RAPPORT DE STAGE MISSION

Développement de l'outil Hermès pour la tarification Groupe

Stagiaire :
Duc Hien VU

Encadrant :
Dimitri MINASSIAN

2 mai 2015

Remerciements

J'adresse mes sincères remerciements à l'équipe Actuariat Réassurance d'AXA Global P&C.

Je remercie plus particulièrement Dimitri MINASSIAN de m'avoir guidé dans mon travail et m'avoir aidé à trouver des solutions pour avancer. Sa disponibilité, son soutien, son encouragement et sa patience m'ont été précieux afin de bien mener mon travail.

Table des matières

Introduction

AXA Global P&C est une filiale dynamique du groupe AXA, créée en fin 2010 pour but de piloter les sous-branches d'activité IARD (property) et responsabilité civile (casualty). Elle est aussi la plateforme de placement des traités de réassurance pour l'ensemble du groupe AXA. En effet, la réassurance P&C au sein du groupe représente plus de 1mds€ de primes. La tarification des traités de réassurance est un enjeu majeur pour le groupe afin de maîtriser ses cessions aux acteurs externes de la réassurance tels que Swiss Re, Munich Re ou SCOR. Elle permet d'optimiser les placements en réassurance et donc de protéger l'entité selon son propore appétit au risque.

Le département Actuariat Réassurance d'AXA Global P&C a pour principale mission le pricing des structures de réassurance, la modélisation des risques atypiques (catastrophes naturelles et techniques) et le support aux équipes commerciales pour le placement sur le marché international de la réassurance.

Un outil interne qui s'appelle Hermès a été développé au sein du département afin de tarifer et évaluer la performance d'une ou plusieurs structures de réassurance P&C. Cet outil est en fait un grand programme R utilisant comme plateforme un logiciel (une surcouche R) développé en interne chez AXA. Il utilise la méthode de Monte Carlo pour simuler un grand nombre de scénarios de sinistralité. Hermès comprend trois modules principaux : Le générateur des pertes, le pricer des structures de réassurance (les traités) et le rapport.

Je participe à un projet de l'équipe Actuariat, en collaboration avec l'équipe IT, pour améliorer Hermès et l'implémenter à la plateforme Web. Les principales missions accomplies pendant ce stage sont :

- S'approprier l'expertise AXA Global P&C en terme de tarification
- Comprendre l'outil de tarification actuel et l'améliorer : identifier les parties de codes R pouvant être optimisées par le langage C/C++ et implémenter ces alternatives
- Etablir une hiérachie des différents modules de l'outil existants et les rendre indépendants afin de permettre l'appel de ces derniers via la plateforme Web d'interface utilisateur.

Ce rapport a pour vocation de présenter notre projet et de résumer mon travail sur l'outil Hermès. Je commencerai par présenter les notions importantes de la réassurance. Je détaillerai ensuite les modules principaux de Hermès. Enfin je parlerai des modifications apportées à ce programme afin d'accélérer les phases de code et de préparer pour la plateforme Web.

1 La réassurance

Avant de parler de l'outil Hermès, il s'avère nécessaire de présenter brièvement les concepts fondamentaux de la réassurance et les types de réassurance.

1.1 Définition

La réassurance est l'assurance achetée par un assureur pour limiter la perte totale qu'il peut subir en cas des événements extrêmes tels que les catastrophes naturelles qui affectent un grand nombre d'assurés.

Le contrat de réassurance est un contrat par lequel le réassureur s'engage envers l'assureur contractant, encore appelé cédante, à couvrir tout ou partie des montants de sinistres du portefeuille de l'assureur, moyennant le paiement d'une prime de réassurance. Les sommes dues à la cédante par le réassureur au titre du contrat sont appelées récupérations.

La réassurance permet une compagnie d'assurance à avoir des résultats plus stables et plus prédictifs en limitant les pertes trop élevées (à la fois en termes de sévérité et en termes de fréquence), mais aussi à réduire le montant du capital requis pour la solvabilité (SCR).

1.2 Transfert de risques et transfert de pertes

Nous distinguons généralement deux grandes catégories de réassurance : la réassurance proportionnelle et la réassurance non-proportionnelle.

- ***la réassurance proportionnelle (le transfert de risque) :*** La perte pour un risque spécifique est partagée entre l'assureur et le réassureur proportionnellement à la part de primes partagée entre eux.
  Exemples : traités ***Quota Share (QS)*** ou traités ***Surplus***.

- ***la réassurance non-proportionnelle (le transfert de pertes) :*** recouvre l'ensemble des formes de réassurance qui ne correspondent pas au principe de la réassurance proportionnelle. Dans ce type de réassurance, l'assureur et le réassureur définissent la façon dont le réassureur interviendra en cas de pertes et ainsi le montant de primes de réassurance associées. La réassurance non-proportionnelle est souvent utilisée pour réduire les expositions extrêmes et catastrophiques.
  Exemples : traités ***Excess of Loss (XL)*** ou traités ***Stop-Loss***.

  En pratique, les activités de réassurance peuvent mélanger les deux catégories de réassurance.
  Exemple : une structure ***XL on QS retention***.

1.3 Traités QS

Nous présentons dans cette partie la forme de réassurance proportionnelle la plus utilisée : le traité en Quote Part (QP) ou en anglais Quota Share (QS). La réassurance QS est le partage de tout business par un ratio fixe (le taux de cession). Notons ***GNPI (Gross Net Premium Income)*** le montant des primes

collectées par la cédante sur le portefeuille réassuré. Le réassureur paiera un certain pourcentage du montant cumulé des sinistres touchant le portefeuille réassuré en échange du même pourcentage sur le GNPI. Un traité QS(70%) signifie 70% du GNPI, et ainsi 70% des sinistres de ce portefeuille, sont cédés au réassureur.

Parlant des traités QS, on spécifie également la ***commision de réassurance***. La cédante a un certain montant de charges liées au portefeuille réassuré (pour collecter les primes et pour la gestion des sinistres). Le réassureur doit donc payer à la cédante une commission afin de partager ces coûts d'acquisition et de gestion. Le montant de cette commission est déterminé grâce au "Loss ratio cible".

D'un point de vue de la cédante, les traités QS ont comme avantages un partage des frais avec le réassureur et une couverture efficace contre les risques de haute fréquence et les risques inconnus lors du lancement d'un nouveau produit d'assurance. Cepedant, les traités QS obligent la cédante à céder une partie importante des primes, même dans le cas des pertes attritionnelles qu'elle est capable de subir.

1.4 Traités XL

Le traité en excédent de sinistre XS (ou XL en anglais) est le plus usité des traités non-proportionnels et aussi le plus usité sur le marché de la réassurance. La cédante et le réassureur définissent une prime de réassurance, exprimée généralement en pourcentage du GNPI.

1.4.1 Traité XL simple avec portée et priorité

Un traité en excédent de sinistres simple est souvent noté ***m XL a*** où m est la ***portée*** et a la ***priorité***. m XL a désigne l'intervalle [a, a + m], appelé tranche de réassurance.

