K=matrix(c(portee,cc,0.01*portee,0.5),ncol=4)
    ll=NULL
    for (j in 1:NN){
      m=PaiementSimu(SIMU222[,j],K)+SIMUATTRI222[j]
      ll[j]=m[1,1]
    }
    test=quantile(ll,0.95)
    if (test>N){b=cc}  #Conditions non respectées ==> priorité trop élevée
    else {a=cc}
  }
  return(cc)
}


t1=Sys.time()

PrioriteMaxDicho=DichoRechPrio()
PrioriteMaxDicho  #Priorité maximale que l'assureur peut choisir

t2=Sys.time()
t2-t1  #1.3 min


################################


#Optimisation de la portée

priorite=33000  #La priorité choisie par l'assureur (inférieure à la priorité maximale)
```

```
DichoOptiPortee=function(priorite){
  b=max(SIMU222)
  a=0
  while((b-a)>1){
    cc=(a+b)/2
    K=matrix(c(cc,priorite,0.01*cc,0.5),ncol=4)
    ll=NULL
    for (j in 1:NN){
      m=PaiementSimu(SIMU222[,j],K)+SIMUATTRI222[j]
      ll[j]=m[1,1]
    }
    test=quantile(ll,0.95)
    if (test>N){a=cc}  #Conditions non respectées ==> portée trop faible
    else {b=cc}
  }
  return(cc)
}


t1=Sys.time()

DichoPorteeOpti=DichoOptiPortee(priorite)
DichoPorteeOpti  #Portée optimisée

t2=Sys.time()
t2-t1  #1.3 min

################################


#Présentation des résultats

KresuPorteeDicho=matrix(c(DichoPorteeOpti,priorite,0.01*DichoPorteeOpti,0.5),ncol=4)

resuPorteeFinalDicho=as.data.frame(c(N,p,priorite,
DichoPorteeOpti,TarifXS(KresuPorteeDicho)[1,1]))
colnames(resuPorteeFinalDicho)[1] = "Valeur"
rownames(resuPorteeFinalDicho)[1] = "Montant que l'assureur est prêt à supporter"
rownames(resuPorteeFinalDicho)[2] = "Probabilité avec laquelle il est prêt à payer
N euros"
rownames(resuPorteeFinalDicho)[3] = "Priorité du contrat XS correspondant"
rownames(resuPorteeFinalDicho)[4] = "Portée du contrat XS correspondant"
rownames(resuPorteeFinalDicho)[5] = "Tarif du contrat XS correspondant"
resuPorteeFinalDicho

View(resuPorteeFinalDicho)

################################
```

```
#Test de la couverture sur 500 années simulées (nouvelle simulation)

t1=Sys.time()

ll=1:500
for (j in 1:500){
  m=MontantPris(KresuPorteeDicho)
  ll[j]=m[1,1]
}

t2=Sys.time()
t2-t1  #1.7 min

N  #Montant que l'assureur accepte de prendre en charge avec une probabilité p
quantile(ll,p)  #Quantile à p% de la nouvelle simulation
quantile(ll,p-0.01)  #Quantile à (p-1)% de la nouvelle simulation
(Parfois la nouvelle simulation est très légèrement au dessus de N)
quantile(ll,p-0.02)  #C'est pourquoi on observe les quantiles inférieurs

################################

#ON FIXE LA PORTEE ET ON TROUVE LA PRIORITE OPTIMALE POUR LE CONTRAT

p=0.95

quantAttri=quantile(funCalculQuant(),p)  #Quantile à p% des sinistres attritionnels

N=25*10^6  #Montant que l'assureur est prêt à supporter au total
n=N-quantAttri  #Montant que l'assureur est prêt
à supporter pour les sinistres extrêmes

################################


#Recherche de la portée minimale avec laquelle les conditions sont respectées
(avec une priorité nulle)

DichoRechPortee=function(){
  b=max(SIMU222)
  a=0
  priorite=0
  while((b-a)>1){
    cc=(a+b)/2
    K=matrix(c(cc,priorite,0.01*cc,0.5),ncol=4)
    ll=NULL
    for (j in 1:NN){
      m=PaiementSimu(SIMU222[,j],K)+SIMUATTRI222[j]
      ll[j]=m[1,1]
    }
```

