EURIA
Euro-Institut d'Actuariat

siapartners

BUREAU D'ETUDE

MASTER 1 Actuariat

Application de méthodes de machine learning au provisionnement non-vie

Aman-Yah Andréa
Ehui
Mayoro FALL

Encadrant en Entreprise
Romain Laily(Actuaire)
Wassim Youssef(Actuaire)
-
Encadrant EURIA
Franck Vermet(Enseignant-Chercheur)
Pierre Ailliot (Enseignant-Chercheur)

Rapport 2017-2018

Remerciements

Nous tenons tout d'abord à remercier toutes les personnes qui nous ont accompagnés de près ou de loin afin de bien mener ce projet à son terme.

Aussi, nous remercions M Romain et M Wassim d'avoir proposé ce sujet intéressant qui nous a permis de comprendre le processus de provisionnement en assurance, mais aussi de développer et mettre en application des connaissances en machine learning.

Nous souhaitons remercier ensuite M Vermet et M Ailliot pour leur soutien et leur encadrement tout au long de ce projet. Nous leur sommes reconnaissants pour toute la valeur ajoutée qu'ils nous ont apportée.

Résumé

Pour calculer la charge ultime d'un portefeuille, les techniques les plus couramment utilisées agrègent les données par année de survenance et année de développement. Ces modèles présentent l'avantage d'être simples à utiliser, mais ils masquent une partie de l'information. Afin de prendre en compte l'ensemble des facteurs pouvant influer sur la cadence des paiements, nous avons établi plusieurs modèles permettant de décrire l'évolution des sinistres en réalisant des prédictions pour chaque sinistre de manière individuelle.
Le développement d'un sinistre est décrit par un ensemble de montants de charges évaluées à un instant donné. Pour modéliser les durées de paiement, nous avons utilisé des modèles tels que les modèles linéaires généralisés, et les montants associés ont été ajustés par d'autres types de modèles. Nous avons ensuite cherché à comparer les résultats obtenus avec les méthodes de provisionnement classiques.

Mots clés : Provisionnement, Chain Ladder, Mack, GLM, Modèles Individuels, apprentissage statistique

Abstract

To calculate the ultimate amount of claims of a portfolio, the most commonly used techniques aggregate the data by year of occurrence and year of development. These models have the advantage of being simple to use, but they hide some of the information. In order to take into account all the factors that may affect the payments, we have established several models to describe the evolution of claims by making predictions for each claim individually.
The development of a disaster is described by a set of amounts of claims evaluated at a given time. In order to model the payment times, we used models such as the generalized linear models. The associated amounts were adjusted by other types of models. We then tried to compare the results obtained with the classical provisioning methods.

Keywords : Provisioning, Chain Ladder, Mack, GLM, individual models, Machine learning

Table des matières

Introduction

Une assurance est un service qui fournit une prestation lors de la survenance d'un événement incertain et aléatoire souvent appelé "risque". La prestation, généralement financière, peut être destinée à un individu, une association ou une entreprise, en échange de la perception d'une cotisation ou prime. Les risques peuvent être corporels (maladies, accidents entraînant parfois des incapacités de travail, décès) ainsi que des préjudices matériels et moraux qui en résultent pour la victime et ses proches. D'autres événements frappent l'homme dans ses biens (incendie, vol, accidents) entraînant ainsi des dégâts matériels ou des pertes de revenus.

De par sa nature aléatoire, le risque encouru n'est pas connu d'avance. Ainsi, son éventuelle coût en est de même inconnu. C'est pour cela que plusieurs méthodes sont mises en place pour évaluer ces montants futurs. Ce sont des méthodes tels que Chain Ladder et Mack , qui sont basés sur des hypothèses très fortes. Les méthodes classiques utilisent des données agrégées observées à périodicité constante.

L'émergence des méthodes de machine learning offre une nouvelle voie aux assureurs de le cadre du calcul des provisions non-vie. L'approche étudiée ici est donc différente, il s'agit d'étudier des données détaillées et de revenir au sinistre individuel. Ainsi, au travers de cette étude, nous essaierons d'instaurer des méthodes de prédiction de l'évolution des coûts des évènements incertains résultant du risque automobile, et ce en se basant sur des méthodes d'apprentissages statistiques. Ces méthodes pourront concurrencer les méthodologies existantes.

Chapitre 1

Assurance non vie en France

1.1 Généralités

L'assurance non-vie regroupe les opérations d'assurance qui n'ont pas pour objet la vie de l'assuré. Elle est principalement composée des assurances de choses ou de biens, des assurances de responsabilité ou de dettes, et des assurances de personnes. La principale différence opposant l'assurance non-vie à l'assurance vie est la survenance même du sinistre, qui est le plus souvent certain en assurance vie, tandis qu'il est juste probable en assurance non-vie. De plus, en assurance non-vie [1], le coût du sinistre est rarement connu, ce qui est encore une particularité propre.

L'assurance non vie est scindé en plusieurs branches (Line of business) qui sont :

- RC Automobile
- Automobile autre
- Assurances maritimes, aériennes et transports
- Incendie et autres dommages aux biens
- RC Générale
- Crédit Caution
- Protection juridique
- Assistance
- Pertes pécuniaires diverses
- Réassurance non proportionnelle MAT
- Réassurance non proportionnelle de dommages aux biens
- Réassurance non proportionnelle de RC

Aussi tout au long de ce Bureau d'étude nous allons nous concentrer sur la branche automobile de l'assurance.

1.2 Faits et chiffres

Sur le marché français, les assurances de biens et de responsabilité représentent un peu plus de 25% de la collecte des assureurs (en 2015). Elles font référence aux assurances IARD (Incendie, Accidents, Risques Divers). Ce secteur de l'Assurance s'est fortement développé avec notamment la branche automobile et celle de la couverture des dommages aux biens des particuliers.

Aussi, les assurances automobiles concentrent l'essentiel du marché des assurances non-vie et constituent près de la moitié des contrats souscrits avec près de 21 milliards d'euros de cotisations en 2015 pour près de 41 millions de véhicules assurés .

Du côté des acteurs de l'assurance non vie en France, ces dernières années ont été marquées par le développement des canaux de distribution même si près de 50% des cotisations d'assurance de biens et de responsabilités restent versées par des sociétés d'assurances avec intermédiaires.

Cependant la rentabilité des assurances de biens et responsabilité présente une tendance à la baisse ces dernières années à l'image des sociétés d'assurance de dommages dont la rentabilité s'élevait à près de 9% en 2015 alors qu'en 2005, elle atteignait presque 14%.

1.3 Provisionnement

Dans l'optique de faire face à la perte engendrée par le paiement futur des sinistres, l'assureur se doit de constituer des provisions. Le provisionnent de manière générale, consiste à constituer des provisions, des réserves. En assurance non vie l'objectif principal est d'estimer les dépenses nécessaires au règlement de l'ensemble des sinistres. Les provisions techniques matérialisent la capacité de l'institution à faire face au règlement intégral des engagements pris envers les assurés et représentent la part la plus importante du passif d'une compagnie d'assurance. Celles-ci sont donc très importantes, d'autant plus qu'elles représentent une part importante du bilan d'un assureur.

ACTIF	PASSIF
Placements	Fonds Propres Provisions Techniques

TABLE 1.1 – Bilan simplifié d'une compagnie d'assurance

Il existe plusieurs types de provisions en assurance non-vie. ce sont par exemple :

— PSAP : Provisions pour sinistres à payer,

— PPNA : Provisions pour primes non acquises,

— PRAE : Provisions pour recours à encaisser,

pour n'en citer que quelques exemples. Aussi pour trouver le montant de ces provisions, il existe plusieurs méthodes dont les plus courantes qui seront détaillées par la suite.

Les provisions qui vont nous intéresser plus particulièrement sont les provisions pour sinistre à payer.

Chapitre 2

Étude statistique de la base de données

2.1 Présentation de la Base de données

l'ensemble des données utilisées proviennent exclusivement de la base de données SIA Partners qui nous a été fournie. Celle ci à été collectée sous le format RDS. La difficulté première que nous avons rencontrée a été de trouver un moyen d'ouvrir ce genre de donnée. Aussi La fonction readRDS de R permet de surmonter cette difficulté.

Une fois les données chargées on peut observer le panel de variables suivant sous les formats spécifiés suivants :

Nom de la variable	Descriptif	format
Vue	Année de vue du sinistre	numérique
Fin_Vue	Fin de la vue en cours	numérique
Numéro_sinistre	Numéro du sinistre	caractère
Date_de_surv	Date de survenance	numérique
Date_de_décla	Date de déclaration	Date
Etat	Etat du sinistre	caractère
Charge	Charge globale du sinistre	numérique
Type_de_sinistre	Type de sinistre	caractère
Type_répar	Type de réparation post-sinistre	caractère
Permis	Type de permis	caractère
Tiers_acc	Véhicule tiers accidenté	caractère
Tiers_domm	Dommage sur les tiers	caractère
Constat	Type de constat rédigé	caractère
Police	Intervention de la police après l'accident	caractère
Chargement_veh	Type de véhicule	caractère
Blessés	Existence de blessés	caractère
Usage	Usage du véhicule	caractère
Type_acc	Lien de parenté entre assurés et tiers	caractère
Type_m	Type de contrat	caractère
Assur	Tiers assuré	caractère
Prop	Type de garanties souscrites	caractère
Exp	Type d'expertise post-accident	caractère
Expa	Type d'expertise post-accident réalisée	caractère
Taux1	Taux de responsabilité constat	caractère
Taux2	Taux de responsabilité gestionnaire	caractère
Duree_sin	Duree du sinistre	caractère
Taux3	Taux de responsabilité final	caractère
Degat	Type de dégat du véhicule	caractère
Lieu	Type de lieu du sinistre	caractère

TABLE 2.1 – Ensemble des variables

La base de données contient 658050 lignes pour 29 colonnes. Ayant 78579 sinistres différents, chaque ligne correspond à une évolution d'un sinistre, un sinistre pouvant être mis à jour plusieurs fois dans la même année.

2.2 Statistiques descriptives et première sélection de variables

Dans le but de comprendre les données mises à notre disposition nous avons décidé d'effectuer une des statistiques descriptives sur notre base de données.

2.2.1 Description

Pour chaque variable explicative de notre base de données, nous avons pu étudier sa distribution ou répartition dans la base de données ainsi que la variation en fonction de la charge des sinistres. On obtient par exemples les résultats suivants :

Etat

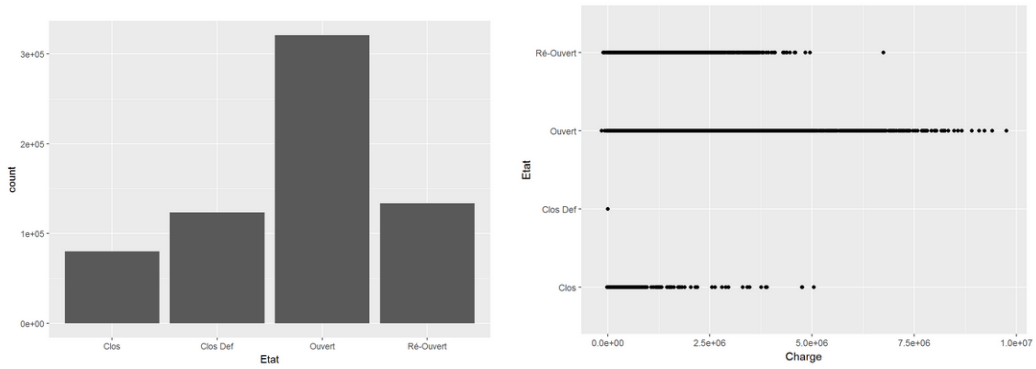


FIGURE 2.2.1 – variable ETAT

Pour cette variable par exemple, cette étude permet d'avoir des informations telles que :

- un sinistre définitivement clos à une charge nulle
- la majorité des sinistres sont encore ouverts
- tous les états sont bien représentés dans la base

Type de sinistre

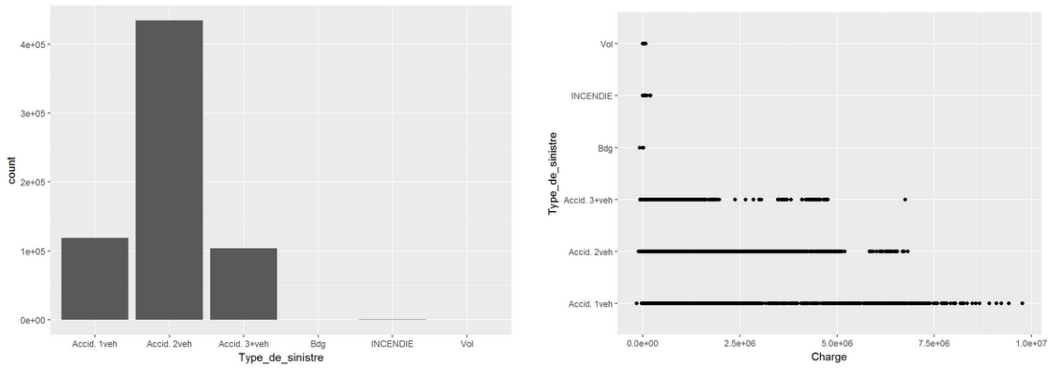


FIGURE 2.2.2 – variable TYPE DE SINISTRE

Pour cette variable, on peut observer que les modalités "Accident entre 2 vehicules", "Accident entre 3 vehicules" et "Accident avec 1 vehicule" sont les plus représentées dans la base au détriment des autres modalités qui le sont très peu. De plus la charge des modalités peu représentées est très proche de zéro.

Type de Réparation

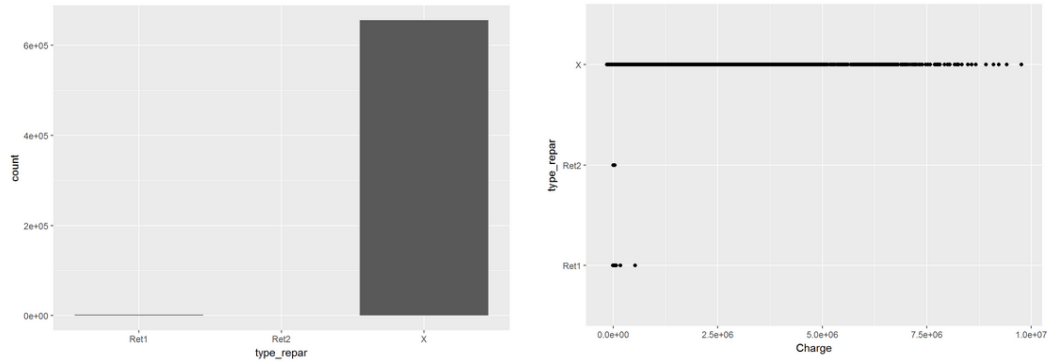


FIGURE 2.2.3 – variable TYPE DE REPARATION

Il existe trois modalités différentes pour cette variable. Cependant, la modalité "X" qui correspond à une absence d'information est la plus présente dans la base de données. Cette information peut donc nous aider à retirer cette variable de notre étude.

Lieu

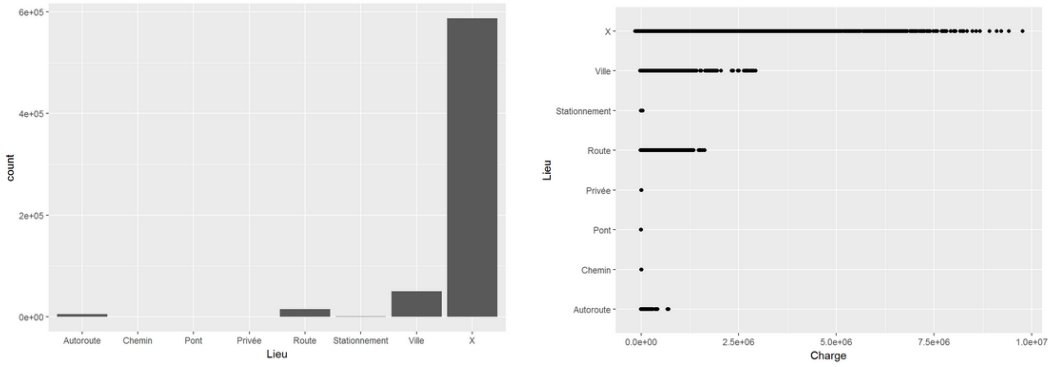


FIGURE 2.2.4 – variable LIEU

Il y a 8 modalités pour cette variable. Cependant comme pour la variable "Type de Réparation", la modalité étant représentée en majorité dans cette base de données est la modalité "X" qui représente le fait que l'information ne soit pas connue. Ainsi par un raisonnement analogue, cette variable pourra être retirée de l'étude.

L'ensemble des études des variables peut être trouvée en annexes.

2.2.2 Une première sélection de variables

Sur la base de l'étude statistique réalisée sur la base de données, on décide de sélectionner les variables explicatives suivantes : Les critères de sélection de ces variables sont principalement la sur-représentation des modalités qui désignent l'absence d'information et l'absence de relation apparente avec la charge.

```
$x
 [1] "Type_de_sinistre" "Expa"          "dureeSin"      "Taux3"         "Type_acc"      "Permis"
 [7] "Usage"            "Tiers_acc"     "Tiers_domm"    "Type_m"        "Police"        "Chargement_veh"
[13] "blesse"           "Vue"           "Fin_Vue"       "Date_de_surv"  "an_de_surv"    "annedev"
[19] "Charge_prec"
```

FIGURE 2.2.5 – variables sélectionnées

2.3 Traitement de la base de données

2.3.1 Choix du montant maximum

La première étape dans le traitement de notre base de données a été de retirer de la base tous les sinistres dont le montant cumulé dépassait un certain seuil à un certain moment. Pour cela nous sommes passés par 2 méthodes qui sont :

- l'étude des valeurs extrêmes avec la méthode de dépassement de seuil
- l'étude de la densité des charges dans la base de données

La méthode de dépassement de seuil (Peak Over Threshold, POT)

La méthode de dépassement de seuil des valeurs extrêmes ([2]) consiste à étudier la loi de l'excès au delà d'un seuil. En d'autres termes, elle consiste à observer toutes les valeurs des réalisations qui excèdent un certain seuil u élevé. La difficulté de cette méthode consiste à choisir ce seuil$u$. On cherche à partir de la loi F de X(la charge) à définir une loi conditionnelle F_u par rapport au seuil u pour les variables aléatoires dépassant ce seuil. On définit alors la loi conditionnelle des excès F_u par

$$F_u(y) = \mathbb{P}(X - u <= y | X > u) \tag{2.1}$$

Le seuil que l'on choisira peut nous servir par la suite de montant maximum des charges.Le seuil u doit être assez grand pour que l'approximation définie par 2.1 soit valide. Ce seuil ne doit pas trop élevé pour garder un nombre suffisant de dépassements pour estimer les paramètres du modèle. Généralement, u est déterminé graphiquement en exploitant la linéarité de la fonction d'excès moyen $e_F(u) = \mathbb{E}[X - u | X > u]$ Celle ci nous fait étudier le graphique suivant :

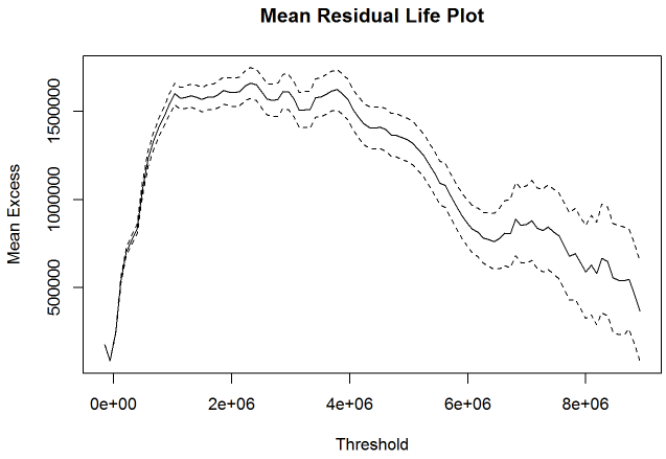


FIGURE 2.3.1 – fonction d'excès moyen

le graphique représenté ci-dessus permet de visualiser une cassure nette de la linéarité de la fonction d'excès moyen $e_F(u)$ à partir de 1 million €. On peut donc supposer qu'à partir de ce montant, les charges ont une toute autre distribution ou loi qui leur est associée. Le seuil à choisir doit donc être au maximum égal à un million.

La densité des charges

La densité des charges présentes dans la base de données est la suivante On remarque que la distribution est très condensée vers la gauche. En effet, la majorité des charges cumulées n'excède pas 500000 €. Aussi,une fois qu'on retire tous les sinistres dont le montant cumulé dépasse un million à un certain moment on a la densité suivante :

La distribution reste condensée vers la gauche mais est déjà beaucoup mieux répartie. On peut aussi affirmer déjà que la loi des charges des sinistres ne suit pas une loi gaussienne.

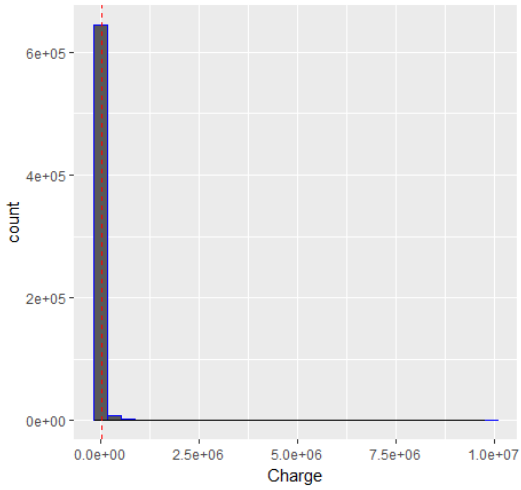


FIGURE 2.3.2 – densité des charges

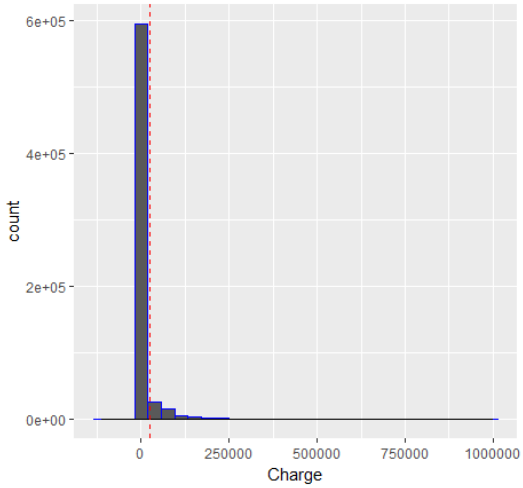


FIGURE 2.3.3 – densité des charges après retrait des sinistres ayant une charge de plus de 1 million

2.3.2 Transformation et construction de variables

Afin de poursuivre notre étude, plusieurs variables supplémentaires sont nécessaires. Ce sont :

Année de survenance

Cette variable est obtenue en récupérant dans la variable Date_de_déclaration, l'année associée.

Année de développement

Cette variable est calculée à partir de l'année de survenance et de la Vue à laquelle on se place.

$$annedev = Vue - an\_de\_surv \tag{2.2}$$

Elle correspond à l'âge du sinistre. Autrement dit, elle permet de se situer dans le temps pour chaque sinistre.

Charge précédente

Cette variable est obtenue en récupérant les charges des lignes $i-1$, tout en s'assurant que pour un nouveau sinistre la charge précédente soit nulle

Changement d'année (chgtreg)

Cette variable permet de connaître le dernier règlement dans l'année si chgtreg = 1 il s'agit du dernier règlement. Elle est construite comme suit :

$$chgtreg = min(1, Fin\_Vue - Vue) \tag{2.3}$$

règlement

Cette variable correspond au règlement effectué pour un sinistre considéré. Elle est construite comme suit :

$$reglement = charge - chargeprcdente \tag{2.4}$$

2.3.3 Étude des corrélations

Étudier la corrélation entre des variables aléatoires ou statistiques numériques, c'est étudier l'intensité de la liaison qui peut exister entre ces variables.Le type de relation utilisé ici est est la relation affine. Ainsi, afin de décrire les variables explicatives présentes dans la base de données, il est aussi nécessaire de se rendre compte des relations affines qui existent entre elles.

étude des corrélations des variables quantitatives

On ne peut calculer des corrélations qu'entre variables quantitatives. Les relations observées sont les suivantes :

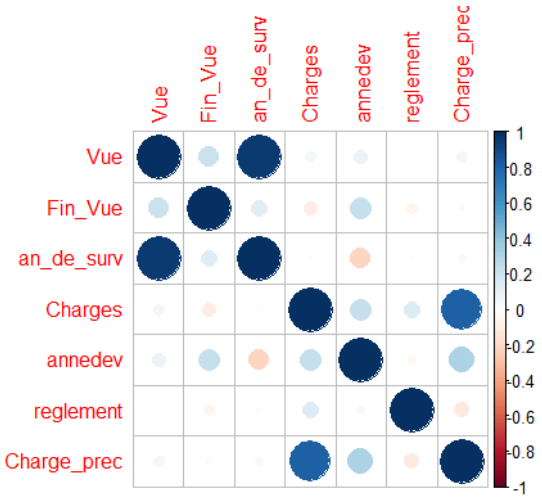


FIGURE 2.3.4 – corrélation entre variables quantitatives

La charge est la variable qui nous intéresse le plus. On remarque qu'elle est principalement corrélée à la charge précédente, mais aussi à l'année de développement et dans une moindre mesure au règlement.

étude des corrélations entre toutes les variables

Afin d'étudier les corrélations avec toutes les variables explicatives y compris celles qui sont qualitatives, on procède à une transformation des variables qualitatives en variables quantitatives. On réalise cette transformation en remplaçant chaque modalité par une valeur numérique. Par exemple, pour la variable binaire "Blessé", un "OUI" correspond à 1 et donc "NON" à 0. On obtient donc les corrélations suivantes :

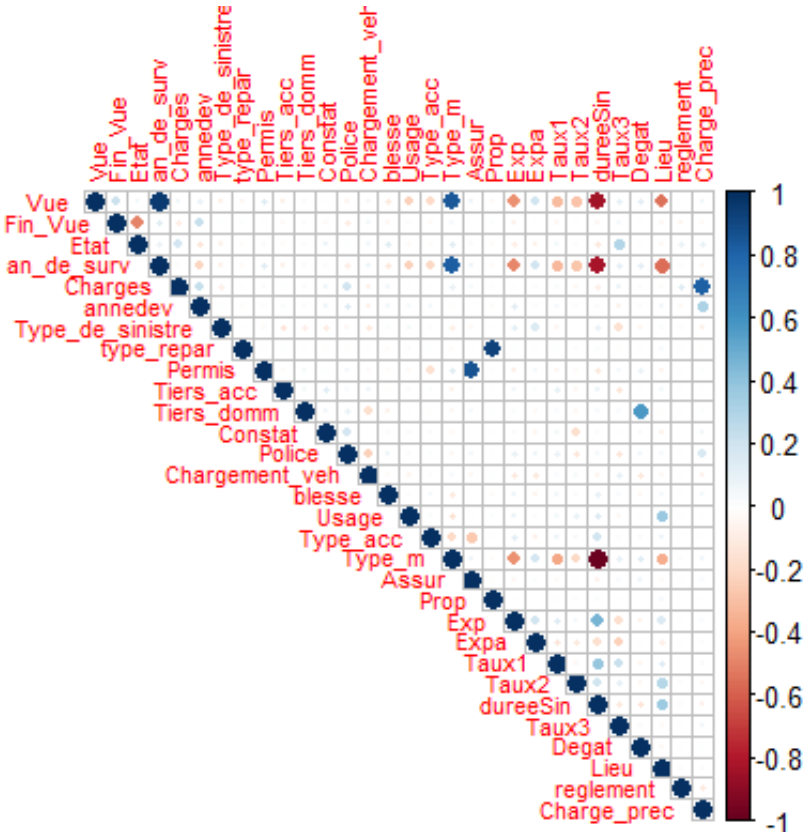


FIGURE 2.3.5 – corrélation entre toutes les variables

Les variables les plus corrélées à la charge demeurent la charge précédente et l'année de développement. D'autres relations peuvent être observées comme :

- entre le type de garanties souscrites ("Prop") et le type de réparation post-sinistre ("$Type\_repar$")
- entre la durée du sinistre ("dureeSin") et : l'année de survenance ("an_de_surv") et le type de garantie("$type\_m$")

2.3.4 Suppression de lignes

Pour certains sinistres, des paiements sont inscrits comme effectués à une date antérieure à leur date de survenance. Aussi il nous a semblé adéquat de supprimer les lignes concernées par cette anomalie.

De plus, dans l'optique de constituer une modélisation année de développement après année de développement, on crée une nouvelle table contenant pour chaque sinistre et pour chaque année de développement le montant de la dernière évaluation du sinistre pour l'année considérée. Cette opération est effectuée à l'aide de la variable *chgtreg* créée précédemment ainsi que d'une procédure SQL. Ce traitement de la base permet de synthétiser de tels sorte que pour un sinistre donné, on ait le montant des charges par année de développement.

Première partie

Méthodes Usuelles de Provisionnement

Chapitre 3

Méthode de Chain Ladder

La méthode Chain Ladder est une méthode déterministe dont le but est de permettre d'estimer sur la base des éléments connus, les éléments futurs. Elle est de loin la plus couramment utilisée par les compagnies d'assurance du fait notamment de sa facilité de mise en œuvre et de sa robustesse. Elle s'applique à des triangles de paiements cumulés ou des triangles de charges. Rappelons qu'on appelle triangles de charges (ou encore triangle de liquidation) un triangle qui agrège les paiements individuels des sinistres avec un axe représentant l'année de l'origine de sinistres, et un autre axe l'année de déroulement de paiements. Plusieurs sources bibliographiques ont été utilisées pour implémenter ces méthodes ([3]).