Soit X le montant de sinistre du traité XL par sinistre. La priorité a est le montant de sinistres au-dessus duquel le réassureur intervient, et il va prendre en charge l'excès des sinistres, à condition que cette somme ne dépasse pas la portée m. Mathématiquement parlant, le montant de sinistres pris en charge par le réassureur, encore appelé ***récupération***, est :

$$min(max(0, X - a), m) = \begin{cases} 0 & \text{si } X < \text{a} \\ X - a & \text{si a} \leq X < \text{m + a} \\ m & \text{si } X \geq \text{m + a} \end{cases}$$

et le montant de sinistres restant à charge de la cédante est :

$$X - min(max(0, X - a), m) = \begin{cases} X & \text{si } X < \text{a} \\ a & \text{si a} \leq X < \text{m + a} \\ X - m & \text{si } X \geq \text{m + a} \end{cases}$$

1.4.2 AAD et AAL

Dans certains contrats de réassurance, le réassureur limite son engagement sur une tranche en définissant une franchise globale ***AAD (Annual Aggregate Deductible)*** et une limite globale ***AAL (Annual Aggregate Limit)***. Pour clarifier ces deux notions, prenons-nous un exemple où la cédante et le réassureur ont signé, pour l'année n, un traité 10XL5 avec AAD = 10 et AAL = 20. Durant l'année n, 4 sinistres ont survenus. Les charges de sinistres du réassureur (les récupérations) sont calculées comme dans le tableau ci-dessous :

Sinistre n	Montant	10XL5	Cumulé 10XL5	Récupération
1	9	4	4	0
2	20	10	14	4
3	13	8	22	12
4	14	9	31	20

TABLE 1 – tranche 10XL5, AAD=10, AAL=20

Ici, la porté vaut m=10 et la priorité a=5. [a, m+a] = [5,15]. Le premier sinistre a pour montant 9, compris entre [5,15], donc 10XL5 donne (une récupération avant l'application d'AAD et AAL) 9-5=4. Comme 4 < AAD, la récupération = 0. Quand survient le deuxième sinistre de montant 20 > m+a = 15, 10XL5 donne m=10. Le montant cumulé vaut 4+10=14 > AAD, 14-AAD=4 < AAL donc la vraie récupération = 4. Quand survient le troisième sinistre de montant 13 $\in$ [5,15], 10XL5 donne 13-5 = 8. Le montant cumulé vaut 22, 22 - AAD = 12 < AAL donc la récupération = 12. Finalement, un quatrième sinistre de montant 14 survient, 14 $\in$ [5,15] alors 10XL5 donne 14-5 = 9, le montant cumulé vaut 9 + 22 = 31, 31 - AAD = 21 > AAL. Donc la récupération finale est AAL = 20. Le réassureur doit donc 20 à la cédante au titre du traité.

1.4.3 Programme de reconstitution

Un contrat XL avec présence d'AAD et AAL nous amène à la notion ***programme de reconstitutions***. En effet, le montant d'AAL est souvent exprimé en nombre de portées (dans l'exemple ci-dessus, AAL=20 = 2 $\times$ 10, soit un AAL de 2 portées). Nous parlons donc des reconstitutions de la portée pour désigner le niveau de consommation de la cédante par rapport à la limite AAL. Dès que la consommation de la cédante dépasse une fois la portée, elle rentre dans la première reconstitution. Dès que sa consommation dépasse deux fois la portée, elle passe de la première reconstitution à la deuxième reconstitution et ainsi de suite.

On parle de temps en temps de la ***capacité*** du contrat qui est le nombre maximum de portées que la cédante peut consommer. Un contrat à n reconstitutions est équivalent à un contrat de capacité n+1 (portées). (En effet, avant de rentrer dans la première reconstitution, la cédante à déjà consommé une fois la portée).

La prime de réassurance totale associée à un contrat avec AAD et AAL peut être définie de deux façons différentes :

– soit elle est payée une seule fois. On parle de la "prime avec reconstitutions payées d'avance", ou encore "prime avec reconstitutions gratuites".
– soit elle est payée au fur et à mesure de la consommation de la capacité. Au début du contrat la cédante paie la ***prime initiale*** P. A chaque fois que la consommation de la cédante touche à nouveau la portée, elle rentre dans une nouvelle reconstitution et doit payer donc une ***prime de reconstitution***. Le montant de prime totale est la somme de la prime initiale et les primes de reconsitution.

Nous nous intéressons plus particulièrement à la deuxième façon de calculer la prime de réassurance. Chaque prime de reconstitution P_i est calculée en pourcentage de la prime initiale P et est payée proportionnellement à la part de portée consommée. Le programme de reconstitutions est souvent noté "n@p". Par exemple, un traité avec trois reconstitutions, la première à 50% et les deux autres à 100% est noté : "1@50, 2@100". Cela veut dire que ce traité a 3 reconstitutions possibles (i.e. sa capacité est 4 fois la portée). Au début, la cédante doit payer une prime initiale P. Si elle consomme toute la portée pour la première fois, elle rentre dans la première reconstitution. Si dans cette première reconstitution elle consomme x% de la portée, elle doit payer une première prime de reconstitution $P_1 = x\% \times 50\% \times P$. Si elle consomme la totalité de la portée dans la première reconstitution et continue à consommer y% de la portée dans la deuxième reconstitution, elle doit payer : une première prime de reconstitution $P_1 = 50\% \times P$ et une deuxième prime de reconstitution $P_2 = y\% \times 100\% \times P$. Au total elle ne peut pas consommer plus de trois reconstitutions. Sa récupération maximale sera 4 fois la portée et le montant de primes total payé dans ce cas là (qui est aussi le montant maximal de primes de réassurance possible) sera P + 50%P + $2 \times$ 100%P.

Un programme "n@0" correspond à n reconstitutions payées d'avance. On parle de programme avec reconstitutions illimitées lorsque AAL est infini.

1.4.4 Clause de stabilité

L'inflation est un risque pour les deux parties d'un contrat de réassurance : la cédante et le réassureur. Dans les contrats à long terme, si le montant de priorité reste constant au cours du temps, l'impact de l'inflation pourrait être défavorable pour le réassureur parce que les montants de sinistres avec l'inflation dépassent plus souvent la priorité. Au contraire, si la portée reste constante au cours du temps, une inflation pourrait être défavorable pour la cédante parce qu'elle sera moins bien couverte.

Pour limiter le risque d'inflation, les contrats de réassurance sur les branches d'activité à long terme (long-tail LoB), telles que la responsabilité civile, définissent souvent une ***clause de stabilité*** (parfois appelée ***clause d'indexation***) pour permettre à la cédante et au réassureur de maintenir le même ratio de contribution dans le paiement des pertes en cas d'inflation.

On a dans chaque générateur un vecteur d'indexation (dont le premier élément est appelé l'indice de base) et une marge de la clause d'indexation. L'application de la clause d'indexation compose 2 étapes :

Etape 1 : Nous calculons le vecteur des paiements ajustés afin d'obtenir un vecteur de ratios de paiements par année. Ci-dessous l'exemple (source : [4]) avec la marge de la clause d'indexation choisie à 10%.

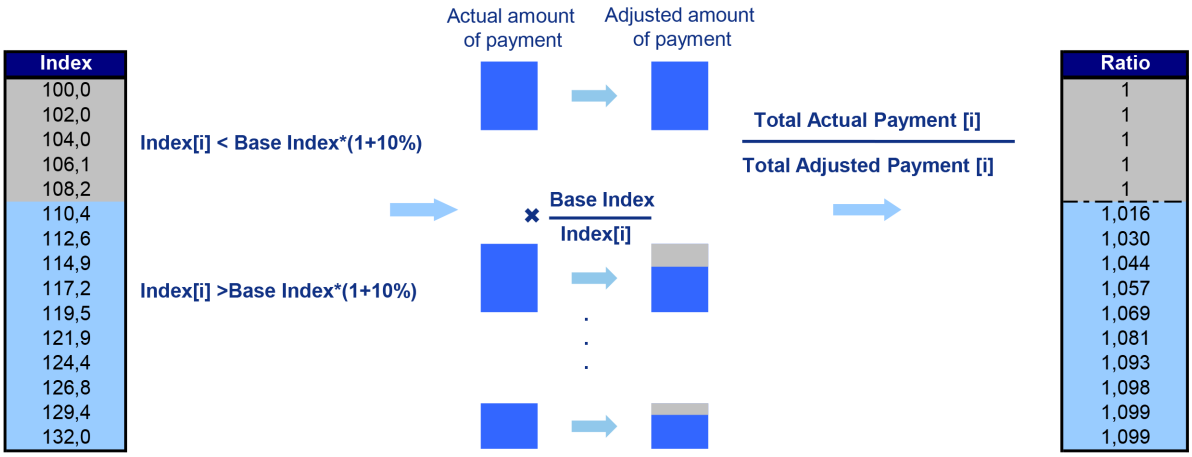


FIGURE 1 – Index Clause 1 : calcul des ratios

Etape 2 : Nous appliquons ce vecteur de ratios au montant de priorité et de portée du contrat.