```
    test=quantile(ll,0.95)
    if (test>N){a=cc}  #Conditions non respectées ==> portée trop faible
    else {b=cc}
  }
  return(cc)
}


t1=Sys.time()

PorteeMiniDicho=DichoRechPortee()
PorteeMiniDicho  #Portée minimale que l'assureur peut choisir

t2=Sys.time()
t2-t1  #1.3 min

################################

#Trouver la priorité optimale pour une portée donnée

portee=PorteeMiniDicho+10^6  #La portée fixée par l'assureur
(supérieure à la portée minimale)

DichoOptiPriorite=function(portee){
  b=max(SIMU222)
  a=0
  while((b-a)>1){
    cc=(a+b)/2
    K=matrix(c(portee,cc,0.01*portee,0.5),ncol=4)
    ll=NULL
    for (j in 1:NN){
      m=PaiementSimu(SIMU222[,j],K)+SIMUATTRI222[j]
      ll[j]=m[1,1]
    }
    test=quantile(ll,0.95)
    if (test>N){b=cc}  #Conditions non respectées ==> priorité trop élevée
    else {a=cc}
  }
  return(cc)
}


t1=Sys.time()

DichoPrioriteOpti=DichoOptiPriorite(portee)
DichoPrioriteOpti  #Priorité optimisée

t2=Sys.time()
t2-t1  #1.2 min
```

```
################################


#Présentation des résultats

KresuPrioriteDicho=matrix(c(portee,DichoPrioriteOpti,0.01*portee,0.5),ncol=4)

resuPrioriteFinalDicho=as.data.frame(c(N,p,
DichoPrioriteOpti,portee,TarifXS(KresuPrioriteDicho)[1,1]))
colnames(resuPrioriteFinalDicho)[1] = "Valeur"
rownames(resuPrioriteFinalDicho)[1] = "Montant que l'assureur est prêt à supporter"
rownames(resuPrioriteFinalDicho)[2] = "Probabilité avec laquelle
 il est prêt à payer N euros"
rownames(resuPrioriteFinalDicho)[3] = "Priorité du contrat XS correspondant"
rownames(resuPrioriteFinalDicho)[4] = "Portée du contrat XS correspondant"
rownames(resuPrioriteFinalDicho)[5] = "Tarif du contrat XS correspondant"
resuPrioriteFinalDicho

View(resuPrioriteFinalDicho)

################################

#Test de la couverture sur 500 années simulées (nouvelle simulation)

t1=Sys.time()

ll=1:500
for (j in 1:500){
  m=MontantPris(KresuPrioriteDicho)
  ll[j]=m[1,1]
}

t2=Sys.time()
t2-t1  #1.7 min

N  #Montant que l'assureur accepte de prendre en charge avec une probabilité p
quantile(ll,p)  #Quantile à p% de la nouvelle simulation
quantile(ll,p-0.01)  #Quantile à (p-1)% de la nouvelle simulation
(Parfois la nouvelle simulation est très légèrement au dessus de N)
quantile(ll,p-0.02)  #C'est pourquoi on observe les quantiles inférieurs


################################################################################
APPLICATION NUMERIQUE
################################################################################

#CREATION DE LA MATRICE DES SINISTRES EXTREMES ET DU VECTEUR DES SINISTRES
ATTRITIONNELS CUMULES

NN=500   #On simule 500 années (On peut augmenter le nombre pour plus de précision,
```

```
mais l'optimisation sera plus longue)

##############################################################################################

#Simulation d'une matrice contenant une simulation de NN années des sinistres extrêmes

t1=Sys.time()

SIMU222=matrix(rep(0,NN*400),ncol=NN)
for (i in 1:NN){
  SIMU=simu2()
  nn=length(SIMU)
  for (j in 1:(400-nn)){
    SIMU[j+nn]=0
  }
  SIMU222[,i]=SIMU
}

t2=Sys.time()
t2-t1  #3 min

sum(SIMU222[400,])  #On somme la dernière ligne en vérifiant que c'est bien égale à 0
(aucune année à 400 sinistres ou plus)

mean(apply(SIMU222,2,sum))  #Moyenne des coûts des sinistres extrêmes par an
quantile(apply(SIMU222,2,sum),0.95)  #Quantile à 95% de la simulation des sinistres
extrêmes
hist(log(SIMU222))