3.1 Principe

Afin d'estimer un montant de provision par la méthode de Chain Ladder, on suppose que les cadences de paiement observées dans le passé vont être maintenues dans le futur. La méthode de Chain Ladder repose sur deux hypothèses, à savoir :

- H_0 : les années de survenance sont indépendantes
- H_1 : il y a une relation de récurrence entre le futur et le passé

Sous ces hypothèses, on effectue une projection en partant d'un triangle de données exprimées en cumulé. On peut appliquer cette méthode sur des triangles de paiements, de provisions, de charges etc... Il faut cependant que les données soient homogènes. Pour k années de survenance et n années de développement considérons le triangle suivant :

	0	1	2	...	n-1	n
0	$C_{0,0}$	$C_{0,1}$	$C_{0,2}$	...	$C_{0,n-1}$	$C_{0,n}$
1	$C_{1,0}$	$C_{1,1}$	$C_{1,2}$	...	$C_{0,n-1}$	
⋮	⋮	⋮				
k-1	$C_{k-1,0}$	$C_{k-1,1}$				
k	$C_{k,0}$					

TABLE 3.1 – triangle de données

Les lignes du triangle correspondent aux années de survenance des sinistres, les colonnes aux années de développement. Toute la partie supérieure du triangle étant connue, l'objectif est la détermination de toute la partie inférieure du triangle. La dernière colonne représentera alors la charge ultime estimée. En notant :

- $i \in \{1, \ldots, k\}$ pour les années de survenance ,
- $j \in \{1, \ldots, n\}$ les années de développement
- et $C_{i,j}$ les paiements cumulés,

L'hypothèse H_1 se traduit dans ce cas comme suit : $C_{i,j+1} = f_{i,j} * C_{i,j}$ .Les $f_{i,j}$ sont appelés facteur de développement.

La démarche de projection du triangle est la suivante :

1. Calcul des facteurs de développement par année de survenance Soit $f_{i,j}$ le facteur de développement individuel

$$f_{i,j} = \frac{C_{i,j+1}}{C_{i,j}} \tag{3.1}$$

Nous allons alors considérer des coefficients de passage(ou encore cadence), d'une année à l'autre, commun pour les années de survenance, et dont l'estimation est donnée par

$$\hat{f}_j = \frac{\sum_{i=0}^{n-j-1} C_{i,j+1}}{\sum_{i=0}^{n-j-1} C_{i,j}}, j = 0, \ldots, n \tag{3.2}$$

2. Application de cette cadence historique et estimation de la charge ultime

Grâce à ces facteurs, nous pouvons estimer :

- les charges ultimes par exercice de survenance : $\hat{C}_{i,n} = C_{i,n-i} * \prod_{j=n-i}^{n-1} \hat{f}_j$
- les provisions par exercice de survenance : $\hat{R}_i = \hat{C}_{i,n}\hat{C}_{i,n-i}$
- les provisions totales : $\hat{R} = \sum_{i=1}^{n} \hat{R}_i$

Avec une méthode type « Chain Ladder », il faut faire attention aux sinistres graves et atypiques : d'une année à l'autre la cadence estimée ne sera pas représentative de la cadence réelle en tenant compte de ces sinistres. Un autre inconvénient de cette méthode réside dans le fait qu'elle ne fait aucune hypothèse sur la loi suivie par les coûts et les fréquences des sinistres, et donc, comme toutes les méthodes déterministes, elle ne permet pas d'évaluer la précision de l'estimation obtenue.

3.2 Modélisation sous R

Dans cette partie, nous allons mettre en oeuvre le principe de Chain Ladder décrit ci dessus sur notre triangle de données.

3.2.1 Création du triangle de données

Pour construire notre triangle de données, nous avons principalement utilisé des requêtes SQL sur la table des données synthétiques i.e la table dans laquelle une ligne correspond à une année de développement du sinistre. Nous avons procédé comme suit :

- Récupération des différentes années de survenance et du nombre d'années de développement.
- Construction d'une matrice triangulaire de taille m x n et de coefficient $x_{i,j}$ où m désigne le nombre d'années de survenance des sinistres, n le nombre d'années de développement et $x_{i,j}$ le montant total des règlements effectués en l'année de développement j pour les sinistres survenus en la même année i.
- Cumul des règlements par ligne pour obtenir un triangle de charges cumulées

3.2.2 La fonction R

Après avoir construit le triangle de données, nous implémentons sous R une fonction ayant pour variable d'entrée le triangle de données et qui renvoie en sortie un tableau complet dont la partie inférieure correspond à un montant annuel de provision estimé par la méthode déterministe de Chain Ladder.

Cette fonction applique le principe expliqué précédemment i.e, elle :

- calcule et applique les cadences historiques
- calcule la réserve

Le montant de la réserve calculé par cette méthode est de : 301 256 138 €

	X0	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16	X17	X18
1	6508792	8420486	8314125	8114550	8798594	8598065	8446768	8374845	8338310	8602130	8583539	8592015	8591985	8673638	8687270	8895993	8921509	8957780	9033088
2	9914236	10568106	11114585	12245513	12885543	13017155	12702364	12771332	12752050	12551939	12538829	12765870	12811454	12753532	12852769	12855677	13032982	12965235	0
3	11997809	14623614	15763371	14305159	14087038	15653767	14096822	13792002	13894189	14035550	13881882	13747073	13702115	13711264	13756178	13838855	13811463	0	0
4	14822593	19118948	17816603	17483860	17707893	17476787	17672607	17515019	17116666	17106964	17242718	17251329	17089993	17142990	17094352	17117345	0	0	0
5	15785320	16129887	16030834	15807705	15433215	15835715	15664877	15142746	14909110	15352264	15363938	15298390	15850261	15832106	15704377	0	0	0	0
6	13425490	13341582	13411622	15151344	15351159	14748369	15006886	15255199	15200665	15075686	15159455	14995227	14723905	14734523	0	0	0	0	0
7	13126280	15654350	15572576	16563026	16641827	16282658	16228312	16240968	15561712	15501801	15424462	15287572	15343771	0	0	0	0	0	0
8	12622754	12661166	12930367	13987336	13825113	14411244	14254030	14665927	14256193	14035819	14190374	14129893	0	0	0	0	0	0	0
9	16379195	18560241	19942613	20094542	19458986	18921437	18516106	18142496	17905463	17624556	17408430	0	0	0	0	0	0	0	0
10	17996102	18445179	18409338	18307013	18597694	19182699	19699216	19475167	19708647	19627278	0	0	0	0	0	0	0	0	0
11	18249102	20687451	21452463	21766369	22438951	22217816	22378165	22499909	23128189	0	0	0	0	0	0	0	0	0	0
12	19211399	21476222	23730445	23789370	23194503	24602484	23916189	23643291	0	0	0	0	0	0	0	0	0	0	0
13	23861979	25802334	28288269	28708171	28778286	28316082	28674818	0	0	0	0	0	0	0	0	0	0	0	0
14	19541776	28761396	31347823	31994813	30909979	30494636	0	0	0	0	0	0	0	0	0	0	0	0	0
15	22457333	27496075	30234548	31804298	32617737	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	23606886	29384119	33749150	36592696	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	28332325	35762170	37013565	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	33066625	44315060	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	27793509	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FIGURE 3.2.1 – triangle de données

	X0	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16	X17	X18
1	6508792	8420486	8314125	8114550	8798594	8598065	8446768	8374845	8338310	8602130	8583539	8592015	8591985	8673638	8687270	8895993	8921509	8957780	9033088
2	9914236	10568106	11114585	12245513	12885543	13017155	12702364	12771332	12752050	12551939	12538829	12765870	12811454	12753532	12852769	12855677	13032982	12965235	13074234
3	11997809	14623614	15763371	14305159	14087038	15653767	14096822	13792002	13894189	14035550	13881882	13747073	13702115	13711264	13756178	13838855	13811463	13791662	13907608
4	14822593	19118948	17816603	17483860	17707893	17476787	17672607	17515019	17116666	17106964	17242718	17251329	17089993	17142990	17094352	17117345	17201718	17177056	17321463
5	15785320	16129887	16030834	15807705	15433215	15835715	15664877	15142746	14909110	15352264	15363938	15298390	15850261	15832106	15704377	15799490	15877367	15854604	15987893
6	13425490	13341582	13411622	15151344	15351159	14748369	15006886	15255199	15200665	15075686	15159455	14995227	14723905	14734523	14730503	14819717	14892765	14871414	14996437
7	13126280	15654350	15572576	16563026	16641827	16282658	16228312	16240968	15561712	15501801	15424462	15287572	15343771	15358294	15354103	15447095	15523235	15500979	15631296
8	12622754	12661166	12930367	13987336	13825113	14411244	14254030	14665927	14256193	14035819	14190374	14129893	14155286	14168684	14164818	14250607	14320850	14300318	14420541
9	16379195	18560241	19942613	20094542	19458986	18921437	18516106	18142496	17905463	17624556	17408430	17359199	17390396	17406855	17402106	17507501	17593797	17568573	17716272
10	17996102	18445179	18409338	18307013	18597694	19182699	19699216	19475167	19708647	19627278	19613212	19557746	19592894	19611438	19606087	19724831	19822056	19793637	19960042
11	18249102	20687451	21452463	21766369	22438951	22217816	22378165	22499909	23128189	23108249	23091688	23026384	23067766	23089599	23083300	23223103	23337571	23304112	23500030
12	19211399	21476222	23730445	23789370	23194503	24602484	23916189	23643291	23493115	23472859	23456038	23389704	23431738	23453916	23447517	23589526	23705801	23671814	23870823
13	23861979	25802334	28288269	28708171	28778286	28316082	28674818	28521260	28340099	28315665	28295373	28215353	28266060	28292814	28285094	28456402	28596666	28555667	28795734
14	19541776	28761396	31347823	31994813	30909979	30494636	30227668	30065793	29874823	29849066	29827674	29743321	29796774	29824976	29816839	29997423	30145283	30102064	30355132
15	22457333	27496075	30234548	31804298	32617737	32826268	32538887	32364636	32159063	32131337	32108310	32017507	32075047	32105406	32096646	32291038	32450203	32403680	32676097
16	23606886	29384119	33749150	36592696	36668808	36903238	36580165	36384272	36153168	36121997	36096111	35994030	36058717	36092846	36082998	36301533	36480467	36428165	36734416
17	28332325	35762170	37013565	38015037	38094107	38337649	38002018	37798511	37558424	37526043	37499150	37393101	37460302	37495758	37485527	37712557	37898445	37844111	38162266
18	33066625	44315060	46712905	47976811	48076601	48383963	47960381	47703545	47400544	47359676	47325736	47191898	47276709	47321455	47308544	47595066	47829667	47761094	48162621
19	27793509	33016269	34802747	35744401	35818749	36047744	35732161	35540809	35315062	35284615	35259328	35159614	35222801	35256139	35246520	35459988	35634774	35583685	35882836

FIGURE 3.2.2 – triangle complété

La figure 3.2.1 désigne les charges historiques payées par année de développement et pour chaque année de survenance.

En appliquant la fonction Chain Ladder sur ce triangle de données, on obtient la figure 3.2.2 où la partie inférieure correspond aux coûts ultimes des sinistres survenus la même année aux années de développement à venir.

3.2.3 Vérification

Il est possible de remplir notre tableau de données en utilisant le package ChainLadder disponible sur R en procédant comme suit :

- Remplacer les 0 de notre tableau de données par NA afin d'avoir un tableau bien adapté à la fonction du package ChainLadder de R
- Puis appliquer la fonction "round" du package Chain Ladder sur le tableau de données nouvellement construit (avec les NA).

Nous avons utilisé les résultats du package ChainLadder de R afin de tester notre fonction. Aussi, notre fonction donne un résultat semblable à celui du package avec une erreur de 10^{-7} en moyenne quadratique des erreurs.

3.2.4 Validation des hypothèses de Chain Ladder

Dans le graphe ci dessous, nous constatons qu'après quelques années (à partir de l'année à laquelle le montant de la charge est estimée par la méthode de Chain Ladder), le facteur de développement individuel ne varie plus en fonction de l'année de survenance des sinistres, ce qui valide l'hypothèse de Chain Ladder.

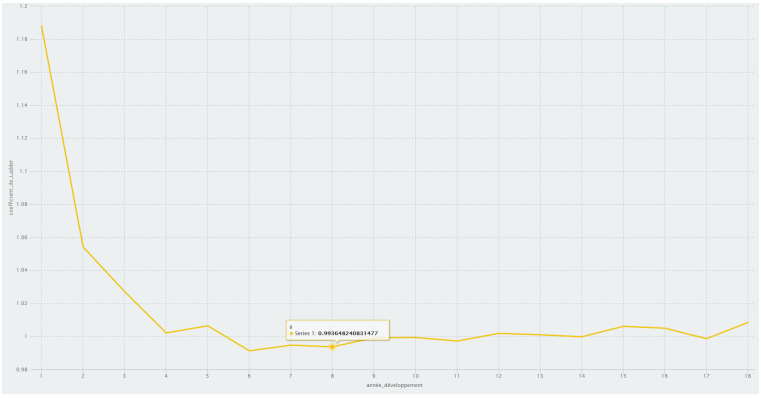


FIGURE 3.2.3 – Représentation des coefficients en fonction des années de développement

Le graphe 10.2.4 nous montre que la charge de l'année n+1 est une fonction linéaire de la charge à l'année n quelque soit l'année de survenance considéré. Ceci satisfait bien l'hypothèse de Chain Ladder car cela signifie que le rapport entre les deux charges d'années consécutives qu'est le coefficient de développement individuel est constant et ne dépend pas de l'année de survenance du sinistre

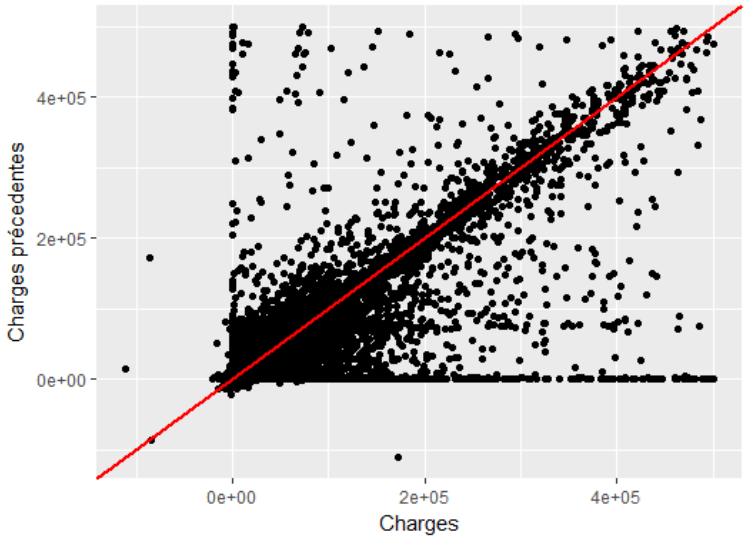


FIGURE 3.2.4 – Représentation de la charge à l'année n en fonction celle à l'année n-1

La figure ci dessus montre que, pour chaque année de survenance, la charge à l'année n+1 est une fonction linéaire de la charge à l'année n quelle que soit l'année de survenance du sinistre, ce qui soutient l'hypothèse de Chain Ladder.

Chapitre 4

Méthode de Mack

Le modèle de Mack est la version stochastique de la méthode de Chain Ladder. Le montant des provisions estimé est identique, mais il a cependant l'avantage d'estimer une erreur de prédiction des provisions. En effet, dans la plupart des situations, la première hypothèse sur laquelle se base Chain Ladder déterministe n'est pas parfaitement vérifiée et cette méthode n'est pas en mesure de quantifier l'erreur de prédiction. Mack a proposé la première approche stochastique permettant de corriger ce défaut. Ainsi, cette approche permettra de mesurer la volatilité des réserves déterministes de Chain Ladder.

4.1 Principe

4.1.1 Hypothèses et estimation

L'idée de Mack est de considérer les lignes de règlements cumulés comme des processus markoviens vérifiant certaines propriétés. En effet, on suppose que les facteurs individuels de développement sont les réalisations d'une variable aléatoire d'espérance inconnue $f_{i,j}$ et estimée par $\hat{f}_j \; \forall j \geqslant 1$ l'estimateur de Chain-Ladder. Ce modèle fonctionne selon les hypothèses suivantes :

- H_0 : pour tout i $\neq$ k, $C_{i,j}$ est indépendant de $C_{k,j}$
- H_1 : il existe des paramètres $f_{i,j}$ tels que

$$\mathbb{E}[C_{i,j+1}|C_{i,1},\ldots,C_{i,j}] = \mathbb{E}[C_{i,j+1}|C_{i,j}] = C_{i,j} \cdot \lambda_{i,j} \tag{4.1}$$

- H_2 : il existe des paramètres $\sigma_{i,j}$ tels que

$$Var[C_{i,j+1}|C_{i,1},\ldots,C_{i,j}] = Var[C_{i,j+1}|C_{i,j}] = C_{i,j} \cdot \sigma_{i,j}^2 \tag{4.2}$$

Les estimateurs $\hat{f}_j$ de Mack coïncident avec les estimateurs de Chain Ladder déterministe. Cependant, on a en plus une estimation de $\sigma_{i,j}$ qui est donnée par la formule suivante :

$$\hat{\sigma}_j^2 = \frac{1}{n-j-1} * \sum_{i=1}^{n-j} C_{i,j} * (\frac{C_{i,j+1}}{C_{i,j}} - \hat{f}_j)^2, j = 1,\ldots,n-2 \tag{4.3}$$

et $\hat{\sigma}_{n-1}^2 = min(\frac{\hat{\sigma}_{n-2}^4}{\hat{\sigma}_{n-3}^2}, min(\hat{\sigma}_{n-3}^2, \hat{\sigma}_{n-2}^2))$

4.1.2 Estimation des erreurs

L'innovation de la méthode de Mack par rapport à Chain Ladder déterministe est la possibilité de quantifier une erreur de prédiction. La provision total R (la provision totale) étant aussi une variable aléatoire dans la méthode de Mack, elle ne peut être obtenue qu'en réalisant une prédiction à partir des données passées. L'incertitude sur cette prédiction peut être quantifiée par la moyenne quadratique des erreurs de prédiction (MSEP) :

$$MSEP(\hat{R}) = \mathbb{E}[(\hat{R} - R)^2 | C_{i,j}, i + j \leq n + 1] \tag{4.4}$$

Sous les hypothèses de Mack on peut obtenir une estimation explicite de $MSEP(\hat{R})$. Cette valeur est aussi appelée variance totale.

$$\widehat{MSEP}(\hat{R}) = \sum_{i=2}^{n} \widehat{MSEP}(\hat{R}_i) + \hat{C}_{i,n}\Big(\sum_{l=i+1}^{n} \hat{C}_{l,n} \sum_{j=n-i+1}^{n-1} \frac{2\hat{\sigma}_j^2}{\hat{\lambda}_j^2 \sum_{k=1}^{n-j} C_{k,j}}\Big) \tag{4.5}$$

avec

$$\widehat{MSEP}(\hat{R}_i) = \hat{C}_{i,n}^2 \sum_{j=n-i-1}^{n-1} \frac{\hat{\sigma}_j^2}{\hat{\lambda}_j^2}\Big(\frac{1}{\hat{C}_{i,j}} + \frac{1}{\sum_{k=1}^{n-j} C_{k,j}}\Big) \tag{4.6}$$

4.2 Modélisation sous R

4.2.1 La fonction R

Sur la base du même triangle de données, nous implémentons sous R une fonction qui tout comme son homologue avec Chain Ladder, a pour variable d'entrée le triangle de données et renvoie en sortie un tableau complet dont la partie inférieure correspond à un montant annuel de provision estimé de la même manière que Chain Ladder. Cependant, la Méthode de Mack ayant une dimension probabiliste, on peut aussi récupérer en sortie de cette fonction les indicateurs des erreurs de prédictions décrits précédemment.

Cette fonction applique donc le principe expliqué précédemment i.e, elle :

- calcule et applique les cadences historiques
- calcule la réserve
- calcule les résidus et les $\widehat{MSEP}(\hat{R}_i)$

Les résultats obtenus avec cette fonction Mack sont les même que ceux de Chain Ladder car l'estimateur des facteurs de développement $\hat{f}_i$ est le même.

4.2.2 Estimations des erreurs

A partir de la fonction explicitée ci-dessus il est possible de retrouver les résidus et l'incertitude au niveau de l'estimation de la provision.

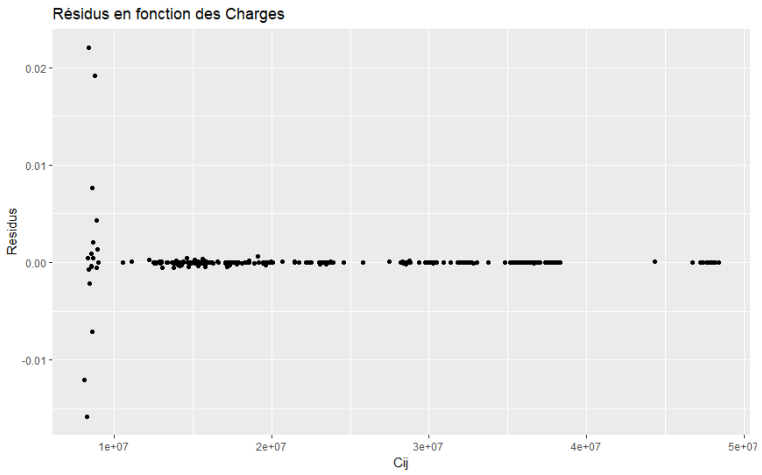


FIGURE 4.2.1 – Résidus en fonction des charges

Dans notre triangle, les résidus sont constants pour toutes les charges sauf pour les petites charges.

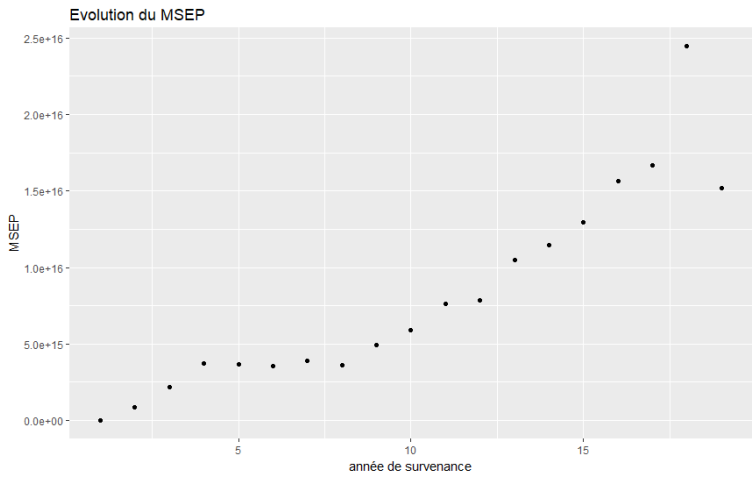


FIGURE 4.2.2 – Evolution du MSEP

Cette représentation graphique permet de constater une tendance à la hausse pour l'évolution de l'incertitude de l'estimation de la provision en fonction de l'année de survenance. Cette incertitude augmente avec les années de survenance, ce qui est cohérent puisqu'on moins d'information sur l'évolution des charges pour les années de survenance récente.

4.2.3 Vérification

Les résultats du package ChainLadder de R permettent aussi de tester les résultats de notre modèle. L'estimation du triangle par la méthode de Mack restant identique à celle de Chain Ladder, les différences avec l'utilisation du package restent les mêmes.

4.2.4 Vers une modélisation ligne à ligne

Les méthodes les plus traditionnelles et les plus souvent utilisées en pratique sont les méthodes basées sur le triangle de liquidation comme Chain Ladder /Mack. Cette forme de représentation a pour avantage d'être très simple, parlante et concise. Néanmoins, ses points faibles sont multiples : elle ne présente pas toutes les informations disponibles, elle contient un nombre d'observations souvent insuffisants pour que les méthodes statistiques puissent fonctionner avec une grande précision. Aussi, il est important de pouvoir challenger ces méthodes usuelles de provisionnemment avec des modèles lignes à lignes qui pourraient être plus précis.

Deuxième partie

Modélisation ligne à ligne

Objectifs

Nous disposons d'une base de données détaillées composée de plusieurs sinistres. Pour chaque sinistre, nous connaissons son année de survenance, son état et son historique de règlement. Effectuer une modélisation ligne à ligne consistera pour chaque sinistre à expliquer la charge en fonction des différentes variables explicatives. Cette thématique récente fait l'objet de recherche ([4],[5]).

Techniques de modélisation

Il existe plusieurs méthodes qui permettent de modéliser notre problème. Étant donné qu'on cherche à expliquer la charge en fonction de variables explicatives, une régression semble être adaptée. Aussi il existe plusieurs méthodes d'apprentissage statistiques qui permettraient de modéliser le problème. Dans la suite de ce rapport, nous utiliserons plusieurs types de modélisation en apprentissage supervisé tels que les GLM, les forêts aléatoires ou encore les Support Vector Machine (SVM), ainsi que plusieurs manières de modéliser le problème.

Hypothèses de modélisation

Tout au long de notre modélisation, nous avons émis les hypothèses suivantes :

- H_1 : Un sinistre Définitivement Clos [1] ne peut être ré-ouvert
- H_2 : La charge à l'ultime dépends de certains paramètres propres au sinistre
- H_3 : on ne considère que les évolutions du sinistres pour lesquelles la charge est strictement supérieure à zéro

De plus, pour toutes les méthodes de modélisation, on pourra définir des protocoles de test et des critères de performance. Ces critères permettront de mesurer la qualité de la modélisation. On pourra donc utiliser des bases de test et d'apprentissage afin de réaliser des validations croisées en utilisant des mesures telles que l'erreur quadratique ou encore le RMSE. On pourra aussi définir des procédures permettant de quantifier l'erreur réalisée sur notre modélisation.

Mesures de l'erreur

Afin de choisir le meilleur modèle de prédiction du nombre d'années de développement, nous avons réalisé une étude comparative des différents modèles ajustés en utilisant les indicateurs de performance suivants, avec y_i la vrai charge et $\hat{y}_i$ la charge estimée par le modèle.

RMSE La Root Mean squar error (RMSE) est la racine carrée de la moyenne arithmétique des carrés des écarts entre les prédictions et les observations. Elle se calcule par la formule suivante :

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{4.7}$$

MAE La Mean absolute error (MAE) est la moyenne arithmétique des valeurs absolues des écarts entre les valeurs observées et les valeurs prédites.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{4.8}$$

RMSLE Le RMSLE (Root Mean Squared Logarithmic Error) mesure le rapport en logarithme entre la valeur réelle et celle prédite.

$$RMSLE = \sum_{i=1}^{n}\log\left(\frac{1+\hat{y}_i}{1+y_i}\right)^2 \tag{4.9}$$

1. modalité "Def Clos" de la variable Etat

Chapitre 5

Modèles linéaires généralisés(GLM)

5.1 Cas général

Les modèles linéaires généralisés sont des modèles qui permettent d'étudier la relation entre une variable dépendante, (ou alors variable réponse) Y et un ensemble de variables explicatives. Un modèle linéaire généralisé est formé de trois composantes :

- **la variable réponse** Y, composante aléatoire à laquelle est associée une loi de probabilité,
- **les variables explicatives** $X_1 \ldots X_k$ définissent sous forme d'une combinaison linéaire la composante déterministe. Elles peuvent être quantitatives ou qualitatives,
- **une fonction lien** h qui décrit la relation fonctionnelle entre la combinaison linéaire des variables $X_1 \ldots X_k$ et l'espérance mathématique de la variable de réponse Y.

le tableau 5.1 donne des exemples de fonction lien :

Loi	**Nom du lien**	**Exemple de fonction de lien**
Bernoulli/Binomiale	lien logit	$g(\mu) = logit(\mu) = log(\frac{\mu}{1-\mu})$
Poisson	lien log	$g(\mu) = log(\mu)$
Normale	lien identité	$g(\mu) = \mu$
Gamma	lien réciproque	$g(\mu) = \frac{1}{\mu}$

TABLE 5.1 – Exemple de fonction lien

Notons Y = (Y_1,...,Y_n) un échantillon aléatoire de taille n de la variable de réponse Y, les variables aléatoires (Y_1,...,Y_n) sont supposées indépendantes. Soit la matrice X telle que

$$X = \begin{bmatrix} X_{1,1} & \ldots & X_{k,1} \\ \vdots & \ddots & \vdots \\ X_{n,1} & \ldots & X_{k,n} \end{bmatrix} \tag{5.1}$$

Le modèle GLM s'écrit comme suit :

$$h(\mathbb{E}[Y|X]) = X \cdot \beta + \epsilon \tag{5.2}$$

avec $\beta = (\beta_1, \ldots, \beta_k)'$ et $\epsilon = (\epsilon_1, \ldots, \epsilon_k)'$
Le paramètre ϵ correspond à la composante aléatoire intrinsèque au modèle. La loi du paramètre ϵ définit la loi de la variable réponse et donc le type de modèle. Par exemple, lorsque ϵ suit une loi normale centrée réduite, on parle de modèle linéaire gaussien. Les β_i sont les paramètres à estimer.