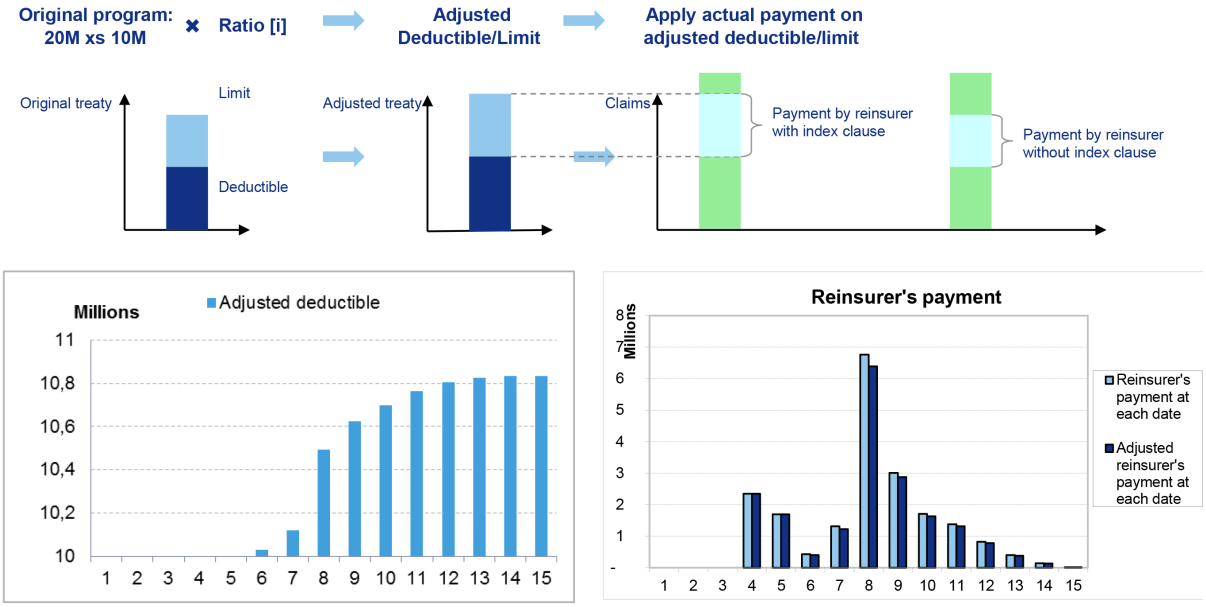


FIGURE 2 – Index Clause 2 : application de ratios sur la portée et la priorité

L'idée générale du fonctionnement de la clause d'indexation est la suivante : le vecteur d'indexation représente l'évolution anticipée de l'indice de prix en tenant compte de l'inflation. la marge de la clause d'indexation représente le taux d'inflation toléré. Si l'indice de prix d'une année de développement dépasse l'indice de base × (1+ taux d'inflation toléré), ce dépassement doit être pris en compte pour la correction de la portée et la priorité.

1.5 Programme et structure de réassurance

Programme de réassurance :

Un programme de réassurance est un ensemble de traités du même type relatifs à un même portefeuille. Par exemple, la cédante peut souscire un programme de réassurance comportant 3 traités XS couvrant des tranches de réassurance comme suivant :

Tranches	Reconstitutions	AAD	AAL
10XL5	1@50, 1@100	10	illimitée
20XL10	2@100	0	illimitée
25XL30	1@100	0	illimitée

TABLE 2 – Exemple de programme de réassurance

Il faut noter dans cet exemple que les AAL sont marquées illimitées, mais elles sont en réalité limitées à cause de la présence des reconstitutions.

Structure de réassurance :

Proprement parlant, la structure de réassurance d'un portefeuille est l'ensemble des contrats de réassurance liés : traités, facultatives, formes alternatives de réassurance. Une structure de réassurance est couramment utilisée pour désigner l'ensemble des programmes relatifs à un portefeuille.

Comme expliqué dans la partie 1.2, les activités de réassurance peuvent mélanger les deux catégories de réassurance. Nous pouvons avoir une structure de réassurance "XL on QS retention" dans laquelle, un programme XL vient en complément d'un programme QS.

2 L'outil Hermès

2.1 Projet Hermès avec la plateforme web

Hermès est un programme interne développé par les membres de l'équipe Actuariat Réassurance d'AXA Global P&C pour modéliser et comparer les structures de réassurance. Il est en fait un gros programme constitué des scripts R, structuré et géré par une surcouche/plateforme développée par l'équipe IT d'AXA Global P&C. Le code final de Hermès est un mélange de code R et code spécifique à la surcouche. L'utilisateur lance Hermès en interagissant avec la plateforme. Un des inconvénients de cette version Hermès est la non-popularité de la surcouche utilisée auprès des entités du groupe AXA. En y ajoutant, cette surcouche ne fonctionne qu'avec les versions R antérieures à 2.15, ce qui limite l'utilisation des nouveaux packages R (notamment le package RCPP pour appeler les fonctions C++ à partir du code R dont je vais parler dans la section 3). Le dernier inconvénient vient de la façon dont Hermès fonctionne actuellement sous la plateforme. En fait, Hermès fontionne fenêtre par fenêtre et nécessite l'interaction de l'utilisateur avec l'interface graphique d'utilisateur (GUI) tout au long d'un lancement du programme Hermès. L'utilisateur après avoir choisi certains inputs doit cliquer sur un bouton dans la première fenêtre pour exécuter un ou plusieurs scripts R, qui vont ensuite renvoyer des outputs utilisés pour proposer les inputs de niveau 2 pour la deuxième fenêtre et ainsi de suite.

Je travaille sur un projet d'optimiser Hermès et l'intégrer dans une plateforme web pour qu'il puisse être utilisé facilement par toutes les entités du groupe AXA. Dans le cadre de ce projet, j'ai dû identifier les grands modules Hermès, les rendre indépendants en modifiant/ajoutant les codes R nécessaires. Le but final est d'avoir un programme Hermès qui ne sera constitué que du code R et qui pourra exécuter une fois fixé tous les inputs dans une première étape. il n'y aura plus de notion fenêtre comme dans l'ancienne version Hermès. C'est la plateforme web qui sera en charge de collecter les inputs saisis ou importés par l'utilisateur et qui les fournira au cœur R. Le programme s'exécute sous R et renvoie les résultats qui seront ensuite affichés sur la plateforme web. Les deux seuls moments de communication entre le cœur R de Hermès et l'interface web est la lecture des inputs et l'affichage des outputs.

2.2 Modules principaux de Hermès

La structure (workflow) de l'ancienne version Hermès est assez compliquée. Nous pouvons cependant identifier trois modules importants de Hermès qui sont : (i) la modélisation des pertes, (ii) la tarification et (iii) le rapport de résultat. Je présente ci-dessous le schéma de Hermès après avoir modifié les scripts R et les regroupé dans les modules afin d'intégrer Hermès dans la plateforme web. Deux modules supplémentaires devraient être exécutés dès que l'utilisateur ouvre Hermès pour gagner du temps. (Pendant que l'utilisateur remplit les inputs, ces scripts vont être exécutés). Le premier module a pour but de charger le package RCPP et de compiler les fonctions C++. Je reviendrai sur ce module dans la troisième section. Ensuite, Hermès va exécuter les scripts du deuxième module qui contiennent les "simples" fonctions utilisées très souvent dans les modules principaux.