##############################################################################################

#Simulation d'une matrice contenant une simulation de NN années des sinistres
attritionnels

SIMUATTRI222=rep(0,NN)
for (i in 1:NN){
  SIMUATTRI222[i]=WH()
}

mean(SIMUATTRI222)  #Moyenne des coûts des sinistres attritionnels par an
quantile(SIMUATTRI222,0.95)  #Quantile à 95% de la simulation des sinistres
attritionnels
hist(log(SIMU222))

##############################################################################################
##############################################################################################


# Contrat non-optimisé mais respectant les conditions
K=matrix(c(1500000,50000,15000,0.5), nrow=1)
```

```
TarifXS(K)
ll=NULL
for (j in 1:NN){
  m2=PaiementSimu(SIMU222[,j],K)+SIMUATTRI222[j]
  ll[j]=m2[1,1]}

quantile(ll,p)
mean(ll)



################################################################################
################################################################################

#Optimisation de la couverture de réassurance et de son tarif :
#L'assureur est prêt à payer N euros avec une probabilité p par an

################################################################################

#ON FIXE LA PRIORITE ET ON TROUVE LA PORTEE OPTIMALE POUR LE CONTRAT

p=0.95  #Probabilité avec laquelle l'assureur est prêt à payer N euros par an

t1=Sys.time()
quantAttri=quantile(funCalculQuant(),p)  #Quantile à p% des sinistres attritionnels
t2=Sys.time()
t2-t1  #19 sec

N=35*10^6  #Montant que l'assureur est prêt à supporter au total
n=N-quantAttri  #Montant que l'assureur est prêt à supporter pour les
sinistres extrêmes



#Recherche de la priorité maximale avec laquelle les conditions sont respectées
(avec une portée infinie)

DichoRechPrio=function(){
  b=max(SIMU222)
  a=0
  portee=10^15
  while((b-a)>1){
    cc=(a+b)/2
    K=matrix(c(portee,cc,0.01*portee,0.5),ncol=4)
    ll=NULL
    for (j in 1:NN){
      m=PaiementSimu(SIMU222[,j],K)+SIMUATTRI222[j]
      ll[j]=m[1,1]
    }
    test=quantile(ll,0.95)
```

```
    if (test>N){b=cc}  #Conditions non respectées ==> priorité trop élevée
    else {a=cc}
  }
  return(cc)
}


t1=Sys.time()

PrioriteMaxDicho=DichoRechPrio()
PrioriteMaxDicho  #Priorité maximale que l'assureur peut choisir

t2=Sys.time()
t2-t1  #1.3 min


# Test avec différentes priorités
t1=Sys.time()
DichoPorteeOpti=NULL
#vecteur des priorités simulées
priorite=seq(0,PrioriteMaxDicho, by=PrioriteMaxDicho/10)
l=length(priorite)
for (i in 1:l){
DichoPorteeOpti=c(DichoPorteeOpti, DichoOptiPortee(priorite[i]))}
t2=Sys.time()
t2-t1 #40 min

################################################################################

#Présentation des résultats

MatrixPrioVariante=matrix(c(DichoPorteeOpti,priorite,0.01*DichoPorteeOpti,rep(0.5,11))
,ncol=4, nrow=11)

Tarif=NULL
for (i in 1:nrow(MatrixPrioVariante)){
resuPorteeDicho=as.data.frame(c(N,p,priorite[i],
DichoPorteeOpti[i],TarifXS(matrix(MatrixPrioVariante[i,],nrow=1))[1,1]))
colnames(resuPorteeDicho)[1] = "Valeur"
rownames(resuPorteeDicho)[1] = "Montant que l'assureur est prêt à supporter"
rownames(resuPorteeDicho)[2] = "Probabilité avec laquelle il est prêt à payer N euros"
rownames(resuPorteeDicho)[3] = "Priorité du contrat XS correspondant"
rownames(resuPorteeDicho)[4] = "Portée du contrat XS correspondant"
rownames(resuPorteeDicho)[5] = "Tarif du contrat XS correspondant"
Tarif[i]=resuPorteeDicho[5,]}