5.1.1 Variables explicatives quantitatives

Lorsque les $X_{i,1} \ldots X_{i,k}$ sont quantitatifs, on obtient un modèle mathématique de la forme :

$$h(\mathbb{E}[Y_i|X]) = \beta_0 + \beta_1 \cdot X_{i,1} + \cdots + \beta_k \cdot X_{i,k} + \epsilon_i \tag{5.3}$$

5.1.2 Variables explicatives qualitatives

Dans le cas où la variable explicative est une variable qualitative alors on peut décrire le modèle en fonction des différentes modalités prises par les différentes variables.
soient X_1 et X_2 deux variables qualitatives, telles que :

— X_1 peut prendre les l modalités : a_1, a_2,..., a_l

— X_2 peut prendre les m modalités : b_1, b_2,..., b_m

On obtient alors un modèle mathématique de la forme :

$$h(\mathbb{E}[Y_i|X]) = \beta_0 + \sum_{j=1}^{l} \alpha_j \cdot 1_{[X_{i,1}=a_j]} + \sum_{j=1}^{m} \beta_j \cdot 1_{[X_{i,2}=b_j]} + \epsilon \tag{5.4}$$

Dans cette modélisation, les β_i et les α_i sont les paramètres que l'on a besoin d'estimer. Ainsi, plus les variables explicatives ont de modalités distinctes plus le modèle complet qui en découle est complexe.

5.1.3 Estimation et prédiction

Notons (Y_1, ...,Y_n) un échantillon aléatoire de taille n de la variable de réponse Y, les variables aléatoires(Y_1, ..., Y_n) étant supposées indépendantes pour tous les individus $i \in \{ 1, \ldots, n\}$, on peut expliquer la variable réponse Y_i par un GLM.

$$h(\mathbb{E}[Y_i|X]) = \beta_0 + \beta_1 \cdot X_{1,i} + \cdots + \beta_k \cdot X_{k,i} + \epsilon \tag{5.5}$$

L'information disponible grâce à tous les individus permet par le biais de l'estimateur des moindres carrés ordinaires (MCO) d'estimer les paramètres β_i. Aussi, dès lors qu'on a une estimation $\hat{\beta}_i$, on peut pour des variables explicatives $\tilde{X_1}$ ...$\tilde{X_k}$ connues, prédire l'espérance de la variable réponse $\hat{Y}$ associée. telles que :

$$\widehat{\mathbb{E}[Y|X]} = h^{-1}(\hat{\beta}_0 + \hat{\beta}_1 \cdot \tilde{X_1} + \cdots + \hat{\beta}_k \cdot \tilde{X_k}) \tag{5.6}$$

5.1.4 Erreur de prédiction et intervalles de confiances

5.2 Modélisation GLM du problème

Dans le cadre de notre étude, on cherche à prédire le montant des charges à l'ultime. La variable réponse correspond donc à une variable aléatoire à valeur positive. Ainsi il n'est pas dénué de sens de considérer les $Y_i \in \mathbb{R}_+$. De plus, dans notre cas, les variables explicatives peuvent être quantitatives et qualitatives.

Nous avons retenu comme types de modélisation une regression **log-normal** : la fonction lien est la fonction logarithme et les ϵ_i suivent une loi gaussienne standard. Ainsi $log(\mathbb{E}[Y|X])$ suit une loi $N(\beta_0 + \beta_1 \cdot X_{i,1} + \cdots + \beta_k \cdot X_{i,k}, \sigma^2)$, et :

$$\mathbb{E}[Y|X] = e^{\beta_0+\beta_1 \cdot X_{i,1}+\cdots+\beta_k \cdot X_{i,k}+\epsilon_i} \tag{5.7}$$

Dans notre problème, on considère aussi bien des variables explicatives positives que négatives. Notre modélisation peut être décrite comme suit :

$$\log(\mathbb{E}[Y|X]) = \beta_0 + \sum_{i=1}^{k} \beta_k \cdot X_k + \sum_{i=k+1}^{l} \alpha_i \cdot 1_{[X_i=a_i]} + \cdots + \sum_{i=c>l}^{m} \beta_i \cdot 1_{[X_i=b_i]} + \epsilon \tag{5.8}$$

5.2.1 Première modélisation simple

Une première manière de modéliser le problème consiste à expliquer la charge à l'ultime du sinistre, uniquement à partir de ses variables explicatives retenues. La base de données est constituée de toutes les évolutions par année et cela pour tous les sinistres. C'est un principe de modélisation que l'on conservera pour tous les algorithmes utilisés.

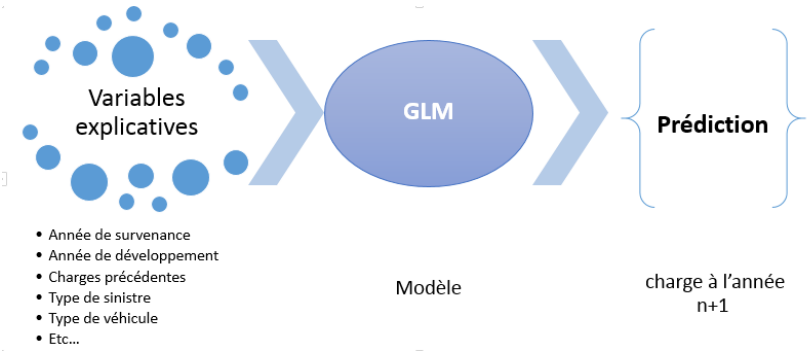


FIGURE 5.2.1 – principe de modélisation

On sépare notre base de données en 2 bases. Une base de test composée de 80% des données et une base de test constituée de 20% des données. Tout le travail de modélisation se fera sur la base d'apprentissage. On pourra plus tard changer la base d'apprentissage pour vérifier la sensibilité du modèle à la base sélectionnée.

Les variables retenues pour notre étude sont les suivantes :

```
$x
 [1] "Type_de_sinistre" "Expa"          "dureeSin"     "Taux3"        "Type_acc"     "Permis"
 [7] "Usage"            "Tiers_acc"     "Tiers_domm"   "Type_m"       "Police"       "Chargement_veh"
[13] "blesse"           "Vue"           "Fin_Vue"      "Date_de_surv" "an_de_surv"   "annedev"
[19] "Charge_prec"
```

FIGURE 5.2.2 – variables retenues pour le glm

En effet, elles avaient été retenues après l'étude des variables explicatives de la bases de données. De plus, même en considérant toutes les variables explicatives, la fonction h2o.glm les supprime automatiquement, car elles sont constantes et la plupart du temps et donc jugés mauvaises.

Sélection de variable

Le modèle complet i.e celui avec toutes nos variables d'études, n'est pas forcement le meilleur modèle permettant de modéliser le modèle. Une sélection de variables permet de rendre le modèle moins complexe, sans perdre une grande quantité d'information. En effet, il existe des variables explicatives dont la p_value est très grande ($>> 0.05$). Ces variables seraient donc moins significatives au vu du test de nullité de leurs coefficients. Il peut donc nous être utile de pouvoir en retirer sans perdre beaucoup d'informations sur notre modèle. Les variables pour lesquelles la p_value est grande sont les suivantes :

	names	p_value
4	Type_de_sinistre.Bdg	0.68701173
6	Type_de_sinistre.Vol	0.93357109
16	Type_acc.Famille2	0.56964197
23	Tiers_acc.OUI	0.54883182
24	Tiers_domm.OUI	0.06168323

FIGURE 5.2.3 – variables non significatives

La méthode de sélection que nous avons choisi d'utiliser est la sélection BIC (Critère d'information Bayésien). Cette sélection consiste à minimiser le BIC qui se calcule comme suit :

$$BIC(Modele) = -2\ln(LL) + p \cdot \ln(n) \tag{5.9}$$

avec LL la vraisemblance du modèle [1] et n le nombre d'observations dans l'échantillon et p le nombre de paramètres du modèle. Le modèle qui sera sélectionné est celui qui minimise le critère

1. La fonction de vraisemblance, notée $L(x_1, \ldots, x_n|\theta_1, \ldots, \theta_k)$ est une fonction de probabilités conditionnelles qui décrit les valeurs x_i d'une loi statistique en fonction des paramètres θ_j supposés connus. Elle s'exprime à partir de la fonction de densité $f(x|\theta)$

BIC, soit :

$$Modele_{BIC} = \arg \min_{Modele} BIC(Modele) \tag{5.10}$$

La méthode de sélection utilisée est la méthode Backward (méthode descendante). Il s'agit d'une méthode de sélection itérative qui à la première étape, toutes les variables sont intégrées au modèle. Nous retirons la variable explicative dont le retrait du modèle conduit à l'augmentation la plus grande du critère de choix. Nous nous arrêtons lorsque toutes les variables sont retirées ou lorsque qu'aucune variable ne permet l'augmentation du critère de choix.

variable retirée	BIC	Effet
Aucune	2498891	
Type de sinistre	2500139	pas d'amélioration
Type_acc	2498891	pas d'amélioration
Tiers_acc	2498880	amélioration
Tiers_acc + Type de sinistre	2500218	pas d'amélioration
Tiers_acc + Type_acc	2498890	pas d'amélioration
Tiers_acc+Tiers_domm	2498876	amélioration
Tiers_acc+Tiers_domm +Type de sinistre	2500207	pas d'amélioration
Tiers_acc+Tiers_domm + Type_acc	2498887	pas d'amélioration

TABLE 5.2 – Selection Backward BIC

Au vu de cette selection backward BIC, on retirerait juste des variables explicatives les variables "Tiers_domm" et "Tiers_acc".
Afin de valider cette sélection de variables on réalise un test de deviance entre 2 modèles emboîtés. Cela correspond à tester si pour deux modèle M_1 et M_2 tel que $M_1 \in M_2$ si :

$$D = -2(l_n(M_1) - l_n(M_2)) \xrightarrow{L} \chi^2_{p_2-p_1} \tag{5.11}$$

p_j désigne le nombre de paramètres du modèle M_j et $L_n(M_j)$ la vraisemblance du modèle M_j. D notre statistique de test est ici égale à 8.156994.

$$p\_value = \mathbb{P}(D > q_{\chi^2}) = 0.004289542 \tag{5.12}$$

avec q_{χ^2} le quantile du χ^2 à $p_2 - p_1 = 2$ degré de liberté.
Cette p_value est inférieur à 0.05 donc on peut garder le résultat de la sélection backward.

```
 [1] "Type_de_sinistre" "Expa"            "Taux3"           "Type_acc"
 [5] "Usage"            "Permis"          "blesse"          "Type_m"
 [9] "Police"           "Chargement_veh"  "Vue"             "Fin_Vue"
[13] "Date_de_surv"     "an_de_surv"      "annedev"         "Charge_prec"
```

FIGURE 5.2.4 – variables finales retenues

Cette procédure de sélection nous a permis de parvenir à 17 variables explicatives au lieu de 19 dans le modèle complet. D'autres méthodes permettent de sélectionner les variables comme l'AIC (critère d'information d'Akaiké) ou encore le R^2ajusté.

Qualité de prédiction

Afin de mesurer la qualité du modèle réalisé, on utilise une validation croisée. En effet Sur notre base d'apprentissage et sur notre base de test on va essayer de re-prédire les charges connus grâce au modèle construit. La mesure de l'ajustement utilisée est le RMSE. Les résultats obtenues sont les suivants :

RMSE	R^2	R^2_{ajuste}	AIC	RMSLE
24844	0.47674	0.47668	2498719	1.904

TABLE 5.3 – performances du modèle

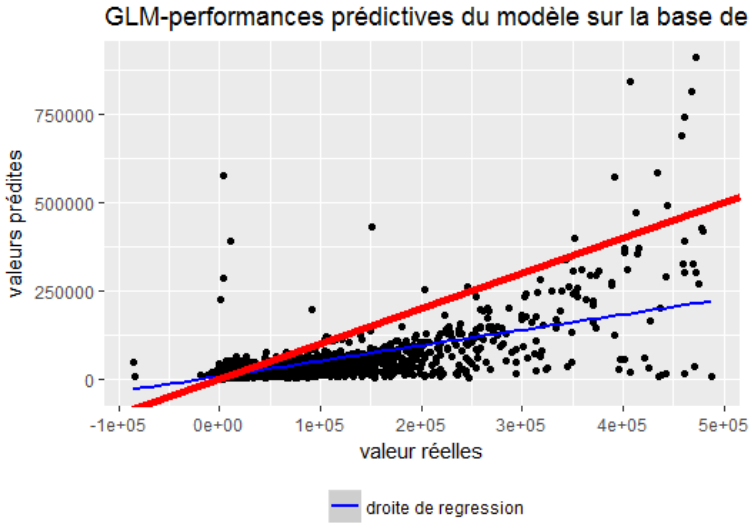


FIGURE 5.2.5 – Comparaison entre prédiction et valeurs réelles d'un GLM log-normal

Cette représentation graphique met en avant une mauvaise corrélation entre la charge prédite par ce modèle et les valeurs réelles avec une droite de régression assez éloignée de la première bissectrice, on peut affirmer que l'ajustement de ce modèle ne semble pas très bon et a tendance à beaucoup sous-estimer les montants de charges.

Validation du modèle

Afin de valider le modèle, on effectue une analyse des résidus. En effet, on veut vérifier les hypothèses liées au modèles telles que la normalité des résidus, leurs homoscédasticité et leur caractère centré. En effet, rappelons que $log(\mathbb{E}[Y|X])$ suit une loi $N(\beta_0 + \beta_1 \cdot X_{i,1} + \cdots + \beta_k \cdot X_{i,k}, \sigma^2)$, d'où le fait que :

$$\epsilon_i = [log(\mathbb{E}[Y|X]) - \beta_0 + \beta_1 \cdot \hat{X}_{i,1} + \cdots + \beta_k \cdot \hat{X}_{i,k}] \; suit \; une \; loi \; N(0, \sigma^2) \tag{5.13}$$

Cette expression s'écrit aussi :

$$\epsilon_i = [log(\mathbb{E}[Y|X]) - log(\hat{Y})] \tag{5.14}$$

car $\hat{Y} = e^{\beta_0 + \beta_1 \cdot \hat{X}_{i,1} + \cdots + \beta_k \cdot \hat{X}_{i,k}}$.

Pour tester ces hypothèses, nous avons calculé les résidus standards ϵ_i et les résidus de pearson rp_i comme suit :

$$\epsilon_i = log(y_i) - log(\hat{y}_i) \; et \; rp_i = \frac{\epsilon_i}{\sqrt{var(\hat{y}_i)}} \tag{5.15}$$

hypothèse de résidus centrés : Afin de vérifier cette hypothèse, on peut tracer les différents résidus afin de constater leurs répartitions.

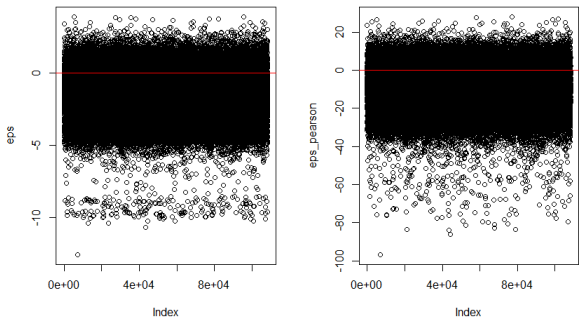


FIGURE 5.2.6 – résidus standards et résidus de pearson

Les résidus standards ou de pearson ne sont pas centrés en zéro comme le montre le graphique précédent.

hypothèse de normalité : Afin de vérifier cette hypothèse, on peut tracer le qqplot des résidus par rapport aux résidus gaussiens.

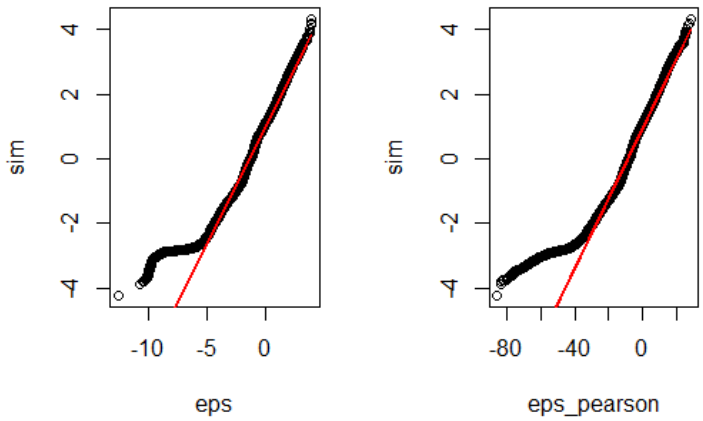


FIGURE 5.2.7 – QQPLOT des résidus standards et résidus de pearson

Les quantiles des résidus et les quantiles gaussiens semblent être liés par une relation linéaire. Cependant pour les petits quantiles, cette relation n'est pas vérifiée.

hypothèse d'homoscédasticité On parle d'homoscédasticité lorsque la variance des résidus de la régression est la même pour chaque observation. Cette hypothèse peut se vérifier graphiquement en traçant la variance des résidus en fonction des observations.

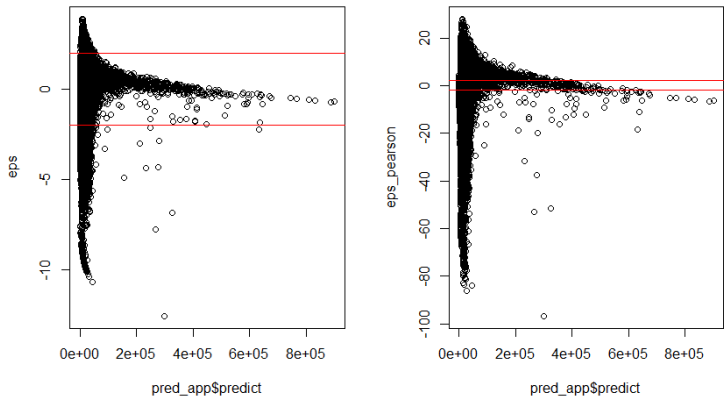


FIGURE 5.2.8 – résidus standards et résidus de pearson en fonction des prédictions

l'hypothèse d'homoscédasticité n'est clairement pas vérifiée. En effet, une structure se dégage de ce graphique, les résidus sont plus importants pour les premières prédictions.
Le fait que toutes les hypothèses ne soient pas vérifiées peut être un élément de réponse quant aux mauvaises prédictions de ce modèle.

Des améliorations au modèle

Une amélioration possible à cette modélisation serait de considérer éventuellement un modèle différent par année de développement.

5.2.2 Modèle évolutif

Principe de modélisation

Le modèle que l'on cherche à calibrer ici prend en compte l'évolution de la charge du sinistre.

On calibre un modèle par année de développement. On a donc 18 modèles [2] qui permettent chacun de prédire la charge à l'année N+1 en fonction des caractéristiques connues de l'année comme la charge connue à l'année N. Pour se faire, on sépare la base de données afin d'en créer 18. Chacune de ces bases correspond à l'évolution d'un sinistre pour l'année de développement considérée. Sur chacune de ces bases, on calibre un GLM simple comme décrit précédemment [3].

Le fait de séparer la base de cette manière, limite énormément les observations disponibles pour calibrer le modèle. Ainsi malgré une modélisation réalisée, il n'est pas possible de réaliser des prédictions de l'évolution avec cette modélisation.

2. nombre d'années de développement maximum présent dans la base
3. avec une sélection de variables puis test de Fisher des modèles emboîtés pour ne retenir que le meilleur

Chapitre 6

Arbres de décision et forets aléatoires

6.1 Arbres de décision

Les arbres de décision sont des algorithmes de prédictions qui fonctionnent en régression et en classification. Ils permettent de trouver une partition qui sépare au mieux les différentes observations. Une fois segmenté, il crée un ensemble de règles (séquences de décision uniques par groupe) en vue de la prédiction d'un résultat ou d'une classe.
En théorie des graphes, un arbre est un graphe non orienté, acyclique et connexe. L'ensemble des noeuds se divise en trois catégories :

- **Nœud racine** : l'accès à l'arbre se fait par ce nœud
- **Nœuds internes** : les noeuds qui ont des descendants
- **Nœuds terminaux** (ou feuilles) : nœuds qui n'ont pas de descendants.

Chaque individu, qui doit être attribué à une classe, est décrit par un ensemble de variables qui sont testées dans les noeuds de l'arbre. Les tests s'effectuent dans les nœuds internes, et les décisions sont prises dans les nœuds feuilles.

6.1.1 Apprentissage avec les arbres de décision

Considérons tout d'abord le problème de classification. Chaque élément x de la base de données est représenté par un vecteur multidimensionnel $(x_1, x_2, \dots x_n)$ correspondant à l'ensemble de variables descriptives du point comme dans le modèle GLM. Chaque noeud interne de l'arbre correspond à un test fait sur une des variables x_i :

- **Variable catégorielle** : génère une branche (descendant) par valeur de l'attribut
- **Variable numérique** : test par intervalles (tranches) de valeurs.

On obtient ainsi que les feuilles de l'arbre spécifient les classes et encodent la règle de décision. Une fois l'arbre construit, classer un nouvel individu se fait par une descente dans l'arbre, de la racine vers une des feuilles. A chaque niveau de la descente on passe un noeud intermédiaire ou une variable est testée pour décider du chemin à choisir pour continuer la descente.

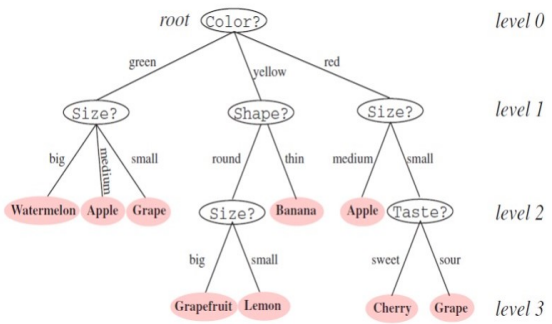


FIGURE 6.1.1 – Exemple d'arbre de décision

Phase 1 : Construction

Au départ, les individus de la base d'apprentissage sont tous placés dans le noeud racine. Chaque noeud est coupé [1] donnant naissance à plusieurs noeuds descendants. Un élément de la base d'apprentissage situé dans un noeud se retrouvera dans un seul de ses descendants. Il existe plusieurs manières de faire des découpes. Il faut donc pouvoir mesurer la qualité de la découpe effectuée.

L'arbre est construit par partition successive de chaque noeud en fonction de la valeur de l'attribut testé à chaque itération. Le critère optimisé est l'homogénéité des descendants par rapport à la variable cible. La variable qui est testée dans un noeud sera celle qui maximise cette homogénéité.

Le processus s'arrête quand les éléments d'un noeud ont la même valeur pour la variable cible (homogénéité).

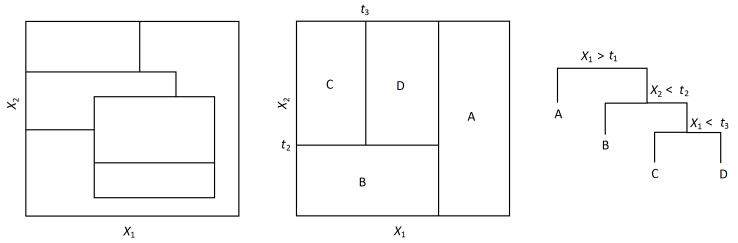


FIGURE 6.1.2 – Construction par partition récursive de l'espace

On effectue des tests successifs sur les $x_1, x_2 \dots x_n$. Chaque noeud feuille est homogène. Cela signifie que ses éléments (points dans chaque région) ont la même valeur pour l'attribut cible. A chaque étape le but est de couper le noeud en deux régions les plus homogènes possibles. Il existe plusieurs types de séparation ou coupes possibles de l'espace des solutions. On peut faire :

- une séparation par partition de variables (figure 6.1.3 gauche) qui donne lieu à des découpes orthogonales.
- une séparation par combinaison linéaire de plusieurs variables (figure 6.1.3 droite) qui donne lieu à des découpes obliques.

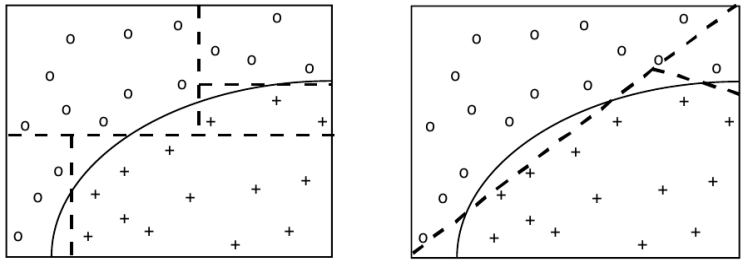


FIGURE 6.1.3 – Séparation de classes par partition itérative des variables.

Phase 2 : Élagage

Dans cette seconde étape, l'objectif est d'améliorer la qualité du modèle d'apprentissage en agissant sur les paramètres de l'arbre décisionnel. En effet, la performance n'augmente pas forcément avec la profondeur de l'arbre.
Après ajustement du modèle, on pourra supprimer les branches peu représentatives pour garder de bonnes performances prédictives. Cependant, nous allons avoir besoin d'un critère pour désigner les branches à élaguer.

1. opération split ou découpe

6.1.2 Modélisation sous R

Construction de l'arbre de régression avec rpart

La librairie rpart, propose les techniques CART avec des algorithmes analogues à ceux développés dans Splus, avec cependant moins de fonctionnalités ; elle fournit des graphes plus explicites, des options plus détaillées et une procédure d'élagage plus performante. Cette fonction intègre une procédure de validation croisée pour évaluer le paramètre de pénalisation de la complexité noté *cp*. Un faible coefficient de pénalisation de la complexité de l'arbre favorise un arbre très détaillé c'est-à-dire avec et donc un nombre important de feuilles.

Optimisation du choix des paramètres du modèle

Pour ne pas subir les paramètres par défaut et améliorer la performance du modèle, on peut déterminer le coefficient de pénalisation ainsi qu'un nombre de split qui minimise l'erreur.

Choix de la complexité cp : Une représentation graphique de l'erreur relative en fonction du coefficient de pénalisation nous permet d'optimiser le choix du paramètre cp. Ce paramètre qui permet d'économiser du temps de calcul en supprimant les divisions qui ne sont évidemment pas valables.

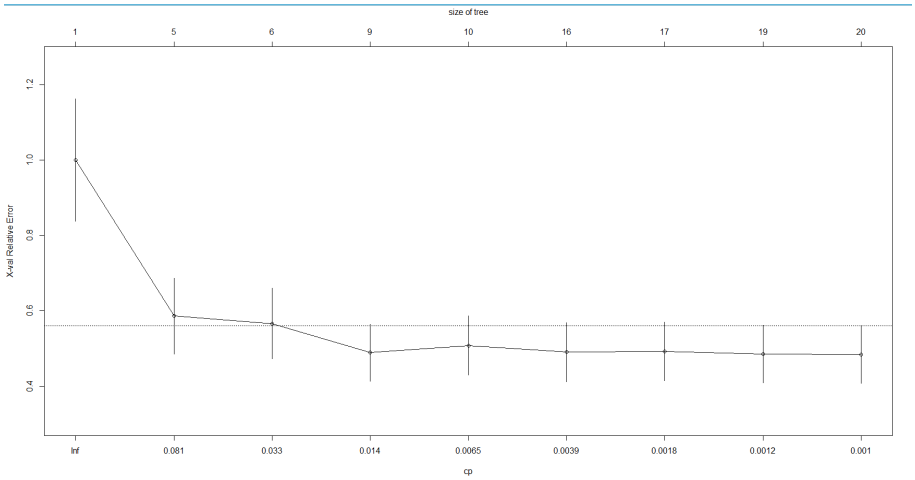


FIGURE 6.1.4 – évolution de l'erreur en fonction du cp

on choisit ici un paramètre de complexité $cp = 0.0012$

Choix du split(découpe) : A partir d'un même type de représentation graphique de l'erreur relative nous permet d'optimiser le choix du split.

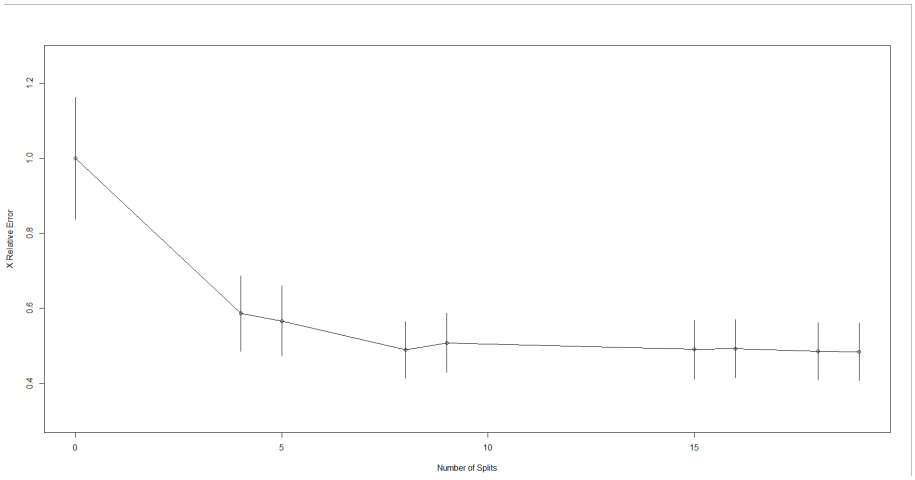


FIGURE 6.1.5 – évolution de l'erreur en fonction du split

on choisit ici un nombre de split = 20

Ajustement du modèle d'arbre de régression avec les paramètres choisis : Après avoir optimisé le choix des paramètres, on ajuste un nouveau modelé d'arbre de régression en intégrant les paramètres de contrôle : cp=0.0012 et minsplit=20

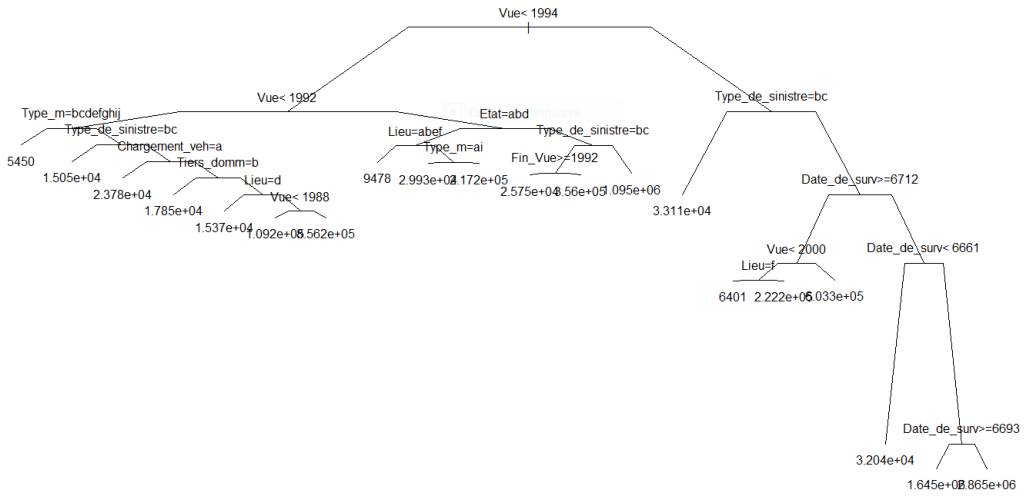


FIGURE 6.1.6 – Arbre décision obtenu

6.2 Forets aléatoires (Random Forest)

Les algorithmes de Forêt aléatoire pour la régression (et la classication) sont des alogrithmes qui utilisent des stratégies adaptatives (boosting) ou aléatoires (bagging). L'idée principale de cet algorithme est d'utiliser une agrégation d'un grand nombre de modèles tout en évitant le sur-apprentissage.