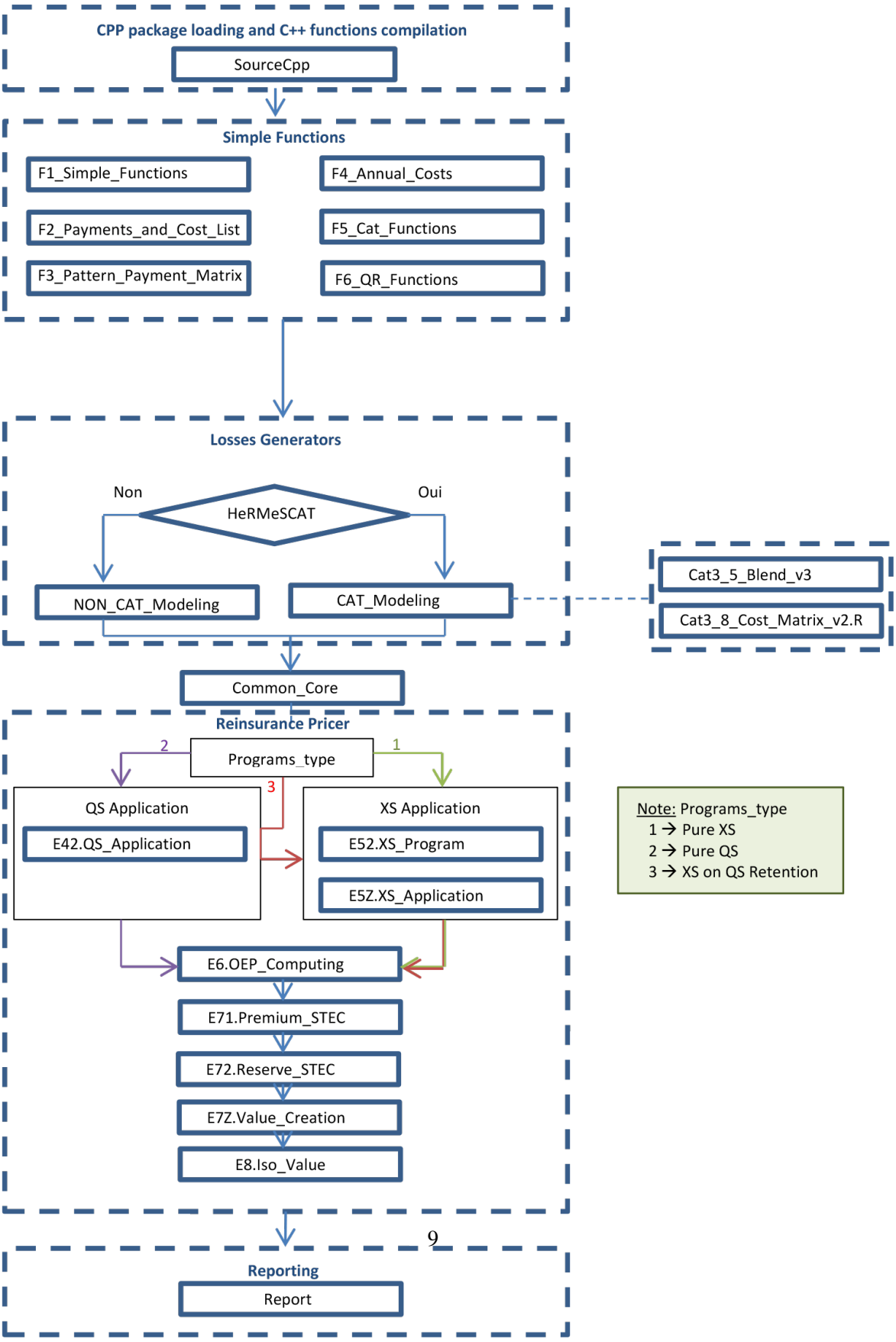


FIGURE 3 – Schéma Hermès (nouvelle version)

Interviennent lors du passage du deuxième module "Simple functions" au troisième module "Losses Generators" les scripts "Web_Platform_Inputs_Collect" et "Initialization". Le premier script sert à récupérer tous les inputs saisis par l'utilisateur via la plateforme web. Le deuxième sert à lire certains paramètres des fichiers sources et initialiser certaines variables globales.

Je vais maintenant rentrer dans les détails des deux modules : la modélisation des pertes et la tarification.

2.3 Modélisation des pertes

Une structure de réassurance s'applique à un portefeuille donné. Afin de tarifer une telle structure, il faut modéliser au préalable la sinistralité générée par ce portefeuille. Nous nous intéressons donc ici aux générateurs des pertes.

Pour pouvoir spécifier le portefeuille étudié, nous avons besoin, au niveau d'inputs saisis par l'utilisateur, les informations suivantes :
- entité (exemple : AXA France)
- année de renouvellement (2013, 2014, 2015, ...).
- type de modélisation (CAT ou Atypique)

Hermès va chercher les fichiers générateurs (de format csv) correspondant dans les répertoires spécifiques dont le chemin est défini par l'ensemble des trois inputs mentionnés ci-dessus.

Chaque année les contrats de réassurance sont renouvelés durant la période de renouvellement. On parle donc de l'année de renouvellement.

Une modélisation CAT correspond à une modélisation "par événement" dans laquelle les pertes sont générées par "Event Set", tandis qu'une modélisation NON CAT (Atypical) est une modélisation "par risque" dans laquelle les pertes sont générées par une approche classique "fréquence $\times$ sévérité".

Comme je n'ai optimisé que la partie de code R pour le générateur Atypqique de Hermès, je ne présente dans ce rapport que la modélisation Atypique. La modélisation CAT sera présentée dans le deuxième rapport avec l'optimisation des scripts R correspondants.

2.4 Modélisation Atypique

Une fois sélectionné l'entité, l'année de renouvellement concernées et le type de modélisation Atypique, Hermès va proposer les ***branches d'activité*** (LoB - Line of Business) disponibles, qui peuvent être :
- Dommage aux biens (DAB)
- Maritime (MARIT)
- Responsabilité civile automobile (RCA)
- Responsabilité civile générale (RCG)

Un fichier de générateurs correspondant à une modélisation Atypique contient en général plusieurs générateurs différents. Chaque générateur est décomposé en certaines composantes :

– le nom du générateur
– la loi de fréquences et ses paramètres
– la loi de sévérité et ses paramètres
– la clause de stabilité
– la cadence de paiement (payment pattern)
– la monnaie utilisée
– le montant du GNPI
– L'attritionnel : le paramètre et la monnaie (les attritionnels sont modélisés par une loi log-normale)

Ce fichier de générateurs sera utilisé, avec les taux de change (stockés dans un fichier de paramètres), pour modéliser la ***matrice de pertes***.

Cette matrice est de taille N × (*Payment_Pattern_Length* + 2) où ***N*** est le nombre de simulations Monte Carlo (dans Hermès, le numéro de simulation i = 1,..., N est appelé year ; N était 10000, il est maintenant passé à 50000 pour avoir des résultats moins volatiles) et ***Payment_Pattern_Length*** est le nombre maximum d'années de développement, i.e. le nombre maximum d'années nécessaires pour pouvoir payer la totalité de la charge ultime. Nous utilisons les matrices et les vecteurs plutôt que les boucles for pour accélérer la vitesse du programme sous R.

2.4.1 Liste des coûts et des fréquences

On commence par simuler les fréquences et les sévérités. Notons G le nombre de générateurs (i.e. le nombre de lois), Hermès va générer la ***liste des coûts et des fréquences*** qui est une liste de taille G dont chaque élément g (g= 1,... G) est encore une liste de deux sous-éléments : un vecteur de fréquences $(f_i^g)_{i=1,...,N}$ et un vecteur de sévérités $(S_j^g)_{j=1,...,M^G}$ où $M^g = \sum_{i=1}^{N} f_i^g$, la taille M^g du vecteur de sévérités dépend donc des réalisations f_i^g de la fréquence.