# Evolution du tarif en fonction de la priorité
plot(priorite,Tarif,'p')
```

```
#L'assureur choisi la priorité qui minimise le tarif
position=which.min(Tarif) # position du tarif minimal


# Contrat optimal mis en place
Kopti=matrix(c(DichoPorteeOpti[position] ,priorite[position],
0.01*DichoPorteeOpti[position] ,0.5),ncol=4)

# Calcul de l'esperance de prise en charge et du maximum de prise en charge
ll2=NULL
for (j in 1:NN){
  m2=PaiementSimu(SIMU222[,j],Kopti)+SIMUATTRI222[j]
  ll2[j]=m2[1,1]}

mean(ll2)
quantile(ll2,0.95)

##############################################################################


#ON FIXE LA PORTEE ET ON TROUVE LA PRIORITE OPTIMALE POUR LE CONTRAT

#Recherche de la portée minimale avec laquelle les conditions sont respectées
(avec une priorité nulle)

DichoRechPortee=function(){
  b=max(SIMU222)
  a=0
  priorite=0
  while((b-a)>1){
    cc=(a+b)/2
    K=matrix(c(cc,priorite,0.01*cc,0.5),ncol=4)
    ll=NULL
    for (j in 1:NN){
      m=PaiementSimu(SIMU222[,j],K)+SIMUATTRI222[j]
      ll[j]=m[1,1]
    }
    test=quantile(ll,0.95)
    if (test>N){a=cc}  #Conditions non respectées ==> portée trop faible
    else {b=cc}
  }
  return(cc)
}


t1=Sys.time()

PorteeMiniDicho=DichoRechPortee()
PorteeMiniDicho  #Portée minimale que l'assureur peut choisir
```

```
t2=Sys.time()
t2-t1  #1.3 min


t1=Sys.time()
DichoPrioriteOpti=NULL
portee=seq(PorteeMiniDicho, PorteeMiniDicho+25*10^6, by=25*10^5)
l2=length(portee)
for (i in 1:l2){
  DichoPrioriteOpti=c(DichoPrioriteOpti, DichoOptiPriorite(portee[i]))}
t2=Sys.time()
t2-t1 # 42 min


#Présentation des résultats
MatrixPortVariante=matrix(c(portee,DichoPrioriteOpti,0.01*portee,rep(0.5,11)),
ncol=4, nrow=11)

Tarif2=NULL
for (i in 1:nrow(MatrixPortVariante)){
  resuPrioriteDicho=as.data.frame(c(N,p,
  DichoPrioriteOpti[i],portee[i],TarifXS(matrix(MatrixPortVariante[i,],nrow=1))[1,1]))
  colnames(resuPrioriteDicho)[1] = "Valeur"
  rownames(resuPrioriteDicho)[1] = "Montant que l'assureur est prêt à supporter"
  rownames(resuPrioriteDicho)[2] = "Probabilité avec laquelle il est prêt à payer
N euros"
  rownames(resuPrioriteDicho)[3] = "Priorité du contrat XS correspondant"
  rownames(resuPrioriteDicho)[4] = "Portée du contrat XS correspondant"
  rownames(resuPrioriteDicho)[5] = "Tarif du contrat XS correspondant"
  Tarif2[i]=resuPrioriteDicho[5,]}


# Evolution du tarif en fonction de la priorité
plot(portee,Tarif2,'p')

#L'assureur choisi la portée qui minimise le tarif
position2=which.min(Tarif2) # position du tarif minimal


# Contrat optimal mis en place
Kopti2=matrix(c(portee[position2], DichoPrioriteOpti[position2], 0.01*portee[position2],
0.5),nrow=1)

# Calcule de l'espérance de prise en charge et du quantile à 95%
ll3=NULL
for (j in 1:NN){
  m2=PaiementSimu(SIMU222[,j],Kopti2)+SIMUATTRI222[j]
  ll3[j]=m2[1,1]}

quantile(ll3,p)
```

```
mean(ll3)
```