6.2.1 Bagging

Soit Y la variable à expliquer, $X_1, \dots X_p$ les variables explicatives et ϕ le modèle appris sur un échantillon $z = \{(x_1, y_1)...(x_n, y_n)\}$. Le bagging consiste en les étapes suivantes :

1. On considère B échantillons bootstrap $z_1, \dots z_B$ d'individus. Ces échantillons sont issus de z par tirage aléatoire avec remise.
2. Sur chacun des échantillons, on apprend un modèle $\phi(z_i)$.
3. On prédit Y en agrégeant les différentes décisions sur chacun des z_i par

$$Y = \hat{\phi}(x) = \frac{1}{B} \sum_{i=1}^{n} \phi_{z_i}(x) \tag{6.1}$$

Notons que cette manière d'agréger les résultats concerne exclusivement la régression. Pour une classification on prend la décision majoritaire.

Lien avec les forêts aléatoires

Les Forêts Aléatoires vont permettre l'amélioration du Bagging dans le cas spécifique de l'algorithme CART. L'objectif est de rendre les modèles (arbres) construits plus indépendants entre eux. Cette indépendance va permettre de rendre l'agrégation plus efficace.
Dans ce contexte, les B échantillons bootstrap représentent le nombre d'arbres formés dans la forêt. Le tirage aléatoire des variables explicatives à chaque noeud aboutit à des arbres non corrélés. Pour la construction de chaque noeud de chaque arbre, on tire uniformément q variables [2] parmi les p variables pour former la décision associée au noeud. En fin d'algorithme, on possède B arbres que l'on moyenne pour la régression.

6.2.2 Modélisation du problème

A l'aide du logiciel R et du package randomForest, nous mettons en oeuvre le méthode des forêts aléatoires sur notre base de données. Cet algorithme nous sera utile dans la prédiction de l'évolution de la charge des sinistres, mais aussi pour le modèle de durée de vie résiduelle d'un sinistre. Concernant la charge des sinistres on a les performances suivantes :

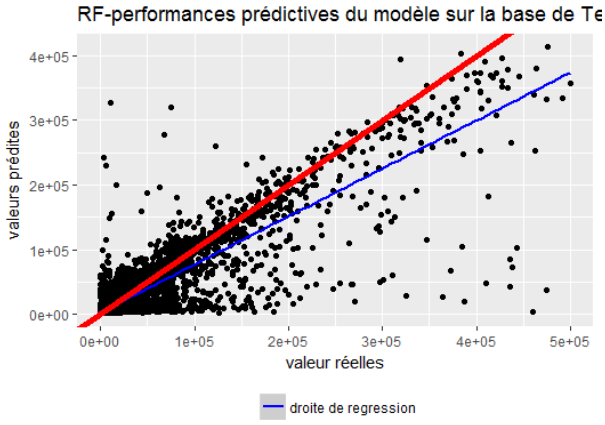


FIGURE 6.2.1 – Comparaison entre prédiction et valeurs réelles d'un random forest

2. En général, un choix optimal pour q est $q = \sqrt{p}$

Cette représentation graphique met en avant une bonne corrélation entre la charge prédite par ce modèle et les valeurs réelles avec une droite de régression assez proche de la première bissectrice. L'ajustement de ce modèle est donc plutôt bon mais il a une tendance à sous-estimer les montants de charges.

6.2.3 Modèle complexe avec intervalles de prédictions

Un concept utile pour quantifier l'erreur liée à la prédiction est celui des intervalles de prédiction. Un intervalle de prédiction est une estimation d'un intervalle dans lequel les futures observations vont tomber avec une probabilité donnée. Il peut donc évaluer l'incertitude liée à la prédiction. Contrairement aux intervalles de confiance, les intervalles de prédiction concernent les prédictions individuelles. Cette thématique a fait l'objet de recherche([6] et [7]).

Afin de construire ces intervalles de confiances et d'estimer une variance pour la prédiction, il est nécessaire d'avoir plusieurs prédictions différentes afférentes à un même modèle. Cependant, avec les packages implémentés sous R, il n'est pas possible de récupérer les différentes prédictions des sous arbres constituant le random forest. Une manière de procéder consiste à séparer la base d'apprentissage en plusieurs bases d'apprentissage sur lesquelles on calibre différents modèles. Chaque modèle engendrera une prédiction différente et sur la base de ces différentes prédictions on peut estimer une variance. Avec la variance et la moyenne, on peut appliquer le théorème central limite qui nous permet d'avoir un intervalle de confiance défini comme suit :

$$IC_{1-} = [\bar{X}_n - t_{n-1,\alpha-\frac{1}{2}} \frac{S_n}{\sqrt{n}} \; ; \; \bar{X}_n + t_{n-1,\alpha-\frac{1}{2}} \frac{S_n}{\sqrt{n}}] \tag{6.2}$$

Avec :

- α le niveau de confiance,
- n le nombre de prédiction considéré,
- $t_{n-1,\alpha-\frac{1}{2}}$ le quantile de niveau α de student à $n-1$ degré de liberté,
- $\bar{X}_n$ la moyenne empirique des différentes prédictions
- S_n la racine de la variance empirique des différentes prédictions

Notons que cet intervalle correspond surtout à une mesure de l'instabilité des prédictions obtenues pour les Random Forest.

La valeur prédite engendrée par cette modélisation correspond à la moyenne de toutes les prédictions. Pour différentes valeurs de n [3],on regarde la corrélation entre valeur réelle et valeur prédite, la pente de la droite$(y_i, \hat{y}_i)$ y_i les valeurs réelles de la base de test et $\hat{y}_i$ les valeurs prédites. On regarde aussi le RMSE et le RMSLE. On obtient les résultats suivants :

n	Correlation	pente	RMSE	RMSLE
10	0.823	0.694	19764.86	1.195
25	0.822	0.695	19824.31	1.205

TABLE 6.1 – comparaison des résultats obtenus pour différents nombres de modèles

La taille des clusters de calcul du package h2o étant limité, nous n'avons pas effectué des tests supplémentaires. Cependant, avec les 2 tests effectués(n =10 et 25) on observe déjà qu'il n'y a pas vraiment de différences au niveau des performances du modèles. Par la suite nous avons retenu une valeur de $n = 10$ subdivisions de la base d'apprentissages. En effet pour des performances identiques, avoir une petite valeur de n permet de réduire le temps de prédiction du modèle.

Ci dessous, quelques prédictions de charges et d'intervalle de confiances réalisés à partir de cette modélisation.

3. nombre de séparation de la base d'apprentissage ;nombre de modèle calibrer

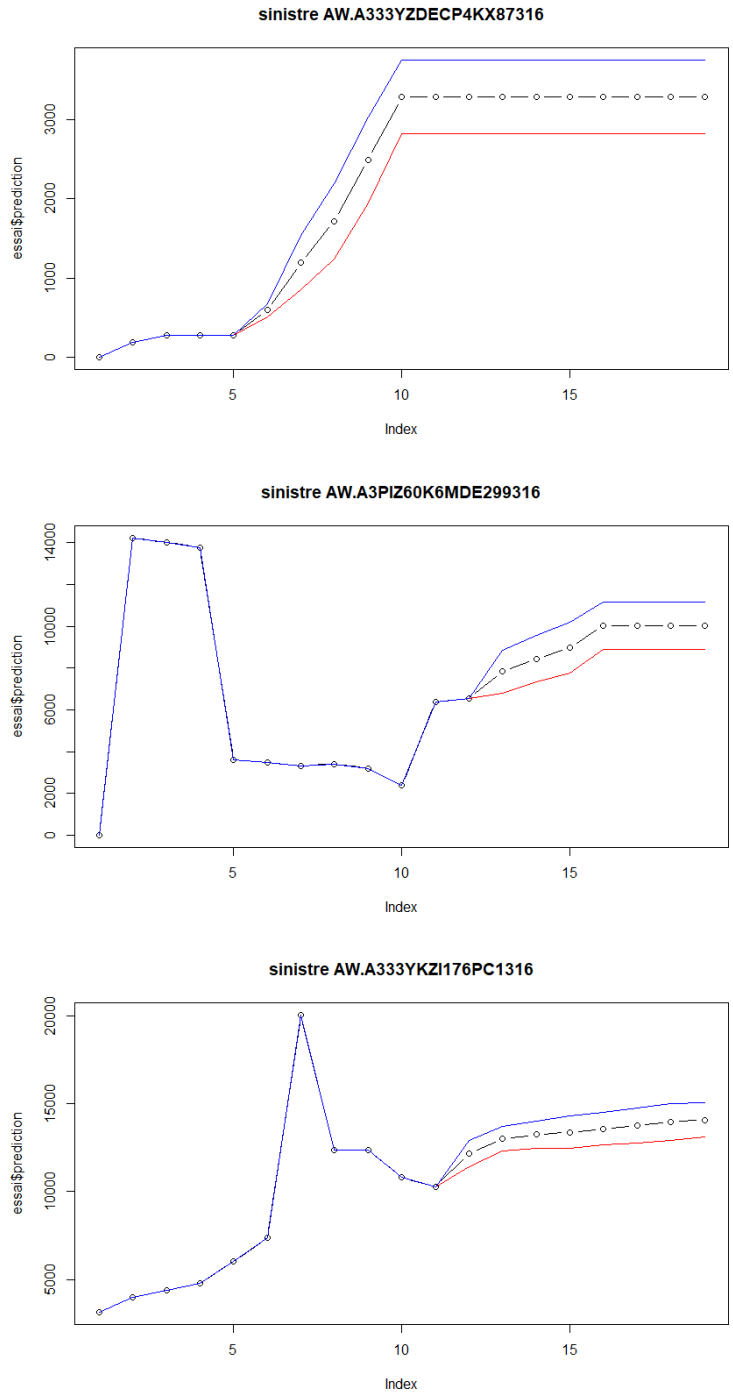


FIGURE 6.2.2 – Exemple de prédiction d'intervalle de confiance pour un sinistre donné

Chapitre 7

D'autres modèles d'apprentissage statistique

7.1 Les Support Vector Machines

Les SVM (Support Vector Machines) sont des classes d'algorithmes très spécifiques, caractérisées par l'utilisation de noyaux (kernel). Au départ défini pour la discrimination entre 2 classes, leur usage à été étendu pour des problèmes multi-classes et pour la régression.cette partie s'appuie sur les références[8] et [9] pour la classification et sur [10] pour la régression.

7.1.1 En classification

On distingue plusieurs cas de figure dans l'application d'un SVM en classification.Les observations peuvent être séparables ou non, pour cela on utilise 2 types de séparateur :

- les séparateurs linéaires
- les séparateurs non linéaires

Séparateur linéaire

Un problème de discrimination est dit linéairement séparable lorsqu'il existe un séparateur linéaire ou encore règle de discrimination, de la forme :

$$S(x) = signe(f(x)) \; avec \; f(x) = w'x + b, w \in \mathbb{R}^p et \; b \in \mathbb{R} \tag{7.1}$$

La fonction $S(x)$ étant supposée classer correctement toutes les observations de l'ensemble d'apprentissage vaut -1 ou 1 [1]. Ce séparateur permet donc de discriminer les variables entres elles. Prenons l'exemple de deux classes linéairement séparables, l'objectif est de trouver la marge maximale permettant de séparer les 2 sous groupes.Cette marge est le résultat d'une optimisation sous contrainte.

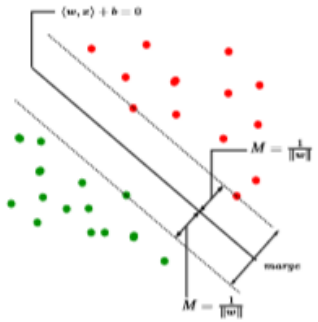


FIGURE 7.1.1 – séparation linéaire

1. Les valeurs -1 et 1 représente les 2 classes

Lorsque les sous groupes ne sont pas linéairement séparables, on peut toujours utiliser un séparateur linéaire en introduisant des variables d'écart. On va autoriser certains exemples à avoir une marge < 1.

Séparateur non linéaire

Dans le cas général, la frontière optimale est non linéaire. Dans le cadre des SVM, la prise en compte de non linéarités dans le modèle s'effectue par l'introduction de noyaux non linéaires. Sans hypothèses sur la nature du domaine des observations D, un noyau k est défini de manière générale comme une fonction de deux variables sur R :

$$k : D, D \to \mathbb{R}, (x, x') \to k(x, x') \tag{7.2}$$

nom	$k(x, y)$
noyau polynomial	$(cx.y)^d$ avec c $\in \mathbb{R}$
noyau gaussien	$exp(-\frac{\|\|x-y\|\|^2}{2\sigma^2})$
noyau tangeante hyperbolique	$tanh(c_1 x.y + c_2)$ avec c_1 et $c_2 \in \mathbb{R}$

TABLE 7.1 – Des exemples de noyaux

7.1.2 En régression

l'algorithme en classification ou en régression utilise les mêmes principes avec seulement quelques différences mineures. Au lieu de valoir -1 ou 1 les étiquettes peuvent maintenant prendre n'importe quelle valeur réelle. De plus, On considère qu'un séparateur linéaire est correct à ϵ près si :

$$\forall i, | < w, x_i > +b - y_i| \leq \epsilon \tag{7.3}$$

avec y_i la valeur à prédire,tout en autorisant des exemples à ne pas satisfaire au critère (au prix d'une pénalité)

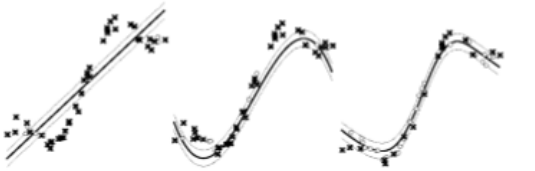


FIGURE 7.1.2 – Un exemple d'ajustement avec différents noyaux

Sur R, cet algorithme s'utilise par le biais du package e1071 et de la fonction svm.

7.1.3 Modélisation du problème sous R

Il a été possible de calibrer un svm radial (noyaux gaussien) à l'aide du package e1071 de R. Le jeu de données étant important et la calibration d'un unique modèle demandant beaucoup d'heures de calibration, l'optimisation du paramètre σ^2 du noyau gaussien, n'a pas été possible. Ainsi, un σ^2 par défaut qui correspond à $\frac{1}{ncol(X)}$ a été considéré. Le résultat de la prédiction sur la base de test est le suivant :

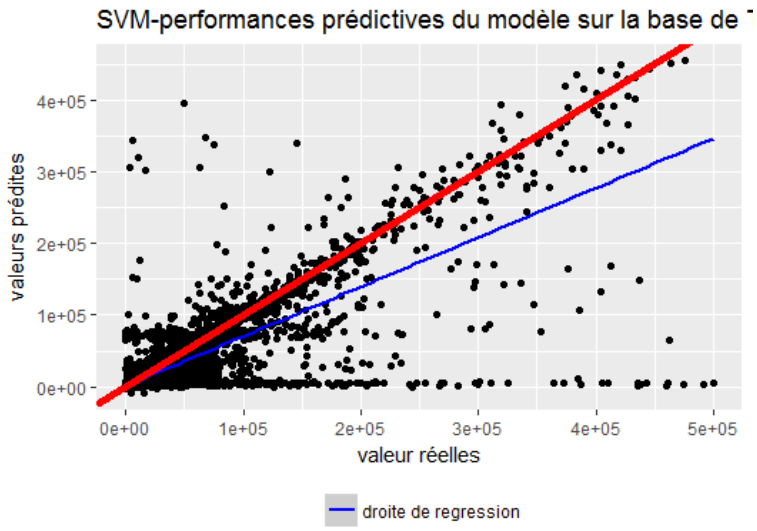


FIGURE 7.1.3 – Comparaison entre prédiction et valeurs réelles d'un SVM

Cette représentation graphique met en avant une assez forte corrélation entre la charge prédite par ce modèle et les valeurs réelles, cependant la droite de régression étant assez en deçà de la droite d'équation $y = x$ (en rouge), on peut affirmer que malgré un assez bon ajustement, le modèle a tendance à sous-estimer les montants de charges.

7.2 Deep Learning

Les réseaux de neurones sont inspirés par les neurones du cerveau humain. Ils sont constitués de plusieurs neurones artificiels connectés entre eux. Plus le nombre de neurones est élevé, plus le réseau est dit profond. Le deep learning est une méthode d'apprentissage statistique qui permet de construire un modèle de réseau neuronal profond. Toujours à l'aide du package h2o, nous avons pu modéliser le montant des charges. On obtient le résultat suivant :

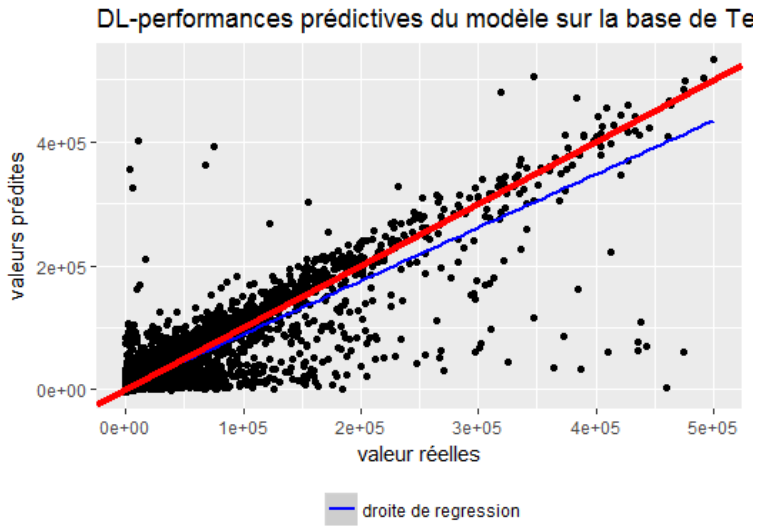


FIGURE 7.2.1 – Comparaison entre prédiction et valeurs réelles d'un Deep learning

Cette représentation graphique met aussi en avant une assez forte corrélation entre la charge prédite par ce modèle et les valeurs réelles avec une droite de régression assez proche de la première bissectrice, on peut affirmer que le modèle réalise un bon ajustement, malgré une légère tendance du modèle à sous-estimer les montants de charges.

Chapitre 8

Modèles de durée du sinistre

La durée d'un sinistre ou encore le nombre d'années de développement d'un sinistre est un paramètre qui n'est pas connu d'avance. Aussi, il nous a semblé pertinent de faire une étude sur la durée des sinistres afin de pouvoir prédire entre autres son nombre d'années de développement. La connaissance de ce paramètre nous permettra par la suite de savoir jusqu'à quand s'arrêter dans l'exécution de nos modèles de prédiction des charges.

8.1 Loi de durée de vie résiduelle du sinistre

Dans l'optique de prédire la durée des sinistres, nous avons tout d'abord commencé par étudier la loi de durée de sinistre observée dans notre base de données. Pour cela, On considère la base restreinte constituée des sinistres définitivement clos au cours de la dernière année. En effet pour ces sinistres en question, on est capable de calculer la durée du sinistre comme étant :

$$duree = Vue - an\_de\_surv \tag{8.1}$$

Aussi, par le biais d'une jointure entre la table et la table constituée de toutes les années de développement de tous les sinistres on est capable d'avoir une table ayant l'information de la durée du sinistre sachant l'année de développement considérée. La variable à étudier est calculée comme suit :

$$duree\_restante = duree - annee\_dev \tag{8.2}$$

Nous avons aussi effectué une randomisation [1] de la durée afin de la rendre continue. Afin d'étudier la loi, on trace tout d'abord son histogramme.

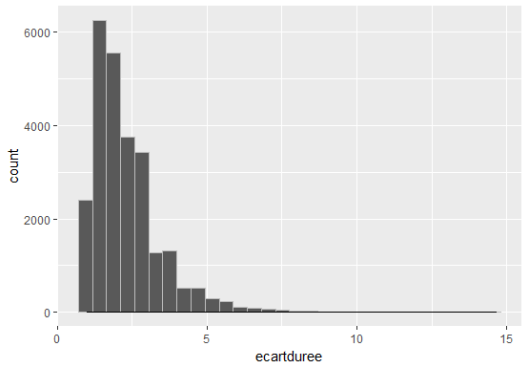


FIGURE 8.1.1 – histogramme de la durée restante

Nous avons par la suite étudié plusieurs lois qui semblaient pertinentes à savoir :

- loi log-gamma
- loi exponentielle
- loi log-normale

1. ajout d'un nombre aléatoire entre 0 et 1

- loi de weibull

A l'aide de la fonction fitdistr de la librarie MASS, on est capable de trouver les paramètres de chaque loi qui permettrait d'expliquer le modèle. On peut ainsi obtenir les différentes densités associées et les comparer à la densité des durées restantes.

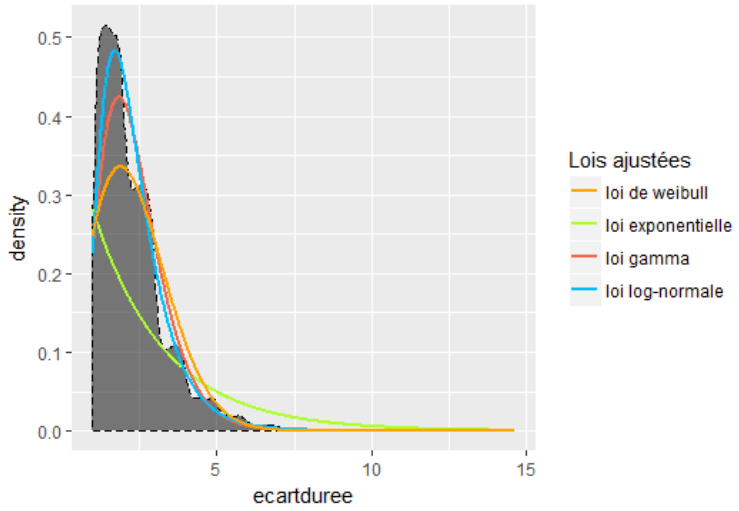


FIGURE 8.1.2 – densité de la durée restante comparée à la densité de différentes lois

Les deux lois qui semblent le plus adaptées sont les lois gamma et log-normale. Cependant, on peut déja se positionner en faveur d'un GLM-log normal quant au type de GLM à choisir car il a un meilleur ajustement pour la densité.

8.2 Modélisation par GLM

8.2.1 Modélisation

En adéquation avec la loi trouvée précédemment, on calibre le modèle GLM log-normale à l'aide du package H2o [2]. Les variables explicatives sélectionnées pour ces modèles sont les variables :

```
"Vue"              "an_de_surv"       "annedev"          "Type_de_sinistre" "type_repar"
"Permis"           "Tiers_acc"        "Tiers_domm"       "Constat"          "Police"
"Chargement_veh"   "blesse"           "Usage"            "Type_acc"         "Type_m"
"Assur"            "Prop"             "Exp"              "Expa"             "Taux1"
"Taux2"            "dureeSin"         "Taux3"            "Degat"            "Lieu"
"ecartduree"
```

FIGURE 8.2.1 – variables choisies pour la modélisation

Les variables retenues par le package h2o pour l'évaluation de ce modèle sont :

```
 [1] "Type_de_sinistre" "Expa"             "dureeSin"         "Taux3"            "Type_acc"         "Permis"
 [7] "Usage"            "Tiers_acc"        "Tiers_domm"       "Type_m"           "Police"           "Chargem
[13] "blesse"           "Vue"              "an_de_surv"       "annedev"
```

FIGURE 8.2.2 – variables retenues

8.2.2 Performances du modèle

Les performances de ces modèles sont évaluées avec les mesures suivantes :

2. voir annexe

Performances	log-normal
RMSE	1.045
R2	0.155
AIC	38009.24
BIC	38139.86
Correlation [3]	0.40

TABLE 8.1 – performances des modèles de durée

On peut aussi visuellement se représenter les performances de ce modèle au travers du graphique suivant, représentant les valeurs prédites par rapport aux valeurs réelles.

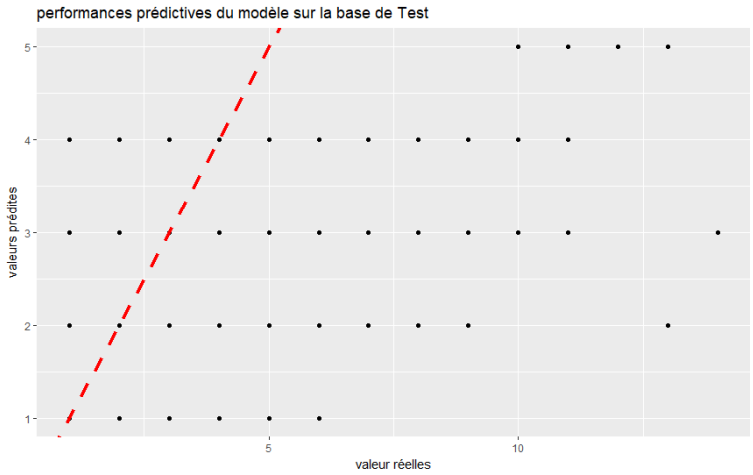


FIGURE 8.2.3 – valeurs prédites par les modèles en fonction des valeurs réelles

Aussi bien graphiquement que au niveau des performances chiffrées, on peut voir que les performances de ce modèle ne sont pas très bonnes. Aussi il sera intéressant pour nous de créer d'autres modèles qui pourraient éventuellement améliorer les performances actuelles.

8.3 Algorithmes d'apprentissages statistiques

Outre les GLM, on peut aussi prédire la durée de vie résiduelle d'un sinistre en utilisant des méthodes parmi celles décrites précédemment tels que les forêt aléatoires et le deeplearning par exemple, sur la même base d'apprentissage.

8.3.1 Deep learning

Les performances de ces modèles sont évaluées avec les mesures suivantes :

	Correlation	pente	RMSE
performances	0.54	0.31	0.990

TABLE 8.2 – performances des modèles de durée

On peut aussi visuellement se représenter les performances de ce modèle au travers du graphique suivant, représentants les valeurs prédites par rapport aux valeurs réelles.

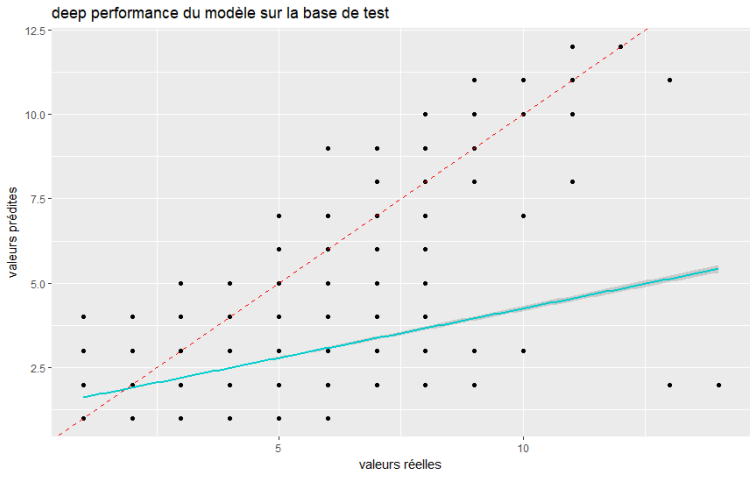


FIGURE 8.3.1 – valeurs prédites par les modèles en fonction des valeurs réelles

8.3.2 Random Forest

Les performances de ces modèles sont évaluées avec les mesures suivantes :

	Corrélation	pente	RMSE
performances	0.46	0.321	1.053

TABLE 8.3 – performances des modèles de durée

On peut aussi visuellement se représenter les performances de ce modèle au travers du graphique suivant, représentants les valeurs prédites par rapport aux valeurs réelles.