Le tableau ci-dessous représente les lois de fréquence et les lois de sévérité qui peuvent être trouvées dans un générateur atypique :

Fréquence			Sévérité			
Nom	Para. 1	Para. 2	Nom	Para. 1	Para. 2	Para. 3
Binomiale négative	r	p	Custom			
Poisson	mean		Log Normale	mean	sd	position
			Pareto (tronquée à droite)	min	max	alpha
			Pareto généralisée	shape	scale	position
			Weibull	position	shape	scale

TABLE 3 – Lois de fréquence et lois de sévérité

La loi dite "Custom" pour la sévérité est en fait une fonction de répartition empirique (représentée sous forme d'un vecteur de quantiles) utilisée quand aucune loi paramétrique mentionnée ci-dessus est adaptée à approcher la vraie distribution de sévérité.

2.4.2 Matrice des montants ultimes de sinistres

Après avoir généré la liste des coûts et des fréquences, Hermès génère la ***matrice des montants ultimes de sinistres*** en concaténant (et réordonnant de façon aléatoire pendant la même simulation (year)) les sévérités simulées pour chaque générateur. C'est une matrice dont le nombre de lignes est $\sum_{g=1}^{G} M^g$ et le nombre de colonne est 3 (une pour le numéro du générateur, une pour le numéro de simulation (year) et une pour le montant de charges). Cette matrice est ordonnée par year.

2.4.3 Application des taux de change

Chaque générateur dans un même fichier générateur peut utiliser une monnaie différente des autres, et différentes de celle utilisée dans la structure de réassurance. Il faut donc tout convertir en monnaie de la structure de réassurance. Hermès va multiplier les montants ultimes de sinistres par les taux de change correspondant. Ces taux de change sont extraits du fichier des paramètres.

2.4.4 Application des cadencements de paiement

Pour les branches d'activités dites "longues", la gestion des sinisres peut s'étaler sur une durée de plusieurs années. La charge ultime doit, dans ce cas, être répartie dans le temps. Hermès utilise la décomposition moyenne des temps de paiement de pourcentages cibles de l'ultime d'un sinistre fournie dans chaque générateur. L'estimation des pourcentages payés chaque année de l'ultime d'un sinistre se fait par deux méthodes : la première dite ***cadencement de paiements déterministe*** qui ne varie pas d'un sinistre à l'autre (plus simple à mettre en œuvre), la seconde dite ***cadencement de paiements stochastique*** (plus réaliste).

Je présente dans ce rapport le ***cadencement de paiements déterministe***. Un générateur founit un cadencement de paiements moyen en associant à des pourcentages cibles cumulés $(C_i)_{i=1,...,k}$ un temps moyen (encore appelé temps caractéristique) $(t_i)_{i=1,...,k}$. Nous voulons effectuer la démarche inverse en associant à chaque année de développement un pourcentage de paiements.

Nous commençons par estimer les temps X_i mis pour accomplir les pourcentages cibles cumulées C_i. Hermès suppose que $X_i \sim \mathcal{E}(\frac{1}{t_i})$. Ci-dessous un exemple :

i	C_i	t_i	x_i	x_i cumulé
1	0,05	0,2	x_1	x_1
2	0,25	0,8	x_2	$x_1 + x_2$
3	0,5	13	x_3	$x_1 + ... + x_3$
4	0,75	2,5	x_4	$x_1 + ... + x_4$
5	0,95	2,2	x_5	$x_1 + ... + x_5$
6	1	1,3	x_6	$x_1 + ... + x_6$

TABLE 4 – Exemple de cadencement de paiement

Hermès impose la condition que le temps total de paiement $\sum_{i=1}^{k} x_i$ est majoré par $T_{max} = \sum_{i=1}^{k} t_i$ (dans l'exemple ci-dessus, $T_{max} = 20$). Tant que les réalisations $(x_i)_{i=1,...,k}$ ne satisfont pas cette condition, Hermès retire les lois exponentielles.

Ensuite, pour un temps t donné entre 0 et T_{max}, exprimé en nombre d'années, le pourcentage p_t de sinistre payé est obtenu par interpolation linéaire sur les x_i cumulés. Reprenons l'exemple ci-dessus et supposons que $x_1 + x_2 < 1 < x_1 + x_2 + x_3$. Le pourcentage p_1 du montant ultime de sinistre payé au bout d'un an sera compris entre $C_2 = 0{,}25$ et $C_3 = 0{,}5$. Par interpolation linéaire,

$$p_1 = 0,25 + \frac{1-(x_1+x_2)}{x_3} \times (0,5 - 0,25)$$

Comme nous utilisons la méthode de simulation Monte Carlo, pour un pourcentage cible C_t, Hermès n'effectue pas un seul tirage mais N tirages indépendants de loi exponentielle $(x_i^n)_{n=1,...,N}$. Ainsi les p_t mentionnés ci-dessus sont les moyennes des $(p_t^n)_{n=1,...,N}$.

Nous obtenons une matrice de cadencements de paiements donc le nombre de lignes est G (le nombre de générateur) et le nombre de colonnes est *Payment_Pattern_Length* qui est le nombre maximum d'années de développements des générateurs utilisés. Nous appliquons cette matrice de cadencements de paiements aux montants ultimes de sinistres pour obtenir la ***matrice de paiements par année de développement***, dont le nombre de ligne et N et le nombre de colonne est (*Payment_Pattern_Length* + 2). Les deux premières colonnes représentent le numéro de générateur et le numéro de simulation (year). Cette matrice est ordonnée par year.

2.4.5 Préparation pour la clause de stabilité

Comme expliqué dans la partie 1.4.4, la clause de stabilité interviendra ultérieurement dans le module de tarification. Néanmoins, pour savoir quel ratio appliqué à la portée et la priorité à une année spécifique, nous avons besoin d'une matrice de ratios. Nous établissons d'abord la ***matrice d'indexations corrigées*** de dimension G × *Payment_Pattern_Length*. Nous calculons ensuite :

- la ***matrice des paiements par année de développement cumulés*** à partir de la matrice des paiements par année de développement et
- la ***matrice des paiements par année de développement désindexés et cumulés*** à partir de la matrice des paiements par année de développement et la matrice d'indexations corrigées.

La matrice de ratios utilisées dans le module de tarification pour prendre en compte la clause de stabilité sera le résultat d'une division terme à terme des deux matrices mentionnées ci-dessus.

2.5 Tarification

La tarification est prise en charge par le module Pricer. Il nécessite une (ou plusieurs) structure(s) de réassurance choisie(s) par l'utilisateur. Hermès envisage trois structures de réassurance possibles : (1) Pure XS, (2) Pure QS et (3) XS on QS retention. (Chez AXA, on utilise souvent les abréviations XS/QS plutôt que les vraies abréviations françaises XS/QP ou les vraies abréviations anglaises XL/QS.)

Suivant les types de programmes choisis par l'utilisateur le module de tarification va exécuter les scripts QS_Application et/ou XS_Application.

Les résultats obtenus sont la prime pure et la prime technique sur chaque layer, la probabilité de toucher et de consommer entièrement les reconstitutions, la table OEP/AEP, etc. Les détails seront abordés ultérieurement dans mon deuxième rapport.

3 Modification et optimisation de l'outil Hermès

3.1 Scénarios d'inputs pour la plateforme web

Avec l'introduction de la plateforme web, le fonctionnement de Hermès sera modifié structurellement. L'utilisateur n'interagit avec Hermès via la plateforme web qu'à deux moments : saisir les inputs et récupérer les outputs. Certains scripts R de la version ancienne de Hermès deviennent inutiles et sont remplacés par le travail de la plateforme web. Il faut en contrepartie fournir une liste exhaustive des inputs au cœur R de Hermès dès le début et envisager tous les scénarios d'inputs susceptibles d'être définis par l'utilisateur.

Pour les scénarios d'inputs, j'ai crée un script R qui va chercher dans le répertoire contenant des fichiers de générateurs Hermès et lister :

- toutes les entités, suivies
- des années de renouvellement,
- des branches d'activité et des lois de générateur disponibles (pour la modélisation atypique)
- des périls et des modèles de vendeur disponibles (pour la modélisation CAT)

et enfin écrire un fichier csv qui stocke ce récapitulatif. Ce fichier csv sera renouvelé une fois par an en lançant le script R et sera utilisé par la plateforme web pour proposer des listes de choix dynamiques aux utilisateurs. Par exemple, quand un utilisateur venant de l'entité AXA France se connecte à la plateforme web de Hermès, une liste des années de renouvellement disponibles pour AXA France sera extraite du fichier csv et proposée à l'utilisateur, ensuite quand il choisit une année de renouvellement et le type de modélisation Atypique, une liste des branches d'activité disponibles et les lois de générateurs correspondantes sera extraite du fichier csv et proposée à l'utilisateur.

Pour la lecture des inputs choisis par l'utilisateur via la plateforme web, j'ai rajouté à Hermès un script "$Web\_Platform\_Inputs\_Collect$" qui sera exécutée une fois que l'utilisateur a choisi tous les inputs et qui les récupérera dans un fichier csv (je suppose pour le moment que les inputs collectés par la plateforme web sont tous stockés dans une fichier csv).

3.2 Restructuration des modules Hermès

J'ai effectué des modifications et des regroupements nécessaires sur le programme Hermès pour rendre indépendants ces modules principaux comme présentés dans le schéma Hermès (nouvelle version). Ces modules seront lancés dans l'ordre du schéma conditionnellement aux choix de l'utilisateur.

Cette restructuration des modules Hermès sert dans un premier temps à bien identifier les parties de code R suivant leur fonctionnalité, pour que les gens qui continueront à développer Hermès puissent accéder rapidement à l'endroit souhaité. Elle facilitera dans un second temps l'ajout des modules supplémentaires pour les nouvelles méthodes de modélisation des sinistres sur lesquelles je travaille dans mon deuxième stage. Une méthode de modélisation nécessite un module de générateur R spécifique, mais les outputs des modules de générateurs seront de même format, et seront ensuite tarifés par un seul module Pricer.

3.3 Optimisation des codes R par C++

Après avoir testé Hermès avec plusieurs fichiers de générateurs et plusieurs structures de réassurance, j'ai trouvé que :

- Le temps de calcul de Hermès est potentiellement long avec les branches longues telles que la responsabilité civile (avec présence des cadences de paiements).
- le nombre de simulations Monte Carlo N=10000 est insuffisant pour obtenir des résultats stables. Nous allons passer N de 10000 à 50000.

Pour arriver au cible N=50000 et résoudre le compromis entre la précision des résultats et le temps de calcul du programme, j'ai essayé dans un premier temps à recoder certaines fonctions (ou parties de fonctions) R mais n'ai pas réussi à améliorer le temps de calcul. J'ai donc dans un deuxième temps eu recours au langage C++. L'économie du temps de calcul avec C++ est énorme.

L'idée est de recoder les bouts de codes R gourmands en temps de calcul sous C++, et d'appeler des fonctions C++ à partir du R. J'ai installé Rtools pour avoir un compilateur C++ et j'ai utilisé le package RCPP pour une intégration transparente de R et C++. Les détails sur l'installation et l'utilisation de Rtools et RCPP peuvent être trouvés dans l'annexe A1.

Ci-dessous les améliorations que j'ai faites en utilisant l'interface R/C++.

3.3.1 Amélioration de la fonction Layer_Cost() par C++

Cette fonction R (définie dans le script F4_Annual_Costs) est utilisée dans le module de tarification afin d'appliquer la structure de réassurance : la limite, la priorité (avec la clause de stabilité), AAD et AAL sur les paiements de sinistres projetés dans le temps. Dans cette fonction, la partie de code R qui sert à l'application d'AAD et AAL est couteux en temps de calcul.

J'ai codé la fonction C++ **AAD_AAL_Appli(Reinsurance_Cost, AAD=AAD, AAL=AAL)** ayant pour but de remplacer toute la partie suivante de code R dans la fonction Layer_cost() :

```
# Step1. Cumulating losses at the charge of the reinsurer per simulation year
Mat_Year<-matrix(Reinsurance_Cost[,2],nrow(Reinsurance_Cost),ncol(Reinsurance_Cost)-2)
Per_Year_Cost<-tapply(as.matrix(Reinsurance_Cost[,-(1:2)]),Mat_Year,cumsum) ##both list and array
# Step2. Appliying the AAD and AAL to the cumulated losses per simulation year
Per_Year_Cost<-lapply(Per_Year_Cost,Reinstatements_and_aggregates,AAL=AAL,AAD=AAD)
# Step3. Decumulating the losses after application of aggregates.
Reinsurance_Cost[,3:ncol(Reinsurance_Cost)]<- do.call(rbind,lapply(Per_Year_Cost,Fonction_with_decalage,nbcol=Payment_Pattern_Length))
```

Les deux fonctions Reinstatements_and_aggregates() et Fonction_with_decalage() ont été définies dans le script F1_Simple_Functions comme suivant :

```
fonction_decale=function(x)
```

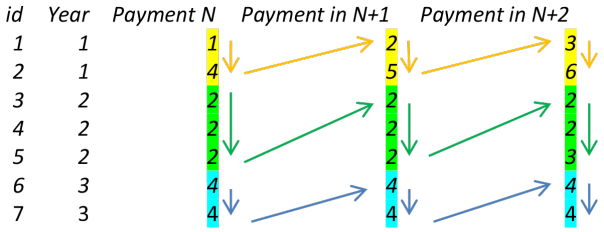
```
{
  result=c(0,x)
  result=result[1:(length(result)-1)]
  return(result)
}

Fonction_with_decalage=function(x,nbcol)
{
  y=fonction_decale(x)
  z=x-y
  return(matrix(z,length(z)/nbcol,nbcol))
}

Reinstatements_and_aggregates=function(AAL,AAD,MatrixIn)
{
  MatrixOut=pmax(pmin(MatrixIn-AAD,AAL),0)
  return(MatrixOut)
}
```

L'idée est d'appliquer AAD et AAL à la matrice Reinsurance_Cost. Prenons un exemple très simple pour comprendre ce que fait cette partie de code R :

Reinsurance_Cost (déjà triée par Year, i.e. numéro de simulation. Dans Hermès, Year prend de valeurs entre 1 et N) :



Etape 1 :

Mat_Year

1	1	1
1	1	1
2	2	2
2	2	2
2	2	2
3	3	3
3	3	3

Per_Year_Cost : cumulé par Year de Reinsurance_Cost.

(Year)									
(1)	1	5	7	12	15	21			
(2)	2	4	6	8	10	12	14	16	19
(3)	4	8	12	16	20	24			

Les flèches représentées dans la matrice Reinsurance_Cost au-dessus expliquent comment Per_Year_Cost a été construite.

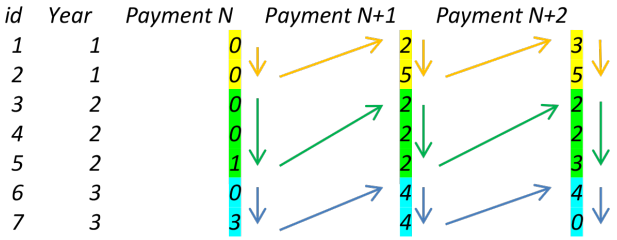
Etape 2 :

Per_Year_Cost (après application d'**AAL = 15** et **AAD = 5**)

(Year)						
(1)	*0*	*0*	*2*	*7*	*10*	*15*
(2)	*0*	*0*	*1*	*3*	*5*	*7*	*9*	*11*	*14*
(3)	*0*	*3*	*7*	*11*	*15*	*15*

Etape3 : C'est l'inverse de l'étape1.

Reinsurance_Cost (après application d'AAL = 15 et AAD = 5) : décumulation par Year de Per_Year_Cost



Cette partie de code est optimale sous R à mon point de vue. On a utilisé les matrices et les fonctions tapply(), lappy() pour éviter les boucles .