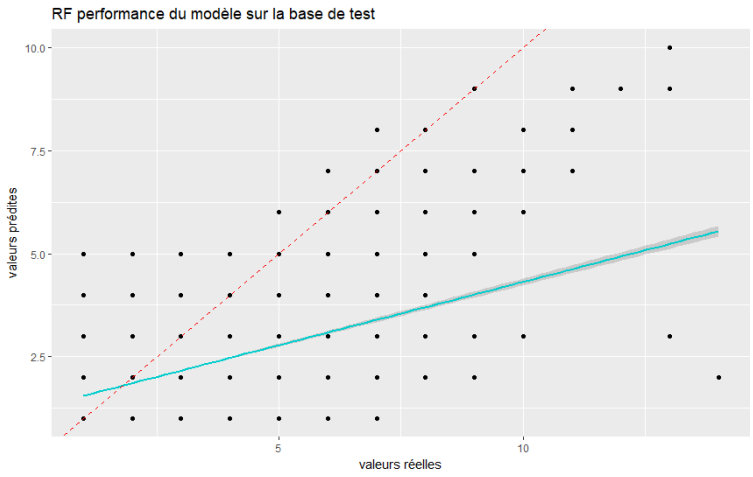


FIGURE 8.3.2 – valeurs prédites par les modèles en fonction des valeurs réelles

Chapitre 9

Prédiction de la charge à l'ultime des sinistres

9.1 Prédiction directe de la charge ultime

Le but ici est de prédire le coût global du sinistre au moment de la clôture définitive. En effet si les résultats de cette prédiction sont cohérents, on aura limité les temps de calcul en gardant la qualité de prédiction.

9.1.1 Principe de prédiction

Afin de réaliser cette prédiction, nous avons modifié notre base en faisant tout d'abord une extraction des sinistres en état définitivement clos, avant d'ajouter sur cette nouvelle base le nombre d'années de développement du sinistre ainsi que le nombre de fois qu'il a été réajusté.Après avoir réordonné notre base comme expliqué précédemment, on peut calibrer notre modèle en trois étapes. Ce sont les suivantes :

1. calibrer un modèle qui prédit le nombre d'années de développement en fonction de toutes les caractéristiques du sinistre [1].
2. Calibrer un modèle qui prédit le nombre d'ajustements du sinistre en fonction de toutes ses caractéristiques.
3. Ajuster un modèle qui prédit la charge en fonction de tous les paramètres du sinistre y compris le nombre d'années de développement et le nombre d'ajustements du sinistre.

9.1.2 Modèles de prédiction des années de développement

Une première étape dans la prédiction de la charge à l'ultime consiste à prédire le nombre d'années de développement d'un sinistre. Dans cette optique nous avons utilisé plusieurs méthodes algorithmique tel que le deeplearning ou encore les Forêts aléatoires.

Le deeplearnig

Le modèle deeplearning est un modèle d'apprentissage statistique qui construit un réseau de neurones artificiels multicouches. C'est une branche des algorithmes d'apprentissage automatique qui sont inspirés par le fonctionnement du cerveau humain et du système nerveux. Les données d'entrée sont bloquées sur la couche d'entrée du réseau profond. Le réseau utilise ensuite une hiérarchie de couches cachées qui produisent successivement des abstractions (représentations compactes) de niveau supérieur des données d'entrée. Il produit ainsi des pseudo-caractéristiques des entités en entrée. Aussi, après avoir implémenté l'algorithme deeplearning du package h2o, nous avons obtenu les résultats suivants.

1. on exclu bien évidement des caractéristiques la charge et le nombre d'ajustement

Importance des variables explicatives du modèle

Le graphique ci dessous permet de voir l'effet significatif des variables explicatives sur le nombre d'année de développement du sinistre.

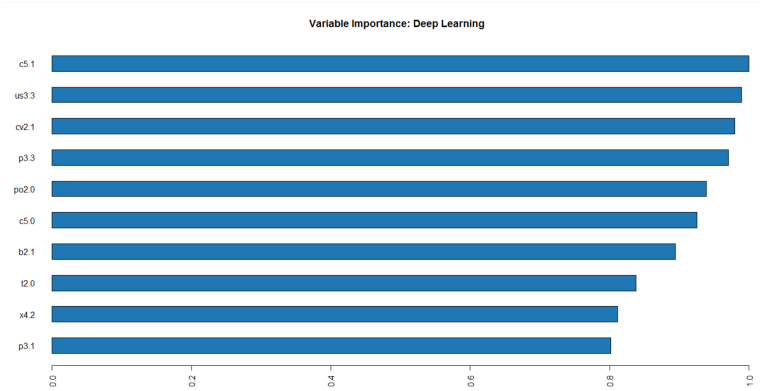


FIGURE 9.1.1 – Importance des variables du modèle deeplearning

D'après la figure ci dessus, toutes les variables explicatives du modèles ont un effet assez significatif sur le nombre d'année de développement (importance $> 80\%$). La variable Constat présente un effet significatf à 100% sur le nombre d'années de développement.

Personnalisation du modèle de deeplearning

Afin d'améliorer les qualités prédictives de notre modèle deeplearning,nous avons décidé d'optimiser le RMSE en agissant sur le paramètre epochs [2]

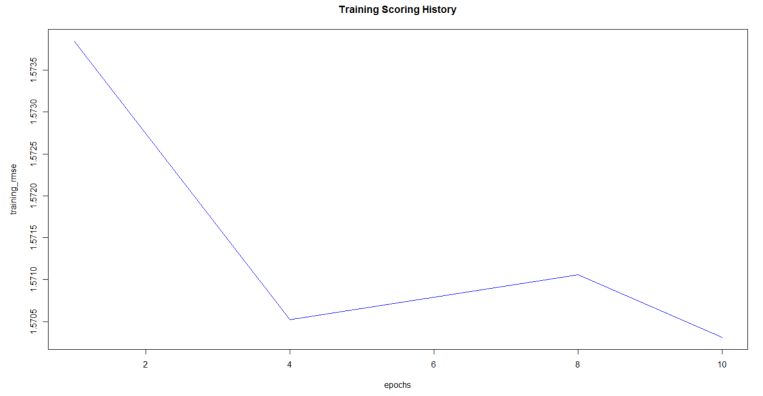


FIGURE 9.1.2 – Représentation du RMSE en fonction de l'epochs

D'après la courbe ci dessus, nous fixons le nombre d'itération à 10 afin de minimiser le RMSE, puis nous ajustons le modèle deeplearning en intégrant ce paramètre (epochs=10).

Forêts aléatoires

Modélisation sous R

Après avoir implémenté l'algorithme Random Forest sur R, nous avons obtenu les résultats suivants :

Importance des variables explicatives du modèle

2. Le nombre de fois que l'ensemble de données doit être itéré.

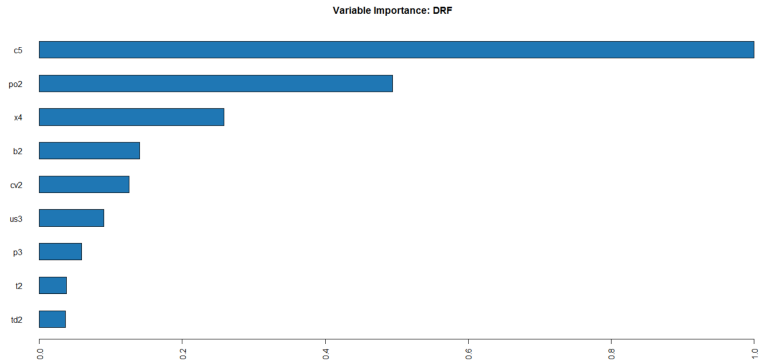


FIGURE 9.1.3 – Importance des variables pour le modèle Random Forest

Comparaison et choix du meilleur modèle

Mesures de performance	deep Learning	random forest
MSE	2.465879	2.449924
RMSE	1.570312	1.565223
MAE	1.141715	1.116305
RMSLE	0.5481111	0.5399305

TABLE 9.1 – Tableau de comparaison des modèles calibrés

En comparant ces indicateurs de performance, on retient le modèle de deep learning pour prédire le nombre d'années de développement des sinistres.

9.1.3 Modèles de prédiction du nombre d'ajustement

Dans cette section, nous ajustons des modèles qui prévoient le nombre de fois que le montant de la charge a été réajusté tout au long de la durée de vie du sinistre (tant qu'il n'est pas définitivement clos).

Modèle Deep learning

Dans un premier temps, on calibre le modèle avec les paramètres à défaut puis on l'améliore en optimisant le choix des paramètres :

Résultat et importance des variables

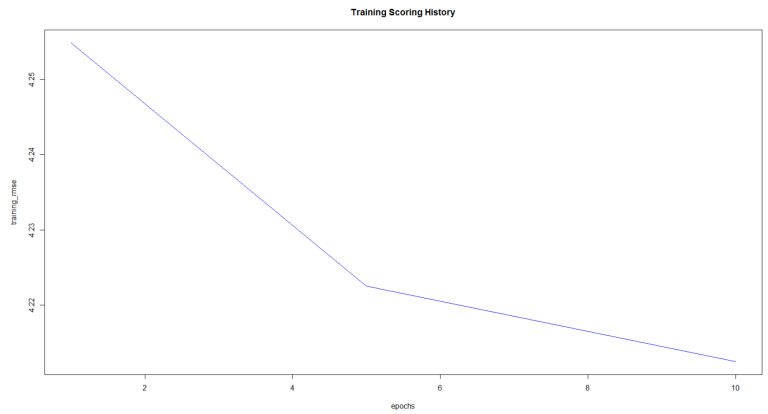


FIGURE 9.1.4 – Représentation du RMSE en fonction de l'epochs

A partir de la courbe ci dessus, on peut choisir epochs=10. Il est ensuite possible d'ajuster le modèle en intégrant ce paramètre.

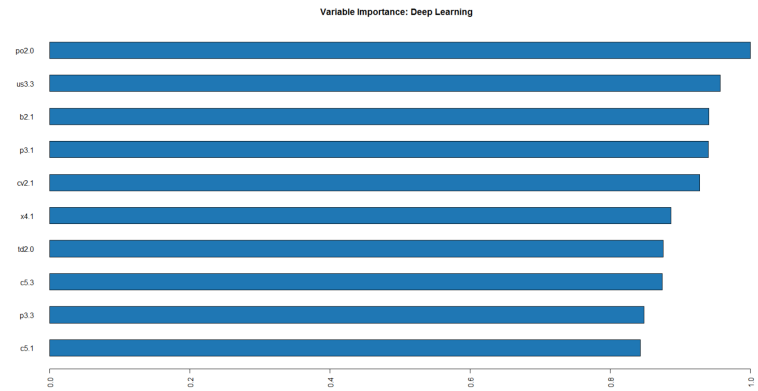


FIGURE 9.1.5 – Importance des variables sur le deep learning

On peut remarquer ici que toutes les variables ont un effet assez significatif sur le modèle deep learning du nombre d'ajustement.

Tableau de performance du deep learning

Mesures de performance	deep Learning
MSE	17.74483
RMSE	4.212461
MAE	3.023791
RMSLE	0.45288

TABLE 9.2 – Tableau de performance du deeplearning

9.1.4 Prédiction de la charge à l'ultime

Après avoir ajusté des modèles de prédictions des années de développement et du nombre d'ajustement des sinistres, nous implémentons par la suite des modèles de prédiction directe de la charge ultime en fonction de toutes les caractéristiques du sinistre y compris le nombre d'années de développement et le nombre d'ajustements. En calibrant un premier modèle deep learning, on constate que la qualité de prédiction du modèle n'est pas très bonne. En effet, les indicateurs de performances sont très élevés. On a par exemple :

$$RMSE = 19129.58 \; et \; MAE = 6234.592 \tag{9.1}$$

Importance des variables

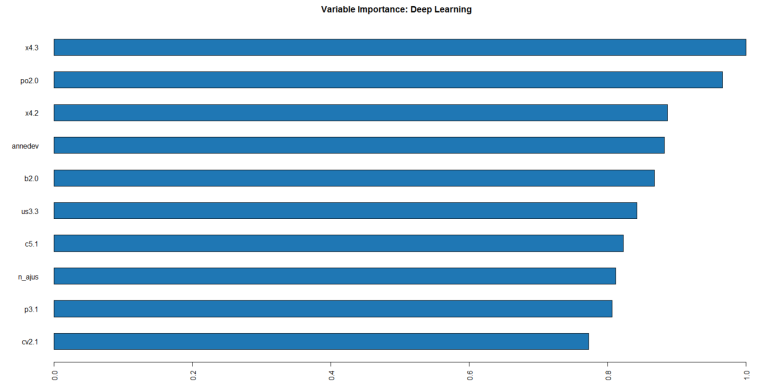


FIGURE 9.1.6 – Importance des variables sur la charge

Variables retenues pour calibrer le nouveau modèle

- Police
- Blessé
- Chargement-veh
- Tiers dommage
- Tiers acc
- annedev
- n-ajus
- Taux 3

Amélioration du modèle de Deep learning

Après ajustement d'un nouveau modèle deep learning avec les variables les plus significatives,nous constatons toujours que la qualité de ce modèle n'est pas assez bonne.

Mesures de performance	deep Learning
RMSE	9997.97
MAE	4191.502

TABLE 9.3 – Indicateurs de performance du nouveau modèle

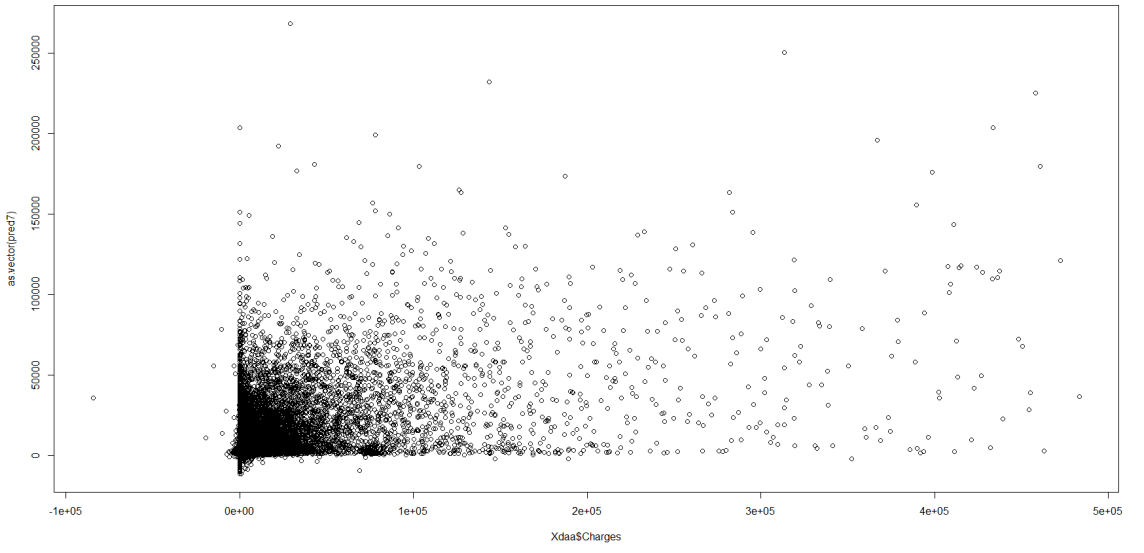


FIGURE 9.1.7 – Représentation de la valeur prédite en fonction de la valeur réelle

Les très mauvaises performances de ce type de modélisation nous ont poussé à plutôt étudier des modèles de prédiction de l'évolution des charges.

9.2 Prédiction de l'évolution des sinistres

L'objectif final de cette étude est de prédire la charge à l'ultime des sinistres. A l'aide de tous les modèles calibrés, on peut essayer de prédire l'évolution du sinistre et donc de sa charge à l'ultime.

9.2.1 Principe de prédiction

On travaille sur la base restreinte des sinistres qui ne sont pas encore définitivement clos et cours de leur dernière année connue [3]. Le principe de prédiction est le suivant :

1. A partir des caractéristiques du sinistre et de l'année de développement en cours, on utilise le modèle de durée restante calibré pour prédire le nombre d'années ($nban$) de développement restant
2. On effectue $nban$ prédictions du sinistre en prenant soin de changer la charge précédente par La charge prédite et d'incrémenter l'année de développement courante de 1

On peut résumer le modèle comme suit :

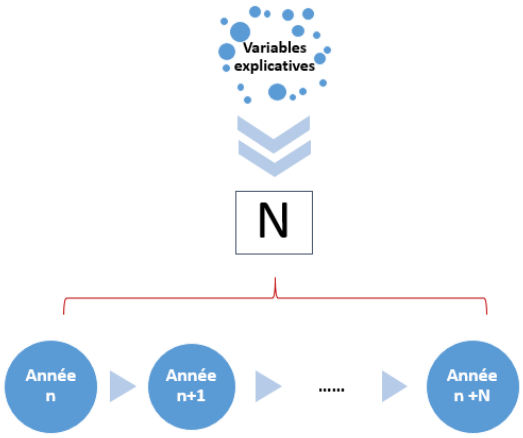


FIGURE 9.2.1 – modèle de prédiction

Il faut distinguer les cas des modèles que l'on a appelé modèles simples et le cas des modèles évolutifs. Pour les modèles simples, on utilise le même modèle pour prédire toutes les années de développement, et pour les modèles évolutifs, on utilise un modèle par année de développement. La dernière charge prédite par ce modèle correspond à la charge ultime.

3. ceux dont les variables $Fin\_vue =$ "9990"", et $Etat =$ "$Clos$"

9.2.2 Des résultats de prédiction pour un modèle GLM de durée restante

Les résultats suivants ont été obtenus en appliquant la méthode de prédiction précédemment explicitée pour chaque sinistre considéré avec un modèle GLM log-normal pour la durée du sinistre. On agrège ensuite les résultats en faisant la somme par année de développement pour tous les sinistres. Notons qu'on se place dans la base des sinistres qui ne sont pas définitivement clos, et on prédit à partir de la dernière année connue. [4].

GLM_simple

On observe les résultats suivants :

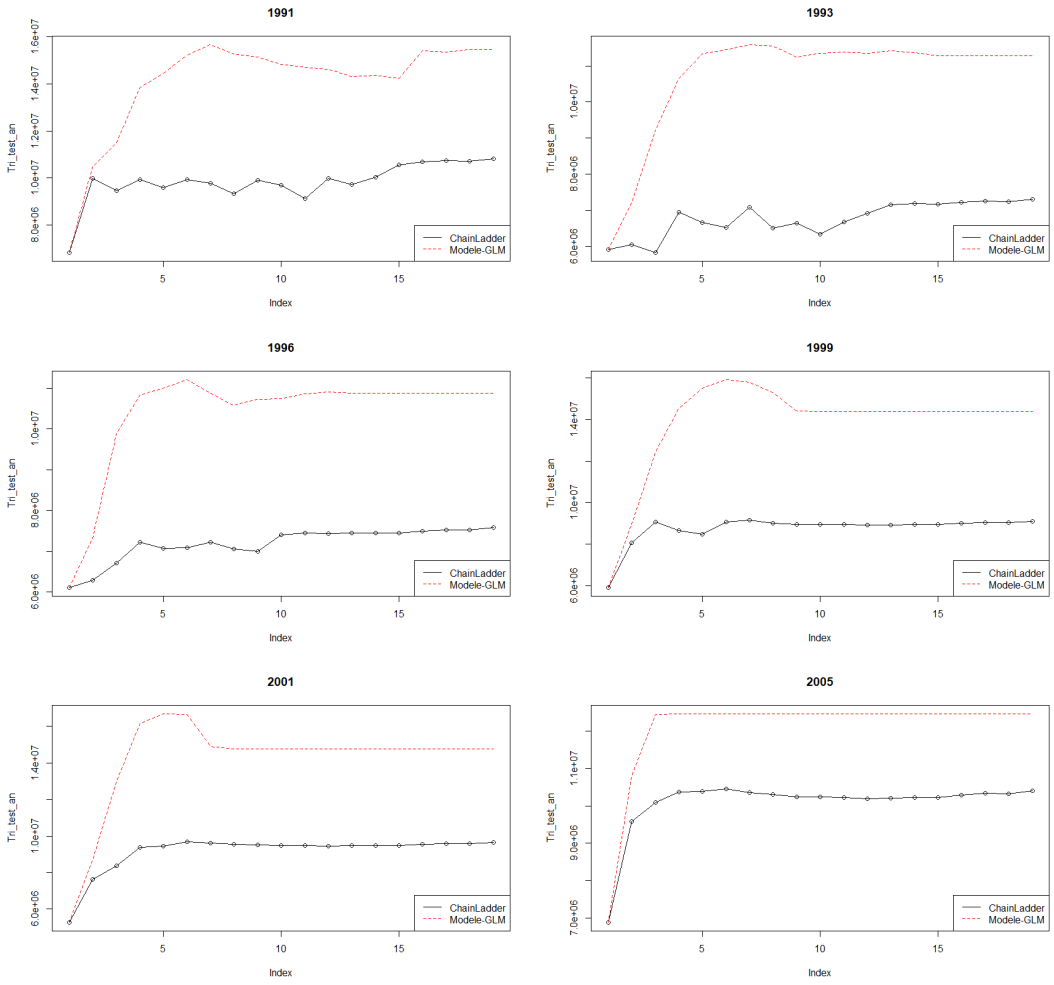


FIGURE 9.2.2 – Prédiction de l'évolution pour un modèle glm log-normal pour plusieurs années de survenance

Ce modèle est très éloigné des prédictions réalisées par chain ladder. Malgré le fait qu'il ait montré une tendance à sous estimer la prédiction sur la base de test, il surestime beaucoup l'évolution des charges. Le fait que toutes les hypothèses du modèle GLM ne soient pas vérifiées peut être un élément de réponse quant aux mauvaises prédictions de ce modèle.

4. la variable "Fin_Vue" = 9990

SVM_simple

On observe les résultats suivants :

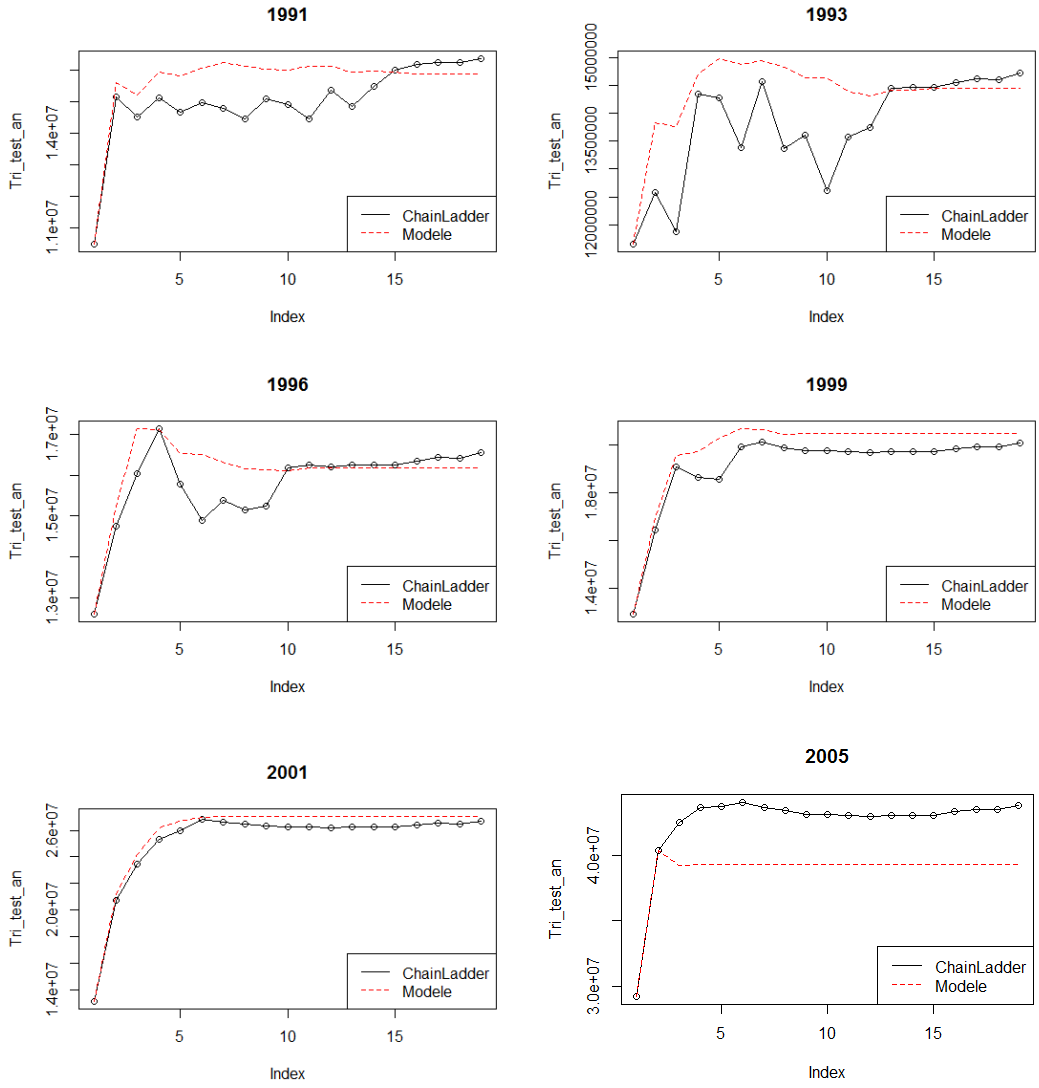


FIGURE 9.2.3 – Prédiction de l'évolution pour un modèle svm pour plusieurs années de survenance

On peut constater que l'évolution prédite par le modèle SVM simple est très proche de l'évolution donnée par les méthodes usuelles que l'on considère tout au long de notre étude comme référence. Cette proximité varie cependant en fonction de l'année de développement et de l'année de survenance considérée. En effet les résultats sont les plus proches pour les dernières années de développement et les dernières années de survenance.

Deep learning

On observe les résultats suivants :

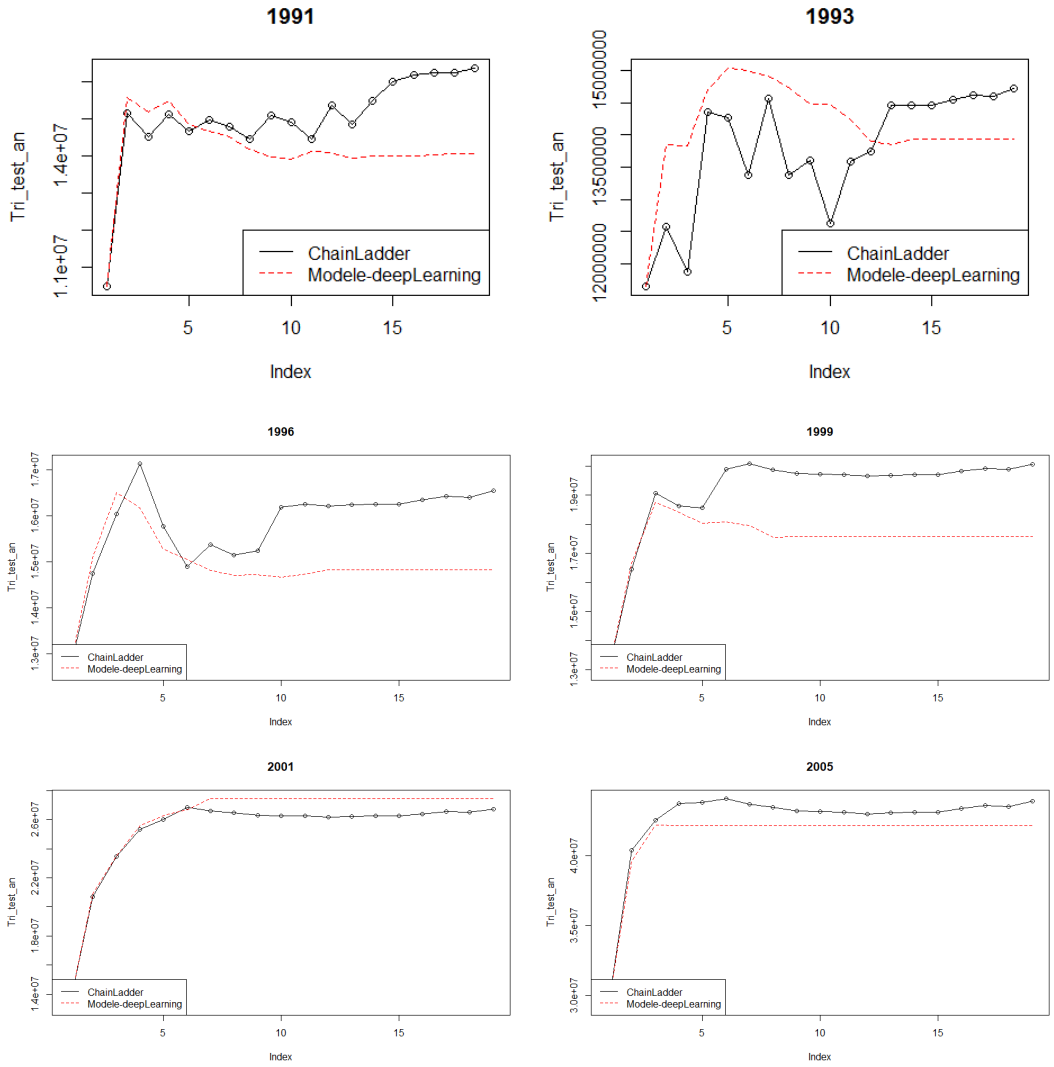


FIGURE 9.2.4 – Prédiction de l'évolution pour un modèle simple Deep Learning pour plusieurs années de survenance

On peut constater que l'évolution prédite par le modèle deep learning est aussi très proche de la prédiction faite par Chain ladder. Cependant le dénivellement des dernières années est moins présent dans ce modèle.