Pour l'améliorer, nous proposons de passer par C++. Nous gardons le même principe et créons la fonction **AAD_AAL_Appli()** sous C++ en regroupant les trois étapes précédentes en une seule boucle.

```
1  // [[Rcpp::export]]
2  NumericMatrix AAD_AAL_Appli(NumericMatrix M, double AAD, double AAL){
3    unsigned int i=0;
4    unsigned int j=0;
5    unsigned int ncol= M.ncol();
6    unsigned int nrow= M.nrow();
7    NumericMatrix Res(nrow, ncol);
8    unsigned int i_old=0;
9    unsigned int year=M(0,1);
10   double s=0;
11   double prec=0;
12   double temp=0;
13   for(unsigned int i = 0; i < nrow; i++){
14     Res(i,0)= M(i,0);
15     Res(i,1)= M(i,1);
16   }
17   while(i_old<nrow){
18     year=M(i_old,1);
19     i=i_old;
20     s=0;
21     prec=0;
22     while(j < ncol-2){
23       i=i_old;
24       while(M(i,1)==year){
25         s+= M(i,j+2);
```

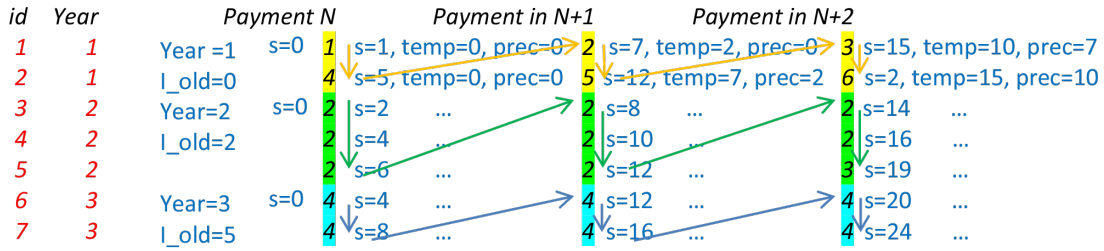
```
26          if(s<=AAD){
27            temp=0;
28          } else{
29            if (s<AAD+AAL)
30              temp = s - AAD ;
31            else
32              temp = AAL;
33          }
34          Res(i,j+2)= temp - prec;
35          i++;
36          prec=temp;
37
38        }
39        j++;
40      }
41      j=0;
42      i_old=i;
43    }
44    return Res;
45  }
```

Le paramètre **M** prend le rôle de la matrice **Reinsurance_Cost** avant l'application d'AAL et AAD. La fonction **AAD_AAL_Appli()** renvoie comme résultat la matrice **Res** qui prend le rôle de la matrice **Reinsurance_Cost** après application de AAL et AAD.

La première boucle for sert à recopier les deux premières colonnes de **M** dans **Res**.

La grande boucle **while** parcourt les lignes de **M**, la petite boucle **while** dedans parcourt les colonnes de **M**. L'semble de la partie de code dans cette boucle **while** suit le chemin défini par les flèches que nous avons représentées ci-dessus. **s** représente le paiement cumulé tant qu'on reste dans une même année, i.e. une même simulation (Etape1 de la partie de code R). Elle est réinitialisée à 0 dès que **year** change. **temp** représente le paiement cumulé après l'application d'AAL et AAD (Etape 2 de la partie de code R). **prec** correspond à la valeur de **temp** à l'itération précédente. La phase de décumulation (Etape3 de la partie de code R au-dessus) est ici établie par la ligne **Res(i,i+2)= temp-prec**.

Reprenons-nous M = Reinsurance_Cost définie tout au début, AAD=5 et AAL=15.



En prenant temp-prec à chaque itération, nous obtenons le résultat :

Res

id	Year	Payment N	Payment N+1	Payment N+2
1	1	0	2	3
2	1	0	5	5
3	2	0	2	2
4	2	0	2	2
5	2	1	2	3
6	3	0	4	4
7	3	3	4	0

3.3.2 Amélioration de la fonction Layer_price2() par C++

La fonction C++ **Exhaustion_Prob_CPP(Reinstatements_paid, AAD=0, AAL=1, N)** a pour but de remplacer toute la partie suivante de code R dans la fonction **Layer_price2()** :

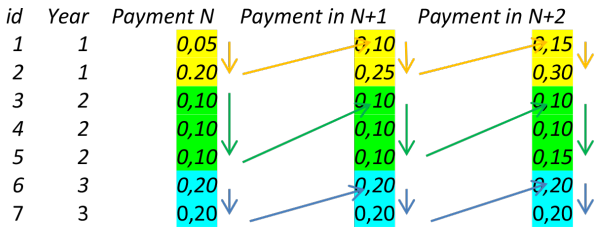
```
# Step1
Mat_Year<-matrix(Reinstatements_paid[,2],nrow(Reinstatements_paid),ncol(
    Reinstatements_paid)-2)
Per_Year_Reinstatements<-tapply(as.matrix(Reinstatements_paid[,-(1:2)]),Mat
    _Year,cumsum)
# Step2
temp<-lapply(Per_Year_Reinstatements,Reinstatements_and_aggregates,AAL=1,
    AAD=0)
# Step3
resultat=mean(lapply(temp,max)==1)
```

Cette partie de code R a pour but de calculer la probabilité de consommmer la totalité de la portée (cas où il n'y a pas de reconstitution et AAL = infini).

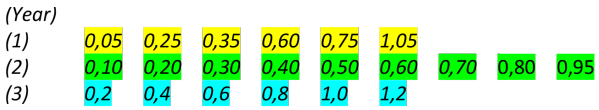
Reinstatement_paid est la matrice des récupérations (i.e. des paiements du réassureur) projetés dans le temps, exprimée en pourcentage de la portée. Je prend l'exemple suivant pour la matrice des récupérations projetés dans le temps :

id	Year	Payment N	Payment in N+1	Payment in N+2
1	1	1	2	3
2	1	4	5	6
3	2	2	2	2
4	2	2	2	2
5	2	2	2	3
6	3	4	4	4
7	3	4	4	4

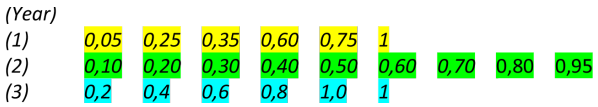
Avec la portée = 20, la matrice des récupérations exprimées en pourcentage de la portée (**Reinstatement_paid**) est :



On cumule cette matrice pour avoir la liste **Per_Year_Reinstatements** :



Ensuite, on applique AAD = 0, AAL = 1 pour avoir la liste **temp** :



On regarde les maximums sur chaque vecteur de la liste temp (i.e. les derniers éléments sur chaque vecteur de la liste). Si le maximum est 1, la portée a été consommé entièrement. On en déduit la probabilité de consommer la totalité de la portée. Dans notre exemple, la portée a été entièrement consommée pour year (1) et (3). La probabilité de consommer entièrement la portée est donc $\frac{2}{3} = 0{,}667$.

Ci-dessous la fonction codée en C++ pour remplacer la partie de code R :

```
double exhaustion_prob_CPP(NumericMatrix M, double AAD, double AAL, double
    N){
  unsigned int ncol= M.ncol();
  unsigned int nrow= M.nrow();
  double s;
  unsigned int i=0;
  unsigned int j=0;
  unsigned int i_old=0;
  unsigned int year=M(0,1);
  double res=0;

  while(i_old<nrow){
    year=M(i_old,1);
    i=i_old;
    s=0;
    while(j < ncol-2){
      i=i_old;
      while(M(i,1)==year){
        s+= M(i,j+2);
        i++;
```

```
20
21        }
22
23        j++;
24
25      }
26      j=0;
27      i_old=i;
28      if (s >= (AAD + AAL)) res ++ ;
29    }
30
31
32    return res/N;
33  }
```

Il suffit d'appeler **Exhaustion_Prob_CPP(M, AAD=0, AAL=1, N)**. La raison pour laquelle j'ai mis AAD et AAL comme paramètres plutôt que de prendre directement AAD = 0 et AAL =1 dans la fonction est pour réutiliser cette fonction dans le cas avec reconstitutions. Dans le cas avec reconstitutions, le code R est un peu plus compliqué mais le raisonnement reste le même. On parcourt par une boucle les reconstitutions et on calcule la probabilité de consommer entièrement la portée pour la première fois (i.e. on va rentrer ensuite dans la première reconstitution), la probabilité de consommer entièrement la première reconstitution, la probabilité de consommer la deuxième reconstitution, etc. Dans chaque parcours de la boucle for, AAL reste 1, mais AAD est incrémenté (pour refléter le fait qu'on passe d'une reconstitution à la suivante).

3.3.3 Comparaison des résultats

J'ai testé les fonctions C++ présentées ci-dessus dans Hermès (N=10000) avec une traité XS de 2 layers sur 6 générateurs de branche longue (12 années de développement). Pour la fonction Layer_Cost(), sur chaque layer, le temps de calcul passe de 7,35s à 0,36s. Pour la fonction Layer_Price2(), le temps de calcul se réduit énormément aussi mais n'est pas encore optimal parce qu'il reste encore une partie de code couteuse en temps de calcul que je vais optimiser et présenter dans mon deuxième rapport. L'économie du temps de calcul est encore plus efficace quand N passe de 10000 à 50000.

Pour utiliser C++ dans R, il faut néanmoins charger le packpage RCPP (0,25s) et compiler les fonctions C++ (une dizaine de secondes) mais cela peut être fait une fois unique tout au début pendant que l'utilisateur saisit les inputs.

Références

[1] Stanislas RAY *Optimalité des structures de réassurance - Mémoire d'actuaire*

[2] Guillaume GORGE *Insurance Risk management and Reinsurance*

[3] Dimitri MINASSIAN *Hermès Guideline*

[4] AXA Global P&C *Index/ Stability Clause*

Annexe

A1. Détails sur l'installation et l'utilisation de Rtools et RCPP

Installation de Rtools

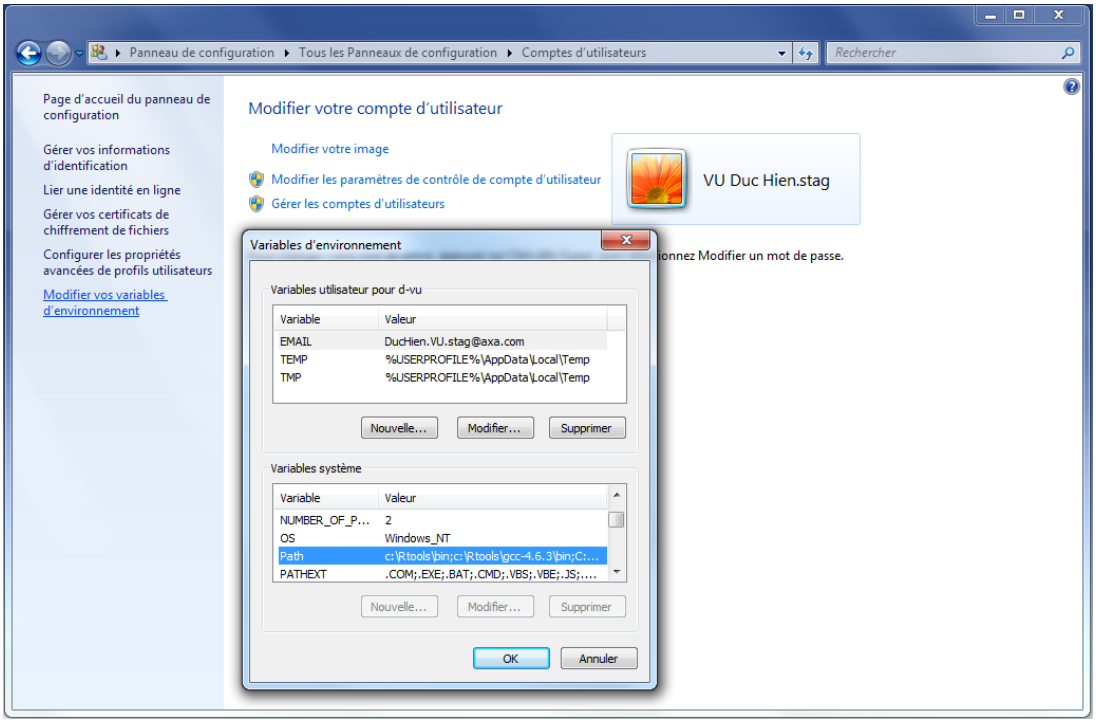
Nous avons besoin d'installer Rtools pour avoir le compilateur C++ :

http ://cran.r-project.org/bin/windows/Rtools/

J'ai téléchargé la dernière version Rtools31.exe (compatible avec les versions R 3.0.x à 3.1.x).

Après l'installation de Rtools, il faut changer la variable de système : PATH pour que le compilateur C++ inclus dans Rtools soit reconnu par R.

- Sur les ordinateurs qui ne demandent pas de droit d'administrateur, ça se fait automatiquement.
- Sur les ordinateurs AXA, nous avons besoin du droit d'administrateur pour le faire :



Il suffit de rajouter dans Path : « c :\Rtools\bin ; c :\Rtools\gcc-4.6.3\bin ; »

Pour vérifier que Rtools soit prêt à utiliser en R : Ouvrir une session R et taper la commande :

```
> Sys.getenv('PATH')
[1] "c:\\\\Rtools\\\\bin;c:\\\\Rtools\\\\gcc-4.6.3\\\\bin;...
```

Pour plus de détails sur Rtools : https ://github.com/stan-dev/rstan/wiki/Install-Rtools-for-Windows

Installation du package Rcpp

Le package Rcpp fournit des fonctions R ainsi que des classes C++ qui offrent une intégration transparente de R et C++. Ce package n'est utilisable que pour les versions récentes de R (à partir de R3.0.x). http ://cran.r-project.org/web/packages/Rcpp/index.html

J'ai téléchargé Rcpp 0.11.3.zip pour Windows binaries

Pour installer le package Rcpp : Dans la fenêtre de la session R, choisir packages, Installer le(s) package(s) depuis des fichiers zip...

Utilisation de Rcpp

Les fonctions C++ à appeler seront écrites dans un fichier .cpp à part. Ci-dessous un exemple : le fichier « test_rcpp.cpp » que j'ai écrit :

```
#include <Rcpp.h>
using namespace Rcpp;

// [[Rcpp::export]]
NumericMatrix just_try(NumericMatrix M, double AAD, double AAL){
// codes C++
}
```

Les deux en-têtes sont indispensables :

```
#include <Rcpp.h>
using namespace Rcpp;
```

Dans ce fichier, nous pouvons définir plusieurs fonctions C++. Chaque fonction devrait commencer par

```
// [[Rcpp::export]]
```

(Attention : ce n'est pas un commentaire).

Note : NumericMatrix et NumericVector sont les types spécifiques pour définir les matrices et les vecteurs dans Rcpp.

Pour appeler cette fonction C++ à partir du R :

```
library(Rcpp)
sourceCpp("test_rcpp.cpp")
M=matrix(bla bla)
M_modified= just_try(M, AAD=0, AAL=Inf) # par exemple
```

Pour plus de détails sur le package Rcpp : http ://cran.r-project.org/web/packages/Rcpp/Rcpp.pdf

A2. Liste des abréviations

1. P&C : Property & Casualty (IARD et Responsabilité Civile)
2. SCR : Solvency Capital Requirement (Capital requis pour la Solvabilité)
3. QS : Quote Share (QP : Quote Part)
4. XL : Excess of Loss (XS : Excédent de Sinistre)
5. GNPI : Gross Net Premium Income (montant des primes collectées par la cédante sur le portefeuille réassuré)
6. AAD : Annual Aggregate Deductible (franchise globale)
7. AAL : Annual Aggregate Limit (limite globale)
8. LoB : Line of Business (branche d'activité)
9. OEP : Occurrence Exceedance Probability
10. AEP : Aggregate Exceedance Probability