Random Forest avec intervalles de confiance

On observe les résultats suivants :

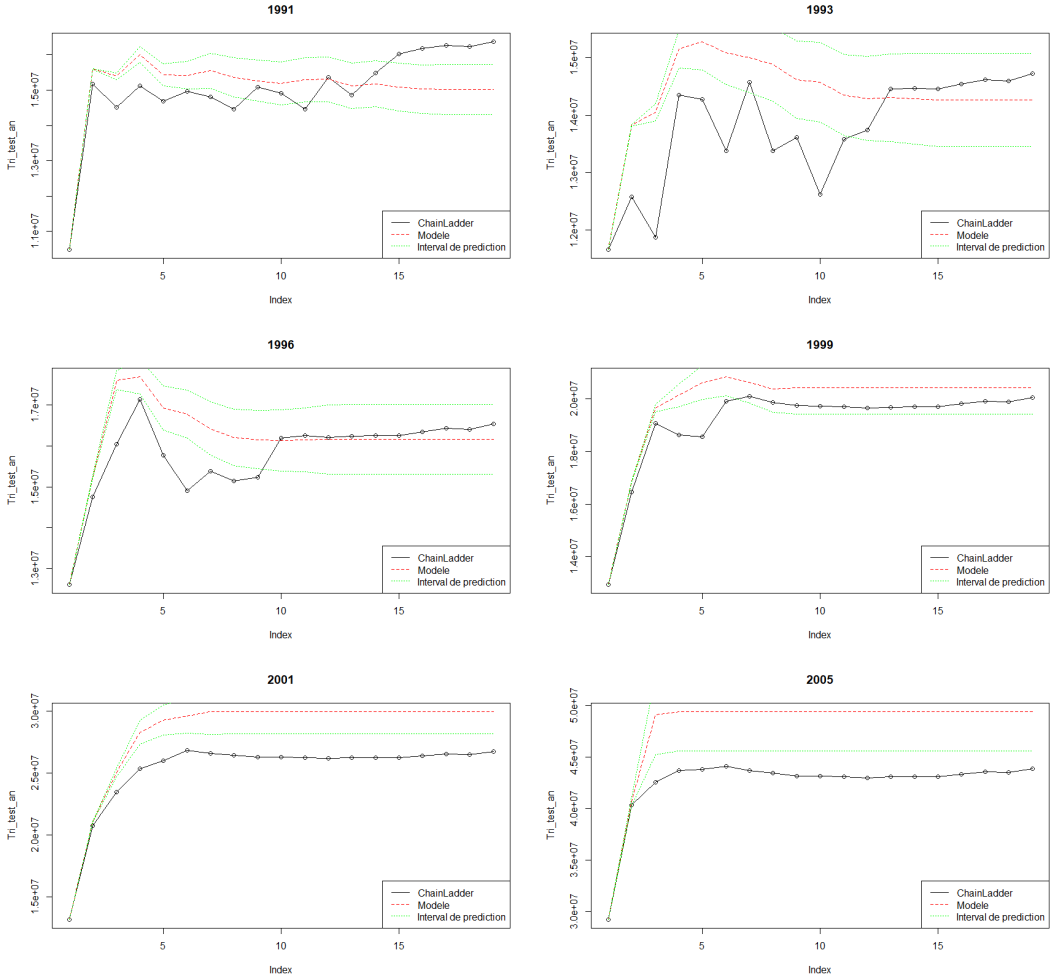


FIGURE 9.2.5 – Prédiction de l'évolution pour un modèle Random Forest pour plusieurs années de survenance

La prédiction réalisée par ce modèle semble être cohérente avec celle réalisée par Chain Ladder. Ce modèle a l'avantage de fournir des intervalles de confiances pour nos prédictions.
Quelque soit le modèle considéré, la plus grande différence entre la prédiction réalisée par Chain Ladder et les nôtres se situent au niveau des dernières années de survenance. En effet cette différence vient très certainement de la prédiction de la durée restante qui ne semble pas adéquate dans les dernières années.

9.2.3 Des résultats de prédiction pour un modèle Deep Learning de durée restante

Les résultats suivant ont été obtenu en appliquant la méthode de prédiction précédemment explicitée pour chaque sinistre considéré avec un modèle Deep Learning pour la durée du sinistre. On agrège ensuite les résultats en faisant la somme par année de développement pour tous les sinistres. Notons qu'on se place dans la base des sinistres qui ne sont pas définitivement clos, et on prédit à partir de la dernière année connu.

Modèle de charges - Deep learning

On observe les résultats suivants :

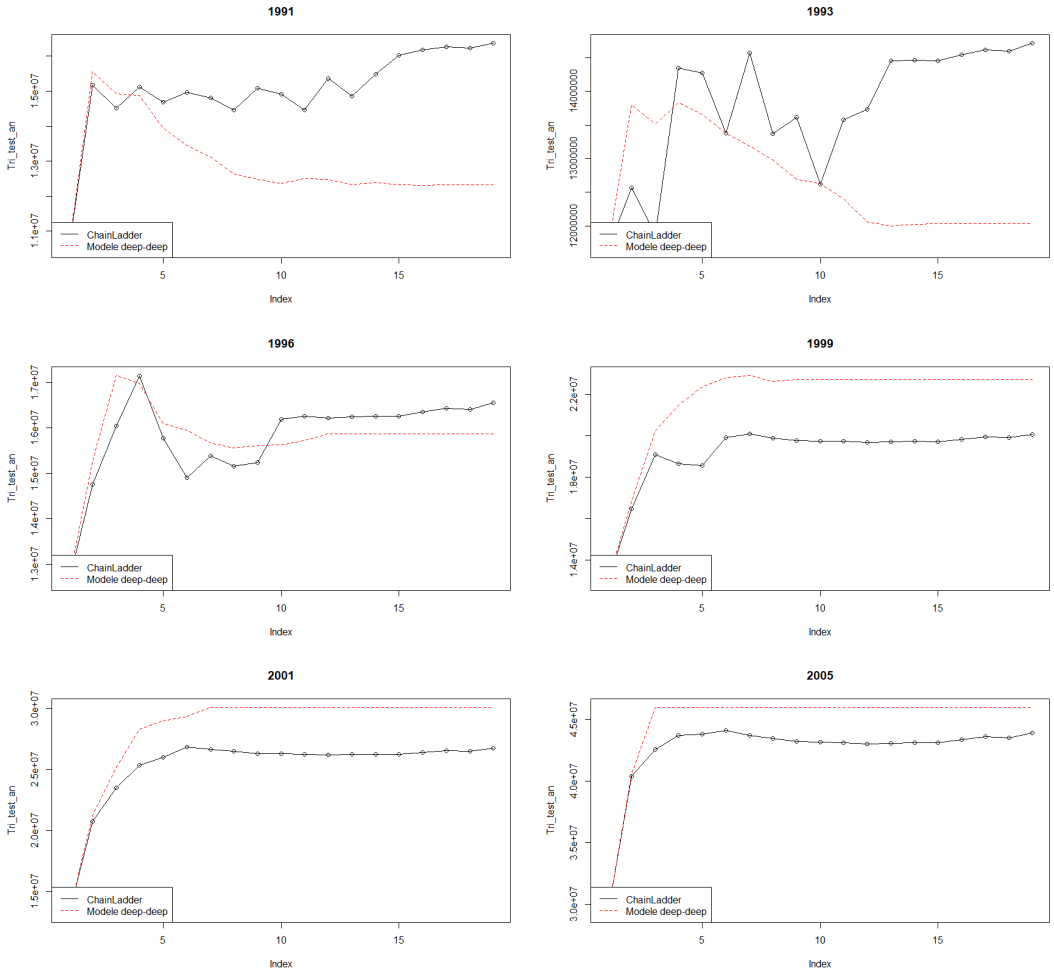


FIGURE 9.2.6 – Prédiction de l'évolution pour un Deep Learning comme modèle de charges et Deep Learning comme modèle de durée

Ces graphiques laissent paraître le fait que ce modèle sous-estime l'évolution des charges pour les premières années de survenance(1989,1993). Cependant pour les après quelques années de survenance, la prédiction est plus proche que celle de chain Ladder. De plus, dans les dernières années de survenance on peut aussi constater que la prédiction réalisée par notre modèle est supérieure à celle de Chain Ladder. C'est en cela surtout que ce modèle réalisé avec un modèle de durée Deep Learning se distingue de celui réalisé avec le GLM. La durée du sinistre pour les dernières années de survenance est moins sous-estimé qu'avec un GLM, ce qui réhausse la prédiction.

9.2.4 Des résultats de prédiction pour un modèle Random forest de durée restante

Les résultats suivant ont été obtenu en appliquant la méthode de prédiction précédemment explicitée pour chaque sinistre considéré avec un modèle Random Forest pour la durée du sinistre. On agrège ensuite les résultats en faisant la somme par année de développement pour tous les sinistres. Notons qu'on se place dans la base des sinistres qui ne sont pas définitivement clos, et on prédit à partir de la dernière année connu.

Modèle de charges - Deep learning

On observe les résultats suivants :

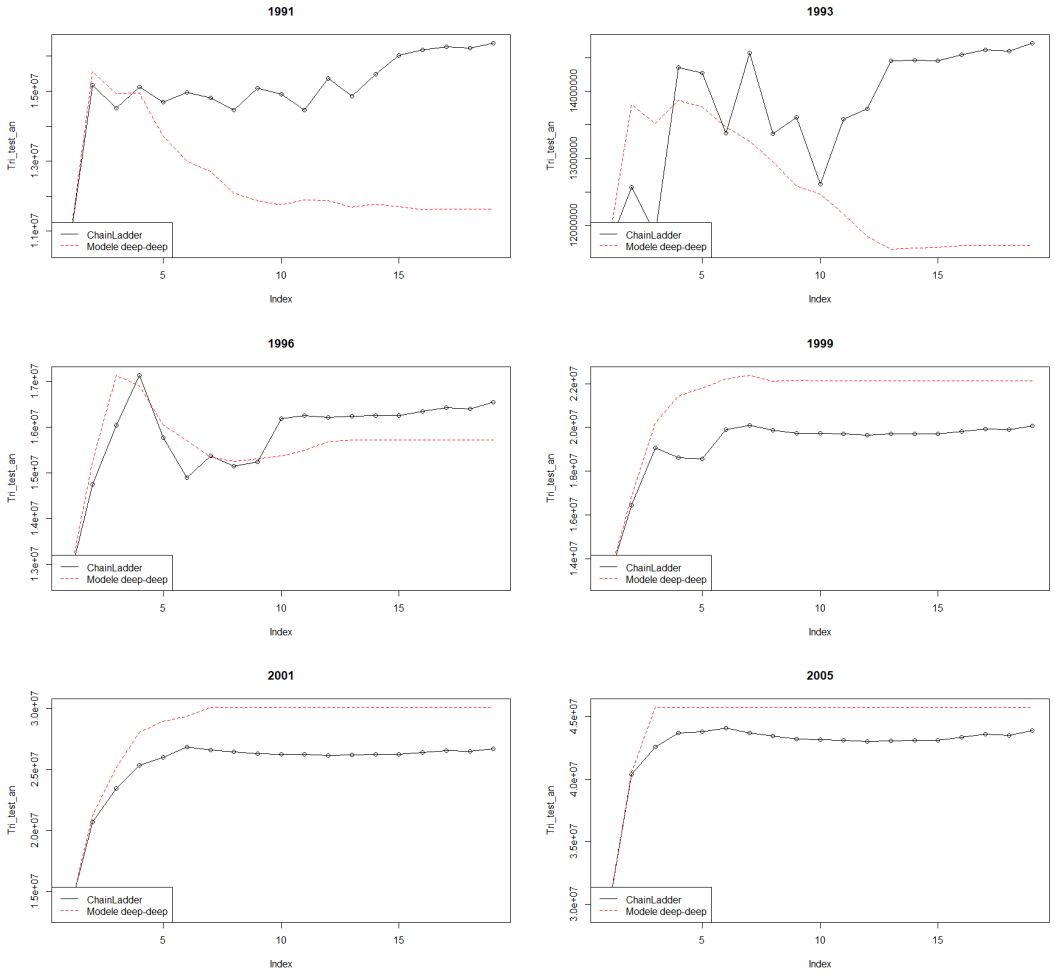


FIGURE 9.2.7 – Prédiction de l'évolution pour un Deep Learning comme modèle de charges et Random Forest comme modèle de durée

L'évolution de ce modèle est sensiblement identique au modèle précédent. Ainsi tout comme le modèle avec comme modèle de durée le deep learning , celui ci sous estime énormément l'évolution de la charge pour les premières années de survenance (les plus anciennes), et est très adapté au années de survenances récentes.

Chapitre 10

Comparaison des modèles

Au cours de ce bureau d'étude, nous avons pu implémenter plusieurs algorithmes afin de répondre à notre problématique qui est la prédiction de la charge à l'ultime des sinistres. Ces méthodes ont été utilisées afin de challenger les méthodes existantes telles que Chain Ladder et Mack. Aussi dans ce chapitre nous mettrons en oeuvre des méthodes de comparaison entre méthodes et modèles.

10.1 Méthodes de comparaisons

Dans un premier temps il est important de définir des méthodes de comparaison entre modèles. On utilisera ici une fonction de désaccord empirique afin de comparer les méthodes lignes à ligne et les méthodes classiques. Cette fonction de désaccord empirique se calculera comme suit :

$$D_n(M) = \frac{1}{n}\sum_{i=1}^{n}(C_{i,n} - M_{i,n})^2 \tag{10.1}$$

avec $C_{i,n}$ la charge ultime pour l'année de survenance i donnée par chain Ladder et $M_{i,n}$ la charge ultime pour l'année de survenance i donnée par le modèle M de Machine Learning considéré. Cette fonction $D_n(M)$ sert à évaluer la proximité d'un modèle aux méthodes classiques. On pourra aussi comparer pour les modèles pouvant le fournir, la réserve calculée. On fera cette comparaison avec les fonctions définies comme suit :

$$ER_M = \frac{R_M - R}{R} \tag{10.2}$$

avec R la reserve de chain Ladder [1] et R_M la réserve calculée par le modèle M de machine learning. Cete mesure permet de quanifier l'erreur relative entre les deux réserves. Un ER_M correspond à une réserve moins importante. De plus, plus ER_M est proche de 0, plus les deux réserves sont proches

Afin de mesurer à quel point les charges prédites par le modèle évoluent comme les charges selon Chain Ladder on utilisera la mesure suivante :

$$C_M = \frac{1}{m}\sum_{i=1}^{m} cor(M_i, C_i) \tag{10.3}$$

avec m le nombre d'années de survenance [2] et $cor(M_i, C_i)$ la corrélation entre l'évolution pour l'année de survenance i considérée pour chain Ladder(C_i) et celle donnée par le modèle M.

On définit par la suite un score de proximité qui mesurera la proximité entre un modèle et Chain Ladder. Ce score se décrit comme suit :

$$score\_prox = log(D_M) + |ER_M| - C_M \tag{10.4}$$

1. Cette réserve est identique à celle de Mack
2. 19 dans notre base de donnée

On pourra comparer les modèles ligne à ligne entre eux sur plusieurs plans tels que le temps d'exécution et les résultats de la prédiction sur une même base de test. Pour quantifier cette proximité entre prédiction et corrélation, on utilise des mesures telles que :

- le RMSE
- la corrélation entre données réelle et données prédites
- la pente de la droite de régression associée aux points $(y, \hat{y})$ avec y les valeurs réelles et $\hat{y}$ les valeurs prédites. Cette pente permet de se rendre compte d'une tendance générale des modèles à sous/sur-évaluer les charges.
- un score de prédiction qui se calcule comme suit :

$$score\_pred = \frac{RMSE}{\bar{Y}} - pente - Correlation \tag{10.5}$$

avec $\bar{Y}$ la moyenne des valeurs réelles.
Ce score permet d'évaluer de manière générale la prédiction du modèle. Plus il est petit, plus la prédiction du modèle est bonne. [3]

10.2 Résultats

10.2.1 Comparaison entre modèles de prédiction de charge

Dans un premier temps, nous avons pu comparer visuellement l'ajustement réalisé par chaque modèle utilisé sur la même base de test. Dans cette partie, nous pourrons comparer quantitativement ces modèles. On peut synthétiser ces résultats avec le tableau suivant :

Modèle	RMSE	Corrélation	Pente	*score_pred*
SVM	20025.74	0.82	0.69	0.223
Random Forest	19776.87	0.82	0.70	0.194
Deep learning	15788.97	0.89	0.79	-0.313
GLM log-normale	26281.91	0.66	0.44	1.169

TABLE 10.1 – comparaison entre modèle de charges

Cette comparaison permet de classer les modèles en fonction de leurs performances prédictives sur la même base de test constituée. On obtient donc un classement en considérant le critère score. Le Deep learning est ici le modèle ayant les meilleures performances prédictives.

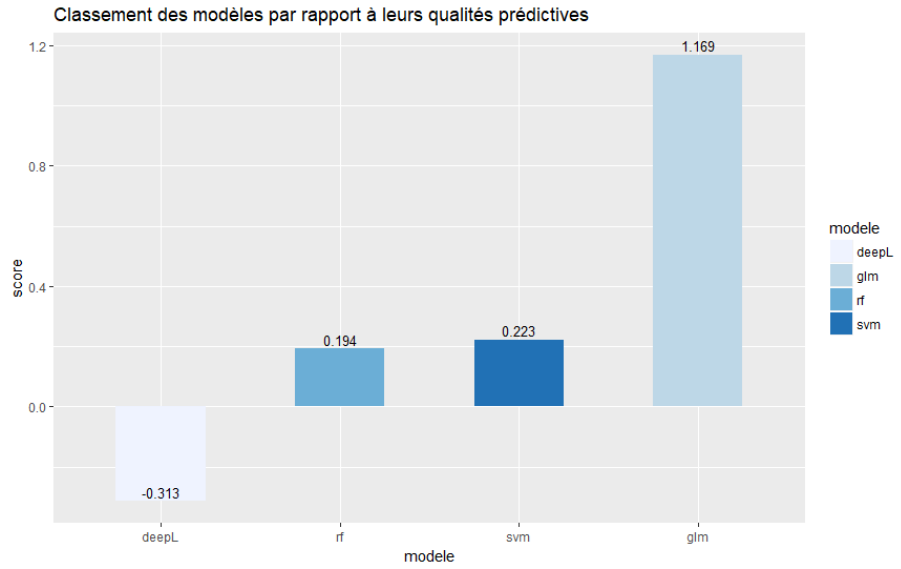


FIGURE 10.2.1 – classement des modèles en fonction de leur qualités prédictives

3. un modèle parfait aurait un score de -2

10.2.2 Comparaison entre modèles de prédiction de durée de vie résiduelle

La description des différents modèles de prédiction de durée de vie résiduelle ayant été expliquée dans la partie modèle de durée, on regroupe tous le résultats dans le tableau suivant :

Modèle	RMSE	Corrélation	Pente	score
Random Forest	1.1	0.35	0.18	0.082
Deep learning	1.02	0.50	0.30	-0.241
GLM log-normale	1.05	0.46	0.30	-0.190

TABLE 10.2 – comparaison entre modèle du durée

Cette comparaison permet de classer les différents modèles considérés en fonction de leur qualité de prédiction sur la même base de test. On obtient donc le classement suivant en considérant le critère score.

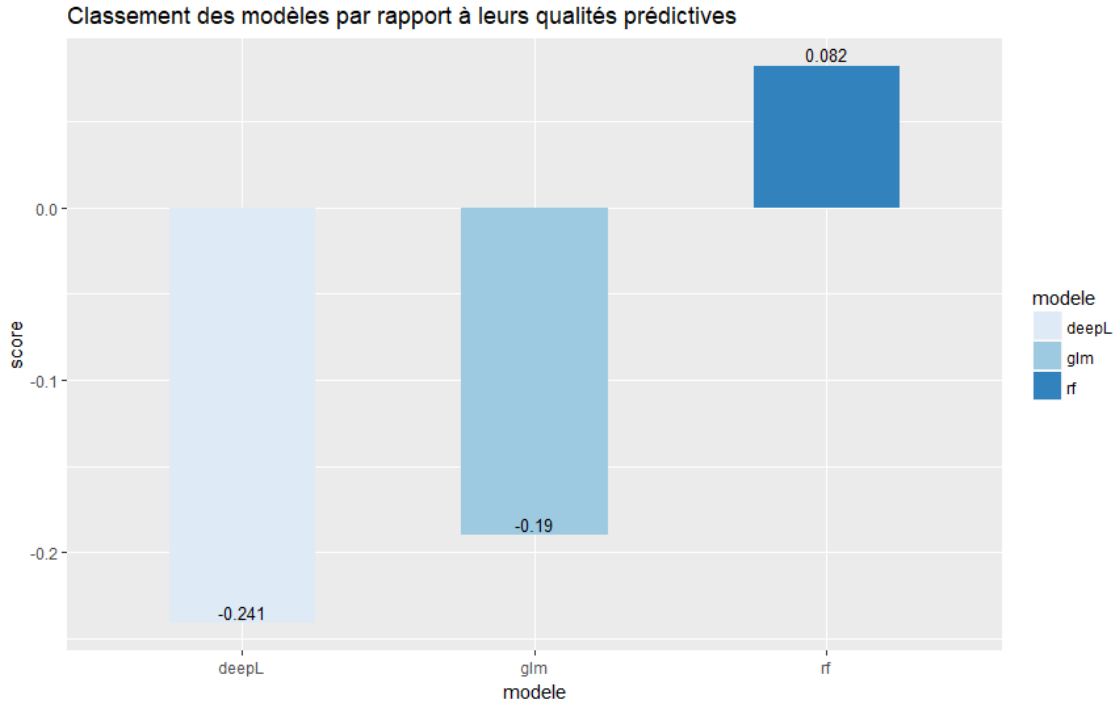


FIGURE 10.2.2 – Classement des modèles de durées

Le Deep Learning semble être le meilleur modèle pour la durée de vie résiduelle au vue de son score.

10.2.3 Comparaison par rapport à Mack/Chain Ladder

On cherche dans cette section à comparer les différents modèles de prédiction de charge à l'ultime. Aussi, peut-on regrouper tous les résultats dans le tableau suivant :

IDmodèle	Modèle de durée	Modèle de charges	$D_n(M)$	ER_M	C_M	$score\_prox$
glm-glm	GLM log-normal	GLM log-normal	4886976	1.00	0.861	14.33
glm-rf	GLM log-normal	Random Forest	5132274	-1.85	0.811	16.49
glm-svm	log-normal	SVM	1676562	0.867	0.863	15.54
glm-dl	GLM log-normal	Deep Learning	1902423	0.027	0.666	13.81
dl-dl	Deep Learning	Deep Learning	2755093	0.50	0.63	14.705
rf-dl	Random forest	Deep Learning	2966369	0.48	0.60	14.783

TABLE 10.3 – Tableau récapitulatif des performances des modèles

Cette comparaison permet de classer les différents modèles considérés en fonction de leur proximité à la prédiction réalisée par chain ladder. On obtient donc le classement suivant en considérant le critère score.

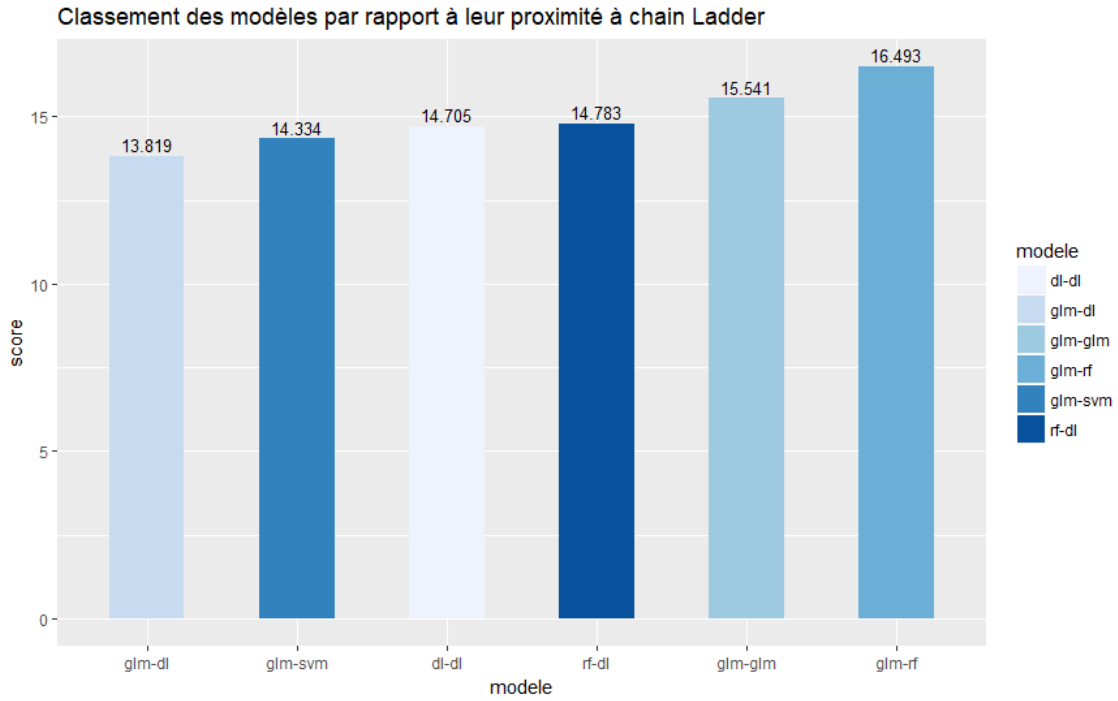


FIGURE 10.2.3 – classement des modèles en fonction de leur proximité à Chain Ladder

Le Deep learning en modèle de charge couplé du glm en modèle de duréee est ici le modèle ayant les résultats les plus proches de Chain Ladder.

10.2.4 Comparaison des temps d'exécutions

Il est aussi possible de comparer les modèles par rapport au temps de calcul pour la prédiction d'une année pour sinistre considéré.
Ces temps d'exécutions doivent être multipliés par le nombre de sinistres à prédire par année

Modèle de durée	Modèle de charges	temps de calcul (sec)
GLM log-normal	GLM log-normal	0.80
GLM log-normal	Random Forest	2.8
GLM log-normal	SVM	0.10
GLM log-normal	Deep Learning	0.88

TABLE 10.4 – Tableau récapitulatif des temps d'exécutions des modèles

de survenance ainsi que par le nombre de prédictions à faire par sinistre. Le graphique 10.2.4 permet de se rendre compte du nombre de sinistres par année de développement.

Le modèle qui permet d'avoir des prédictions les plus rapide est le svm. Cela est principalement dû au fait qu'il s'agit de la seule méthode pour laquelle on utilise pas le package h2o dans la prédiction. En effet ce package permet de calibrer les modèles plus rapidement, cependant on perd du temps au niveau de la prédiction. Aussi, avec ces nombres de sinistre par année de survenance, le temps de calcul pour toute une année atteint facilement 40 minutes par année de survenance pour les modèles tel que le deep-learning et le glm-log normal, contre 5 minutes pour le SVM et 2 heures pour le Random Forest. Notons que le Random Forest doit la longueur de son calcul au fait que 10 prédictions vont constituer la prédiction finale. Ces temps de calculs varient bien évidement en fonction de la puissance des machines utilisées.

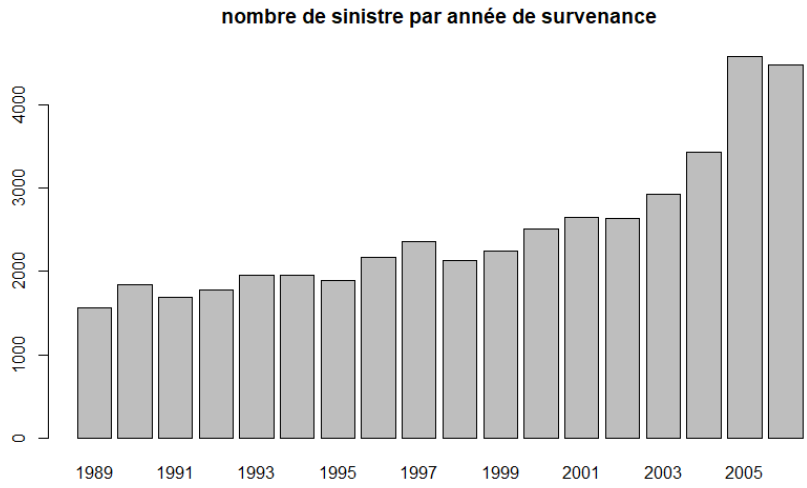


FIGURE 10.2.4 – nombre de sinistre par année de survenance

10.3 Bilan

Les résultats obtenus sont assez satisfaisant. En dépit d'une longue procédure de calcul, les modèles obtenus permettent d'anticiper une évolution cohérente de la charge des sinistres. Ces modèles fournissent une évolution plus ou moins proches des méthodes classiques de Chain Ladder. Entre autre, ces méthodes permettent de s'affranchir des hypothèses restrictives des méthodes usuelles.

Une telle approche du provisionnement non-vie est donc pertinente pour un assureur dans la mesure où elle permet de prévoir les évolutions individuelles des sinistres. En effet elle permettrait d'anticiper par le biais de plusieurs modèles différents, et donc de prévoir plusieurs scénarios de développement possibles. De plus il serait aussi possible d'identifier des profils et groupes homogènes de sinistralité.

Cependant, en ce qui concerne la charge totale de tous les sinistres, la procédure de prédiction étant plus longue que celle des méthodes classiques, ces dernières restent selon nous plus adéquates. Ainsi l'utilisation de ces modèles ligne à ligne pourrait être réservée au cadre des prédictions individuelles.

10.4 Pour aller plus loin

Des modèles par nature de flux

Afin d'améliorer les performances des modèles, il pourrait être intéressant de constituer différents modèles en fonction de la nature des flux dans les charges (charge effective, honoraires, frais, recours etc...)

Des modèles de classe

Une autre manière d'améliorer la modélisation serait éventuellement d'effectuer une classification des profils de sinistres afin d'avoir une meilleure modélisation par groupe homogène de sinistres.

Des modèles de survie

Concernant la durée de vie résiduelle des sinistres nous nous sommes contenté de réaliser des modèles d'apprentissage statistique afin de prédire les durées de vie résiduelle. Cependant cette approche peut être améliorée en considérant des modèles de survie comme ceux de Kaplan-Meier et de Cox. Cette approche permettrait de mieux appréhender la durée de vie résiduelle en matière de probabilité.

Une interface R Shiny

Dans l'optique de rendre l'étude réalisée plus visuelle, et plus accessible, une application R shiny pourrait être utile. En effet, celle ci pourrait permettre à son utilisateur de constituer une provision pour un sinistre donné et une méthode choisie en ayant toujours une comparaison entre modèles. Cela sous-entendrait qu'une base de données de l'entité qui utilise l'application serait téléchargée et mise à jour.

Conclusion

Ce bureau d'étude fût une expérience enrichissante. Nous avons pu effectuer une approche concrète du métier d'actuaire. En effet la prise d'initiatives, le respect des délais et le travail en équipe seront des aspects essentiels de notre futur métier. De plus, il nous a permis d'exploiter les connaissances acquises pendant nos études, autour d'un projet pluridisciplinaire.

Au cours de ce projet, nous avons développé plusieurs méthodes de provisionnement fondées sur des concepts d'apprentissage statistique. Ce projet s'est déroulé en 3 principales étapes. Une première phase de recherche afin de s'immerger dans le sujet qu'est le provisionnement et de comprendre le fonctionnement des principales méthodes usuelles. Cette étape fût suivie d'une phase de modélisation de notre problématique. Enfin nous avons implémenté plusieurs algorithmes permettant de prédire les charges à l'ultime.

Il y a eu plusieurs phénomènes à observer et à modéliser : le montant des charges d'une année à l'autre et la durée de paiements associés. Il ressort de notre analyse que l'utilisation du modèle individuel peut être une alternative intéressante aux modèles les plus classiques. Cette approche individuelle demeure cependant plus complexe et coûteuse en calculs mais elle permet d'avoir des perspectives plus intéressantes dans le sens où les évolutions des charges seraient connues par sinistre.

Les principaux problèmes que nous avons rencontrés concernaient tout d'abord la compréhension des données. La base de données étant très complexe, une phase d'appropriation des données a été nécessaire. Toutefois, si une telle étude devait être complétée, il serait important d'aller plus loin dans la modélisation en scindant la charge en fonction de sa nature (recours, frais, honoraires, etc...) ou encore en affinant les modélisations par des classifications d'individus afin de réaliser des prédictions adaptées à des profils de risque différents.

Bibliographie

[1] F. PLANCHET, "Assurance non vie,le modèle collectif," 2003-2004. support de cours ISFA.

[2] B. RAGGAD, "Fondements de la théorie des valeurs ertêmes,à la gestion des risques du marché pétrolier," 2009. https ://journals.openedition.org/msh/11069 ?file=1.

[3] F. V. Korotoumou TRAORE, "Méthodes de chain ladder et mack."

[4] N. A. D. . G. CHAU, "Mesures de provision cohÉrentes et mÉthodes ligne À ligne pour des risques non-vie," 2012. Mémoire présenté devant l'ENSAE ParisTech pour l'obtention du diplôme de la filière Actuariat et l'admission à l'Institut des Actuaires.

[5] G. BENETEAU, "Modele de provisionnement sur donnees detaillees en assurance non-vie," 2003-2004. www.ica2006.com/Papiers/3022/3022.pdf.

[6] "Prediction intervals for random forests." http ://blog.datadive.net/prediction-intervals-for-random-forests/.

[7] N. Meinshausen, "Quantile regression forests." quantile regression, random forests, adaptive neighborhood regression.

[8] L. Rouvière, "Apprentissage," 2017. Support de cours INSA RENNES.

[9] F. Vermet, "Apprentissage statistique, une apporche connexionniste," 2017. support de cours EURIA.

[10] J. Mary, "Methode d'appentissage avancée : Svm," 2006.

Table des figures

ANNEXES

Annexe 1 : Fonctions et packages utiles

Listes des packages utilisés

nom du package	utilité
sqldf	requêtes sql
rpart	construction d'arbre de décision
randomForest	modélisation en forêt aléatoire
e1071	utilisation de SVM
ggplot2	graphiques
factomineR	ACM
gev	étude des valeurs extrêmes
chainLadder	comparaison avec les packages existants
h2o	Implémentation machine learning
corrplot	étude des correlations
caret	tuning des paramètres

TABLE 10.5 – liste des packages utilisés

Le package H2O

H2O est une plate-forme d'apprentissage automatique et d'analyse prédictive open source, en mémoire distribuée, rapide et évolutive qui permet de construire des modèles d'apprentissage automatique sur des données volumineuses et de faciliter la production de ces modèles. Le package h2o nous a été utile dans le cadre de la modélisation car il permet d'avoir une procédure d'apprentissage rapide. Cependant malgré la rapidité d'ajustement des modèles, on perd en matière de vitesse de prédiction.

Nom de la fonction	Utilité
h2o.init()	initialisation d'un cluster
h2o.shutdown()	fermeture d'un cluster
as.h2o()	conversion d'une table en format h2o
h2o.predict()	prédiction
h2o.randomforest()	ajustement d'un random forest
h2o.glm()	ajustement d'une regression penalisée
h2o.deeplearning()	ajustement d'un deep learning

TABLE 10.6 – Les fonctions utiles de h2o

Annexe 2 : Etude de la distribution des réserves, la méthode de Boostrap

La méthode du Bootstrap est une méthode relativement récente consistant à fabriquer de l'information et à fournir des réponses là où les autres méthodes ne sont pas applicables pour des raison comme le manque d'information, des calculs impossibles etc...Le principe général de la méthode du Bootstrap est le rééchantillonage par replacement. Dans notre étude, nous choisissons d'appliquer la méthode du Bootstrap dans le cadre des hypothèses de Chain Ladder/Mack.

La fonction BootChainLadder

La procédure BootChainLadder fournit une distribution prédictive des réserves ou IBNR pour un triangle cumulatif de développement de sinistres

Principe :

La fonction BootChainLadder utilise une approche de simulation en deux étapes. Dans la première étape, une méthode d'échelle de chaîne ordinaire est appliquée au triangle des charges cumulées. À partir de là, nous calculons les résidus de Pearson mis à l'échelle que nous avons bootstrapés R fois pour prévoir les futurs paiements incrémentiels de sinistres par la méthode standard de l'échelle de chaîne [4]

Implémentation de la fonction BootChainLadder et résultats obtenus sous R

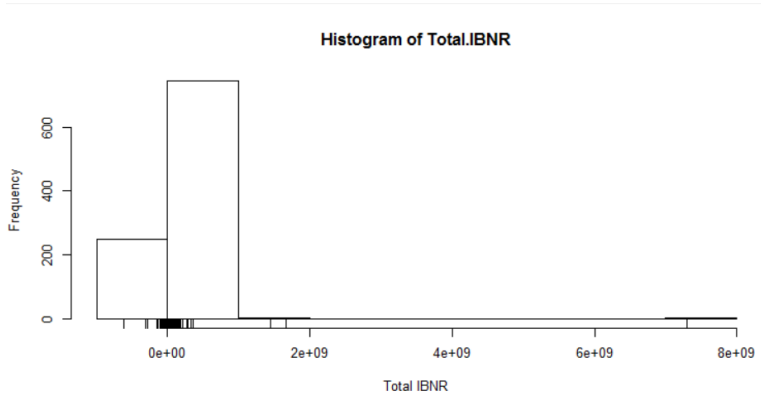


FIGURE 10.4.1 – Histogramme des réserves

La figure ci-dessus représente la fonction répartition de la distribution des réserves qui est nulle pour IBNR <0, croit progressivement et se stabilise en la valeur 1.

4. A chaque itération du Bootstrap, le triangle de résidus de Pearson est ré-échantillonné : pour chaque cellule un tirage aléatoire avec remise est effectué parmi les valeurs des résidus non exclus. De ce fait, une même valeur peut se retrouver plusieurs fois dans le triangle. De nouveaux triangles supérieurs de montants incrémentaux sont alors calculés à chaque itération. Pour chaque itération, à partir du triangle incrémental supérieur, il est possible de calculer le triangle cumulé supérieur correspondant et de le développer en utilisant la méthode classique de Chain Ladder. Ceci permet alors d'obtenir un vecteur de réserves pour chaque année ainsi que la réserve totale pour chaque itération

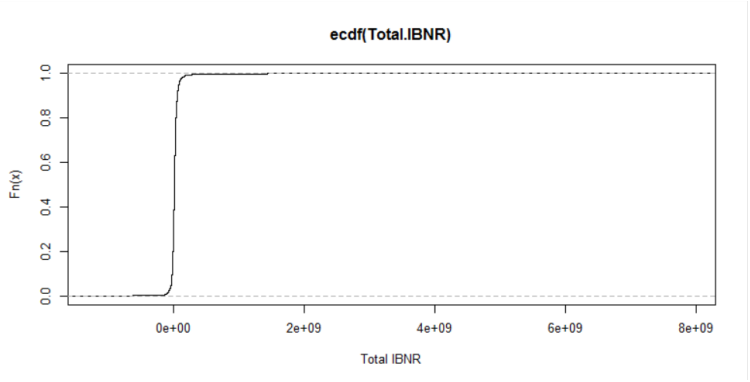


FIGURE 10.4.2 – Distribution des réserves

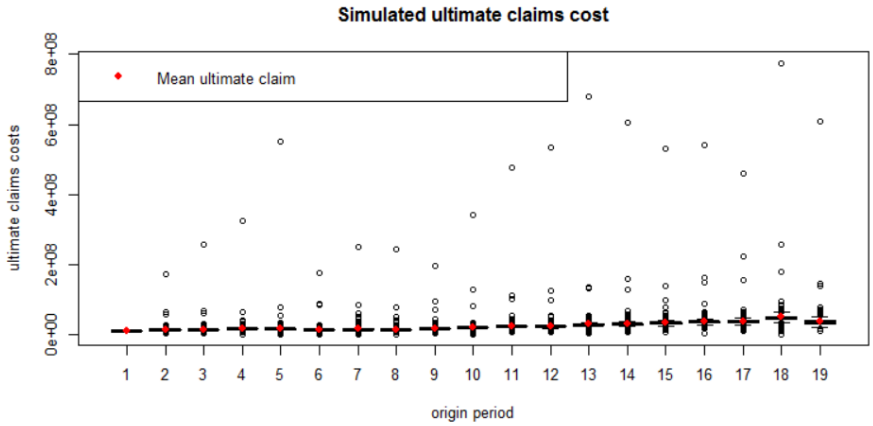


FIGURE 10.4.3 – Boxplot de la chargé à l'ultime estimée

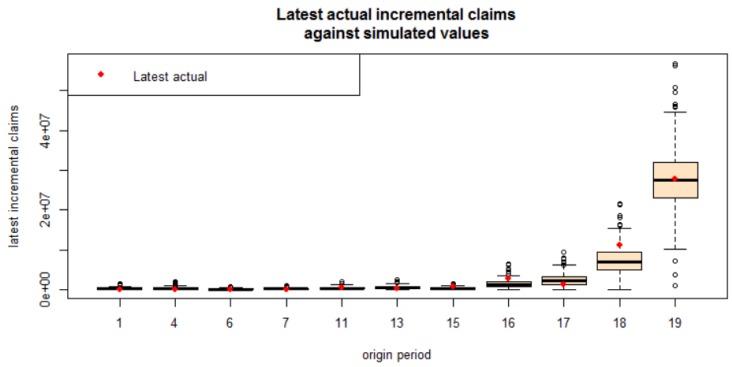


FIGURE 10.4.4 – Boxplot du montant de la réserve

Modélisation de la loi des réserves

Ajustement d'une loi log-normale

Rappelons qu'en théorie des probabilités et statistique, on dit qu'une variable aléatoire X suit une loi log-normale de paramètres (μ, σ^2) si la variable $Y = \ln(X)$ suit une loi normale d'espérance μ et de variance σ^2.

Aussi, pour cet ajustement, on utilise le maximum de vraisemblance des distributions univariées, implémenté dans la fonction fitdistr du package MASS disponible sur R. On obtient ainsi une estimation des paramètres et un intervalle de confiance associé.

Validation du modèle

On peut aussi vérifier l'ajustement en utilisant le graphe des quantile-quantile-plot(qqplot) qui compare les quantiles théorique au quantile approché :

Paramètres estimés	valeur	borne 2.5%	borne 97.5%
moyenne	16.8752	16.7974	19.9596
variance	1.3248	1.0751	1.1898

TABLE 10.7 – estimation et intervalles de confiance des paramètres

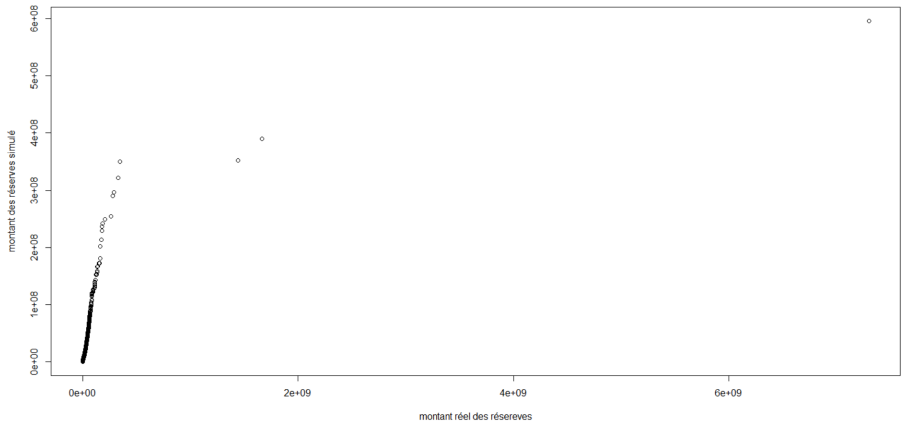


FIGURE 10.4.5 – qqplot

Sur le graphe ci dessus, on peut remarquer que la plupart des points sont approximativement alignés. Cet alignement permet de justifier la modélisation des réserves par la loi log-normale.

La value at risk des réserves

Soit X une variable aléatoire et p un niveau de probabilité,ainsi la La value at risk VaR(X,p) est définie comme étant le quantile d'ordre p de X.

$$VaR(X,p) = \inf[x \in R | P[X \leq x] \geq p] \tag{10.6}$$

	IBNR 75%	IBNR 95%	IBNR 99%	IBNR 99.5%
1	0.00	0	0	0
2	58858.34	2512450	8668389	14219832
3	195413.90	2932195	9102184	11804369
4	500844.69	3910242	10442700	15522112
5	837544.13	4379682	9865015	13042276
6	503741.47	3802042	7827377	9796628
7	933794.74	4599391	10592861	17525928
8	860222.39	4385738	9489099	13308746
9	1128123.31	4793991	10332857	15532362
10	1021066.49	5251088	9367955	12885830
11	1686575.24	6173766	12337781	21577008
12	1564063.60	6627881	14272242	26455913
13	1489116.18	6599158	13855354	17687190
14	1502342.45	6894971	13732324	19144644
15	2125631.90	8128502	18394786	29472890
16	2536720.41	9812187	18902788	26422388
17	3691456.67	11125151	19999803	25664796
18	7938326.04	16565885	25574506	38101500
19	12119276.58	20950188	34169223	42419966

FIGURE 10.4.6 – VaR des réserves par année de survenance

Tableau de calcul des quantiles à différents niveaux

	Totals
IBNR 75%:	34271181
IBNR 95%:	85255945
IBNR 99%:	181153721
IBNR 99.5%:	288724428

FIGURE 10.4.7 – VaR de la réserve globale

Le tableau ci dessus permet de visualiser l'évolution de la value at risk pour chaque année de survenance et en fonction du niveau de confiance.

On peut remarquer que la VaR au niveau de confiance 99,5% est de 288 724 428 € donc de 1988 à 2006, la compagnie d'assurance a 99,5% de chance d'avoir des encours futurs globaux inférieurs à 288 724 428 € et 0.5% de chance de dépasser cette VaR, ce qui nous semble tout à fait logique puis que le montant total de la réserve donnée par la méthode de Chain Ladder est de 301 256 138 € qui est bien supérieure à 288 724 428 € .

Annexe 3 : CODES

Mise en forme préliminaire

```
#Récuperation des données utiles
setwd("C:/Users/andrea/Desktop/EURIA/BE")
Data = readRDS("base_sinistres_indiv.rds")

# utilisation du package sqldf qui permet
# d'effectuer des requêtes sql dan R
library(sqldf)
library(MASS)

X1 = Data
colnames(X1)[3] = "numsin"
colnames(X1)[16] = "blesse"
colnames(X1)[26] = "dureeSin"
colnames(X1)[9] = "type_repar"
colnames(X1)[5] = "date_declar"

X1 = sqldf('select *,charge as Charges
            from X1
            order by numsin')
```

Étude statistiques des variables

variables

```
setwd("C:/Users/andrea/Desktop/EURIA/BE")
Data = readRDS("base_sinistres_indiv.rds")

library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.4.2
```

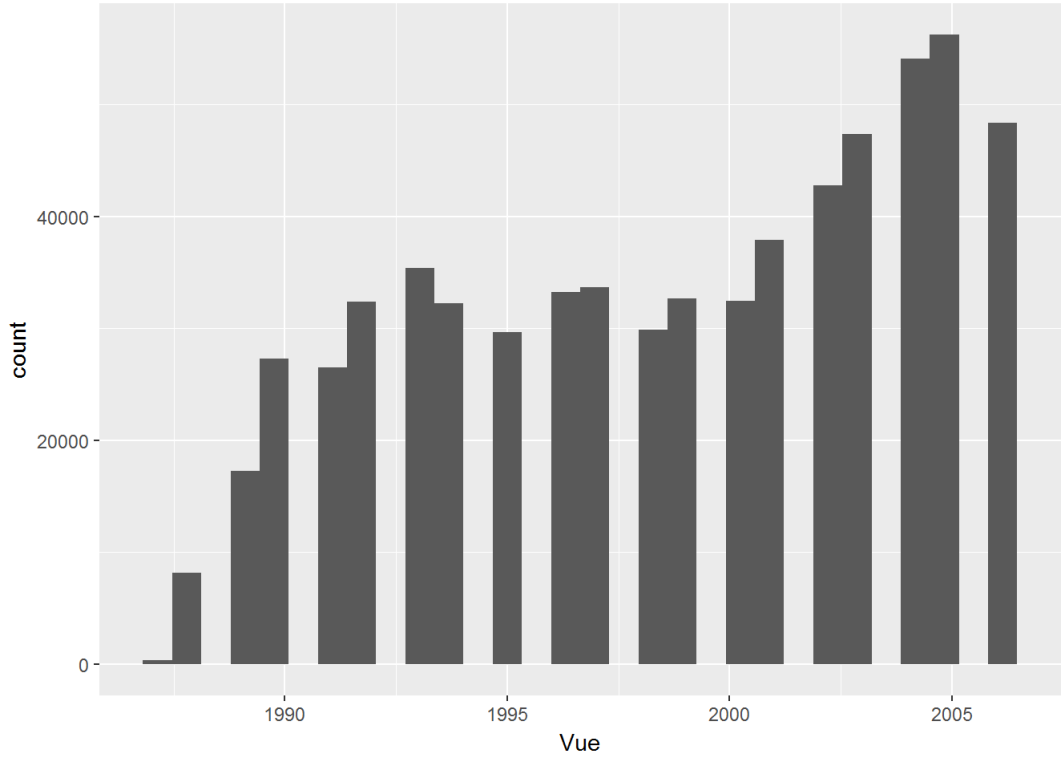
```
str(Data)
```

```
## 'data.frame':    658050 obs. of  29 variables:
##  $ Vue             : num  1988 1988 1988 1988 1988 ...
##  $ Fin_Vue         : num  1988 1988 1988 1988 1988 ...
##  $ Numéro_sinistre : chr  "AW.YAC3MI0KP74207317" "AW.YAC3MI0KP74207317" "AW.YAC3MI0KP74207317" "AW.YAC
3MI0KP74207317" ...
##  $ Date_de_surv    : Date, format: "1988-10-30" "1988-10-30" ...
##  $ Date_de_dÃ©cla  : Date, format: "1988-11-01" "1988-11-01" ...
##  $ Etat            : chr  "Ouvert" "Ouvert" "Ouvert" "Ouvert" ...
##  $ Charge          : num  7185 0 7263 15551 0 ...
##  $ Type_de_sinistre: chr  "Accid. 2veh" "Accid. 2veh" "Accid. 2veh" "Accid. 2veh" ...
##  $ Type_rÃ©par     : chr  "X" "X" "X" "X" ...
##  $ Permis          : chr  "A" "A" "A" "A" ...
##  $ Tiers_acc       : chr  "NON" "NON" "NON" "NON" ...
##  $ Tiers_domm      : chr  "NON" "NON" "NON" "NON" ...
##  $ Constat         : chr  "NON" "NON" "NON" "NON" ...
##  $ Police          : chr  "OUI" "OUI" "OUI" "OUI" ...
##  $ Chargement_veh  : chr  "Lourd" "Lourd" "Lourd" "Lourd" ...
##  $ BlessÃ©s        : chr  "OUI" "OUI" "OUI" "OUI" ...
##  $ Usage           : chr  "Loisir" "Loisir" "Loisir" "Loisir" ...
##  $ Type_acc        : chr  "Autre" "Autre" "Autre" "Autre" ...
##  $ Type_m          : chr  "Tech" "Tech" "Tech" "Tech" ...
##  $ Assur           : chr  "X" "X" "X" "X" ...
##  $ Prop            : chr  "X" "X" "X" "X" ...
##  $ Exp             : chr  "X" "X" "X" "X" ...
##  $ Expa            : chr  "X" "X" "X" "X" ...
##  $ Taux1           : chr  "X" "X" "X" "X" ...
##  $ Taux2           : chr  "X" "X" "X" "X" ...
##  $ DurÃ©e_sin      : chr  "4" "4" "4" "4" ...
##  $ Taux3           : chr  "100" "100" "100" "100" ...
##  $ Degat           : chr  "X" "X" "X" "X" ...
##  $ Lieu            : chr  "Ville" "Ville" "Ville" "Ville" ...
```

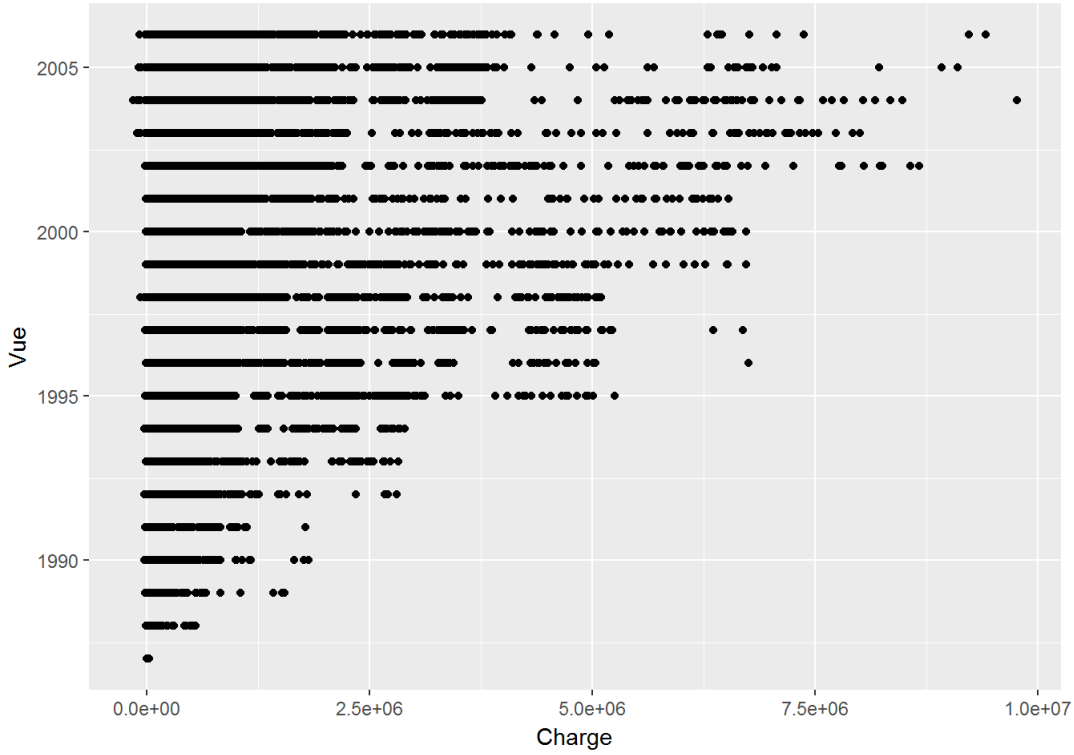
```
colnames(Data)[3] = "numsin"
colnames(Data)[5] = "date_declar"
colnames(Data)[9] = "type_repar"
colnames(Data)[16] = "blesse"
colnames(Data)[26] = "dureeSin"
```

```
ggplot(data =Data ,aes(x=Vue),binwidth =50)+
    geom_histogram()
```

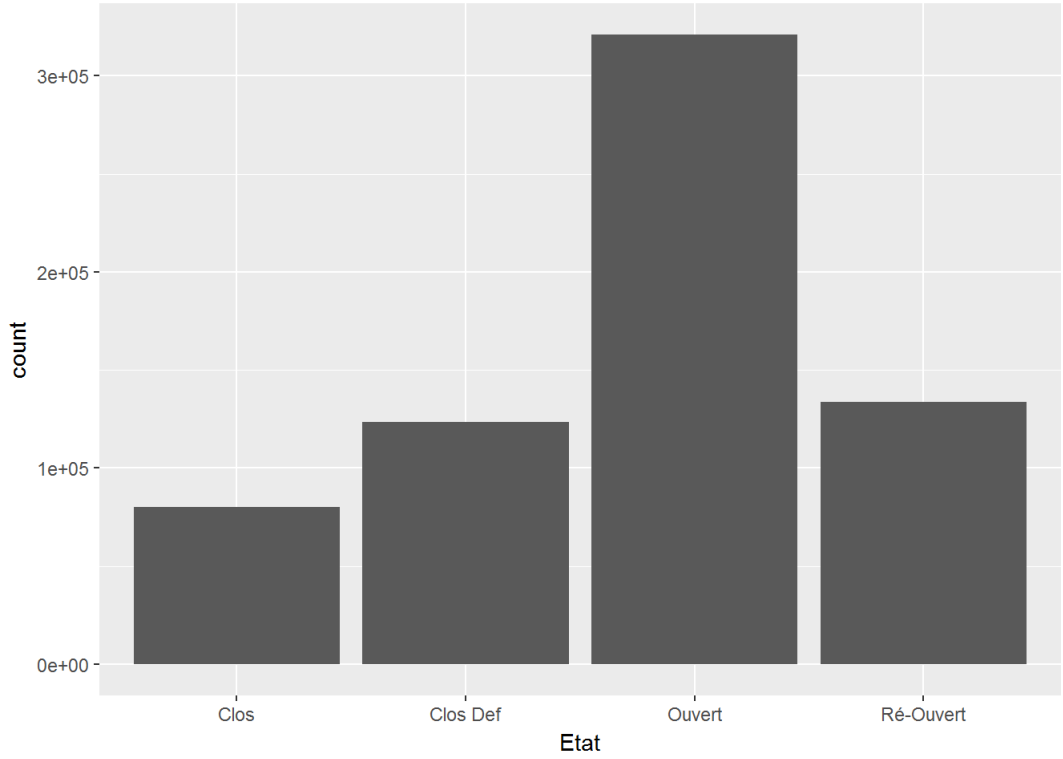
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



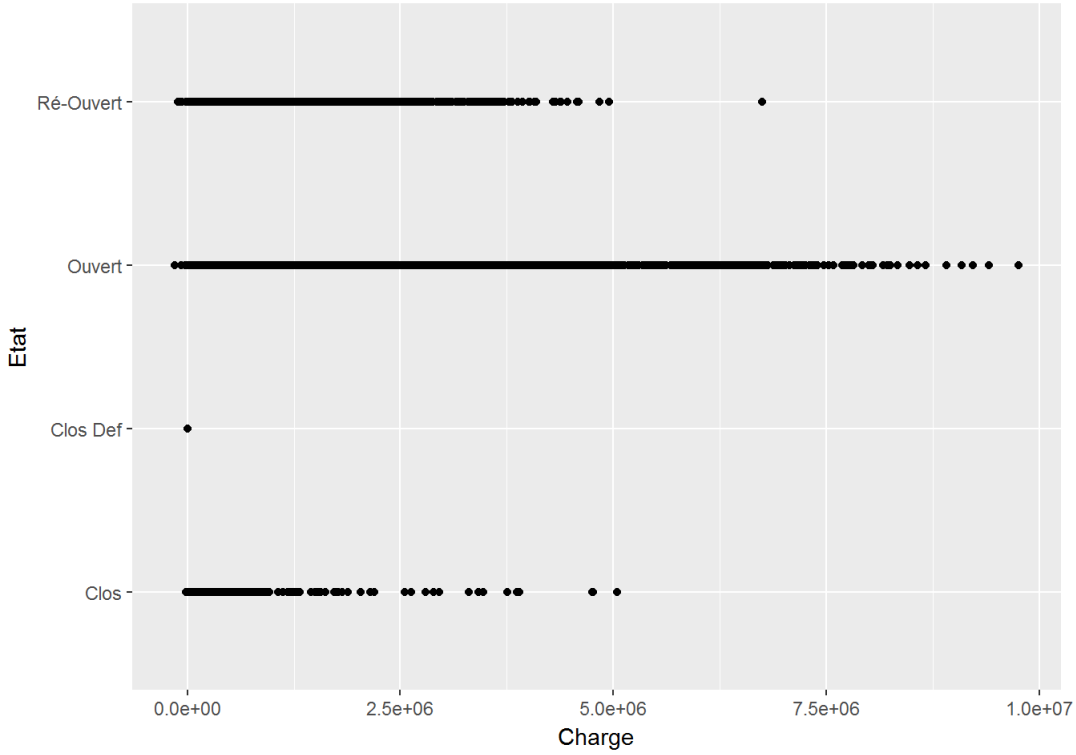

```
ggplot(data =Data ,aes(x=Charge,y=Vue))+
    geom_point()
```



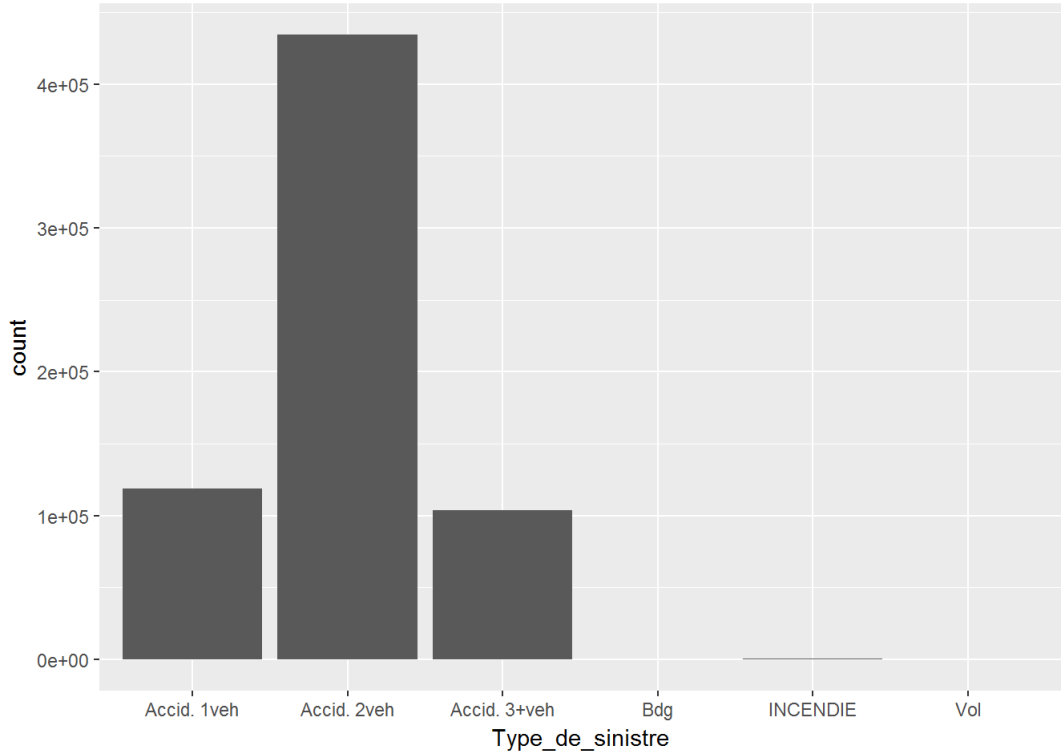

```
par(mfrow=c(1,2))
ggplot(data =Data ,aes(x=Etat),binwidth =50)+
    geom_bar()
```



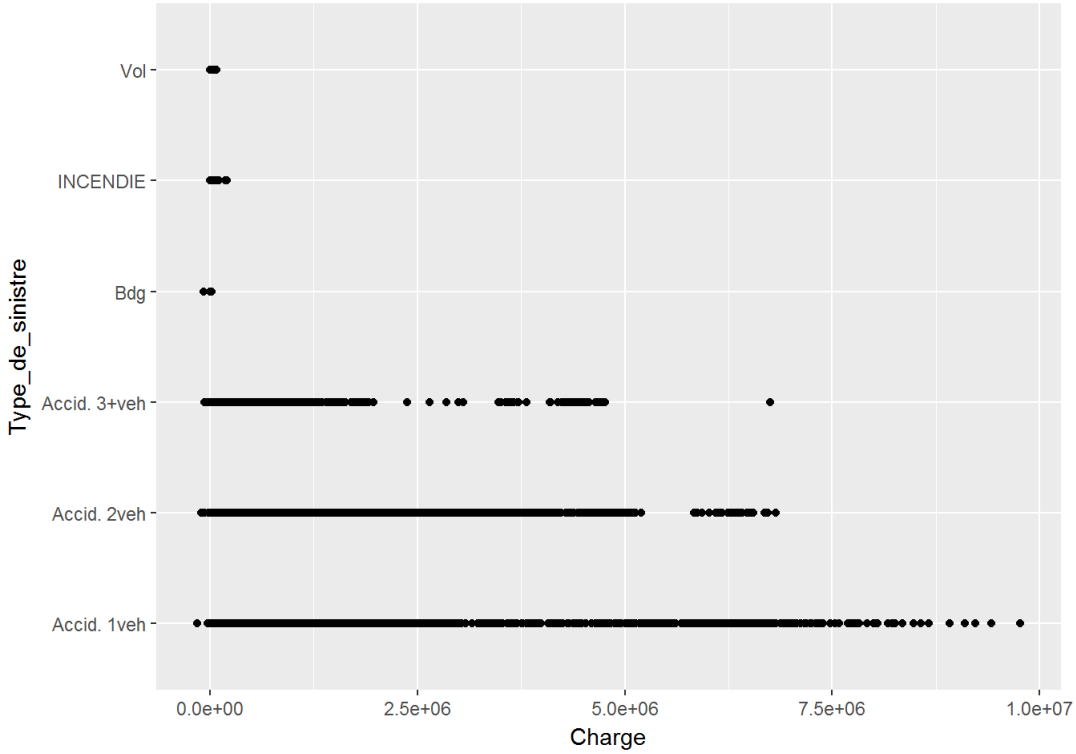

```
ggplot(data =Data ,aes(x=Charge,y=Etat))+
    geom_point()
```



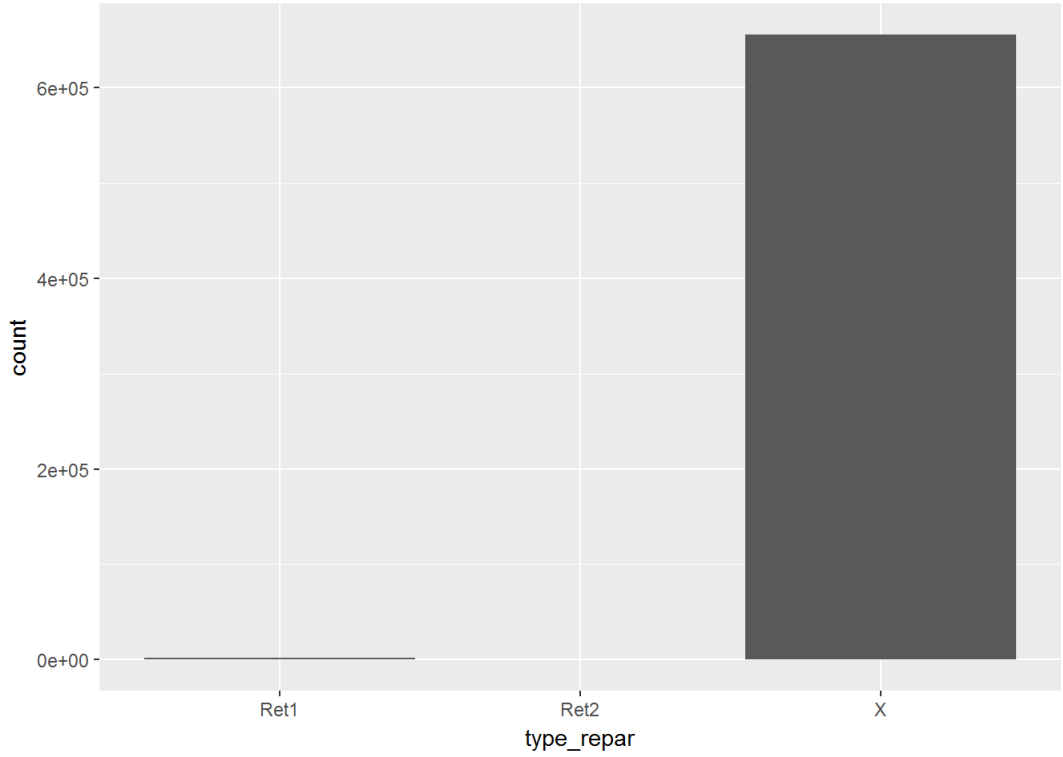

```
ggplot(data =Data ,aes(x=Type_de_sinistre),binwidth =50)+
    geom_bar()
```



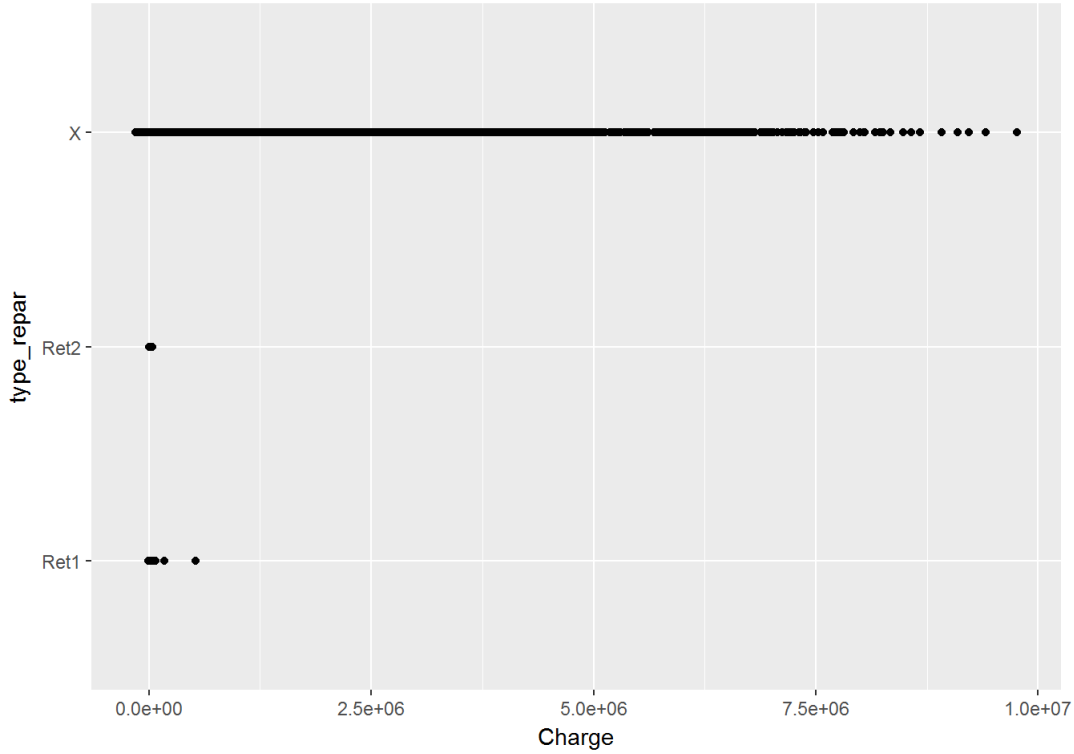

```
ggplot(data =Data ,aes(x=Charge,y=Type_de_sinistre))+
    geom_point()
```



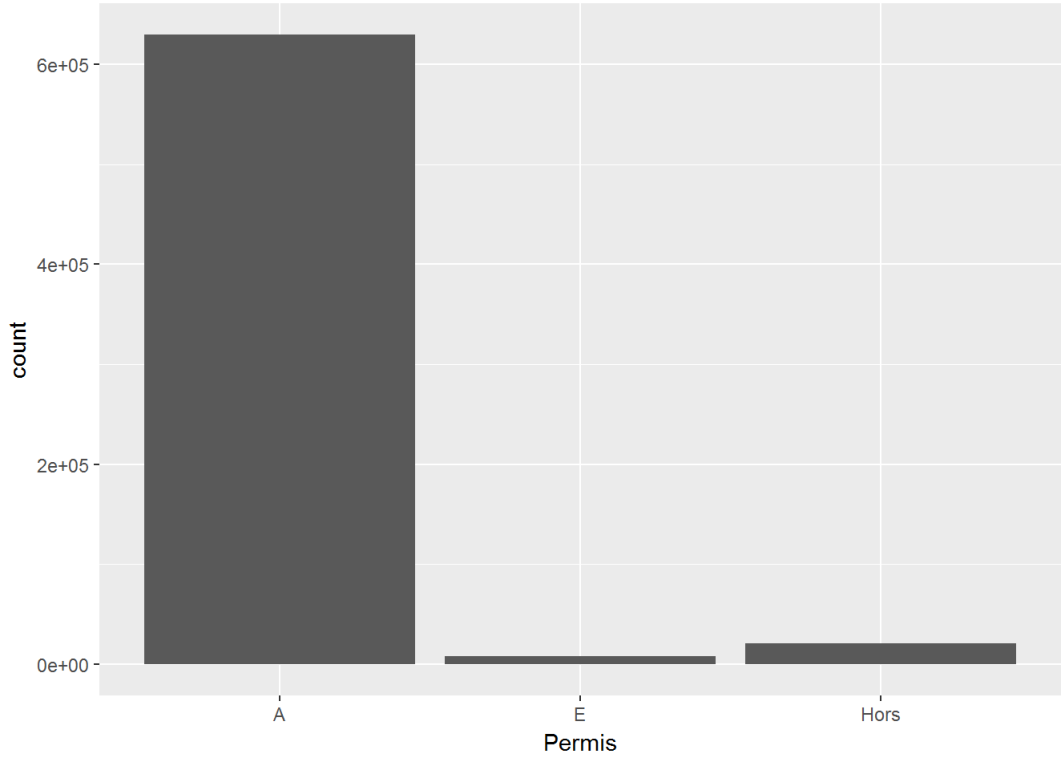

```
ggplot(data =Data ,aes(x=type_repar),binwidth =50)+
    geom_bar()
```



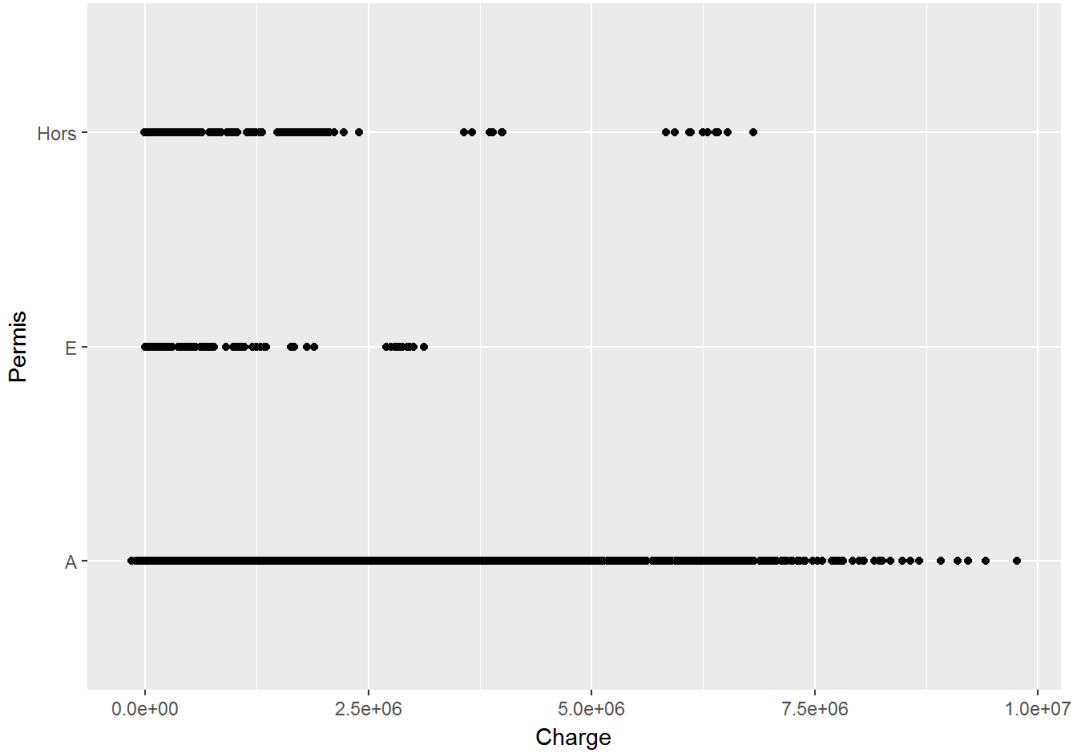

```
ggplot(data =Data ,aes(x=Charge,y=type_repar))+
    geom_point()
```



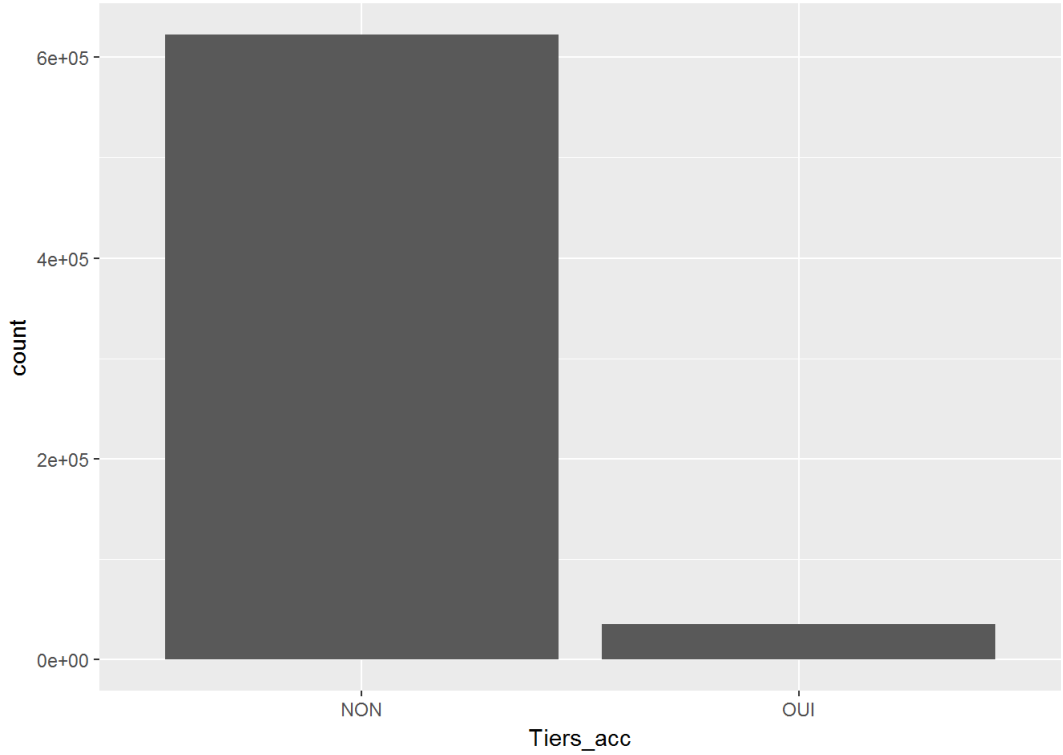

```
ggplot(data =Data ,aes(x=Permis),binwidth =50)+
    geom_bar()
```



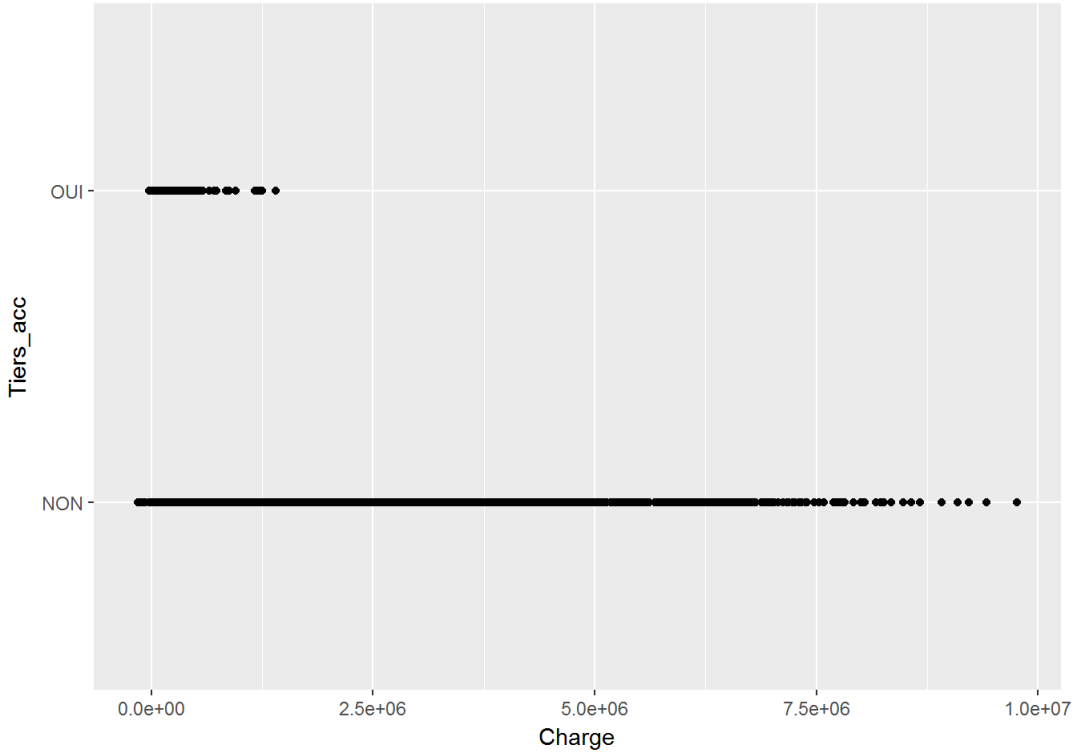

```
ggplot(data =Data ,aes(x=Charge,y=Permis))+
    geom_point()
```



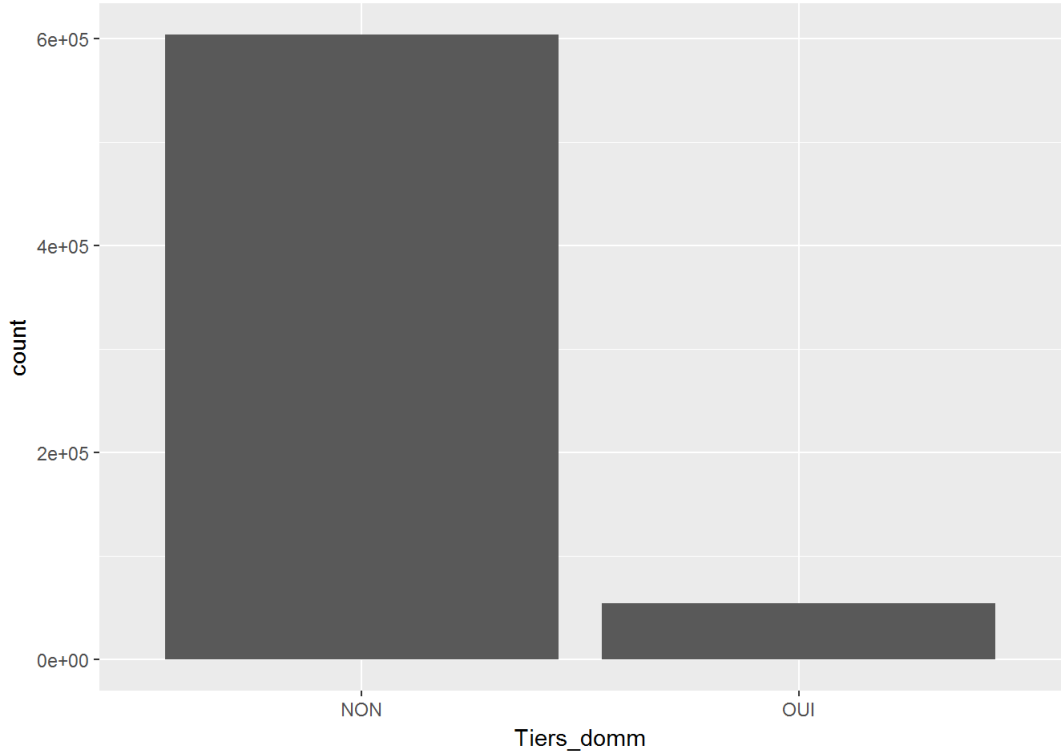

```
ggplot(data =Data ,aes(x=Tiers_acc),binwidth =50)+
    geom_bar()
```



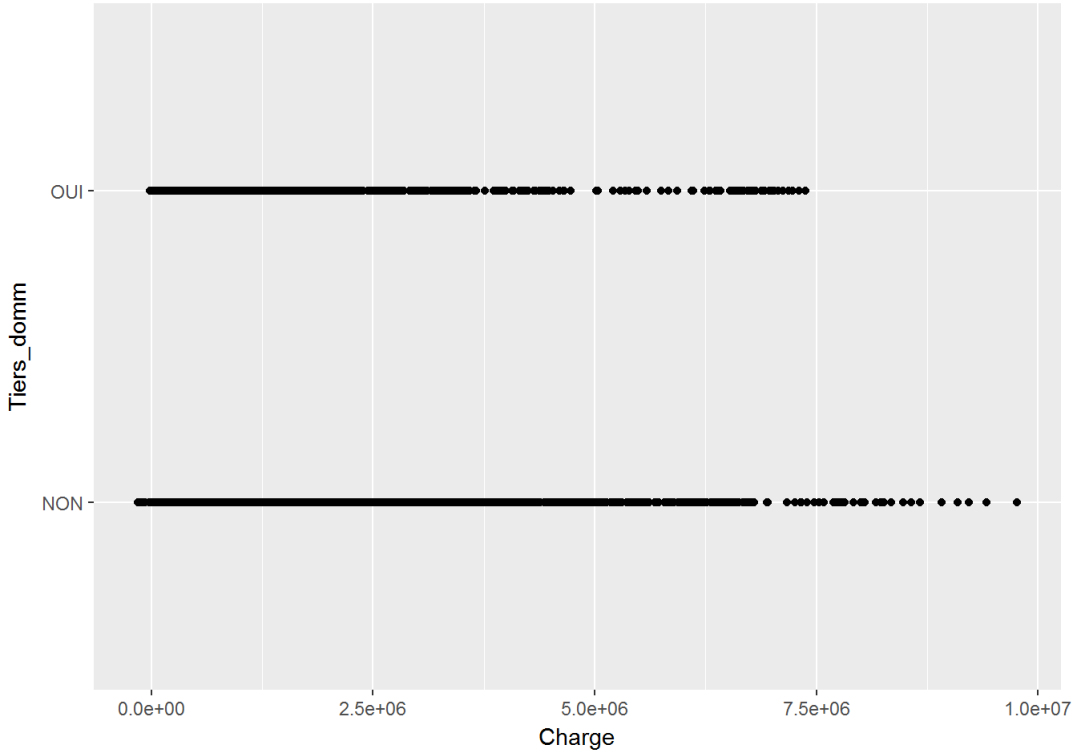

```
ggplot(data =Data ,aes(x=Charge,y=Tiers_acc))+
    geom_point()
```



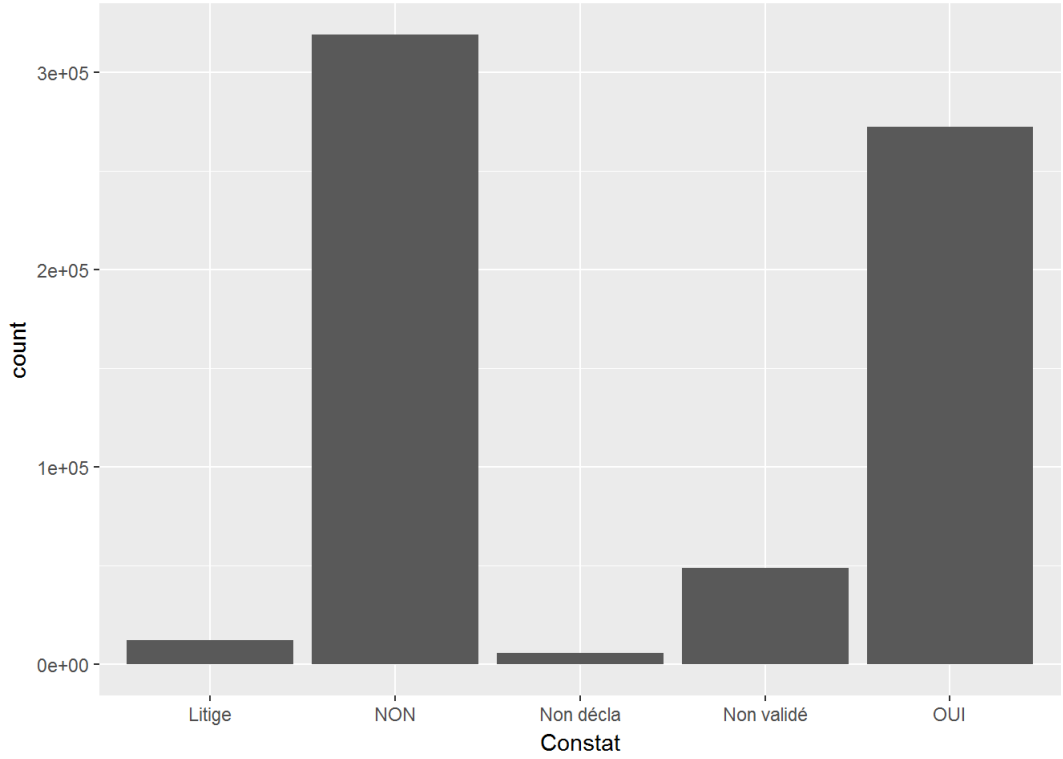

```
ggplot(data =Data ,aes(x=Tiers_domm),binwidth =50)+
    geom_bar()
```



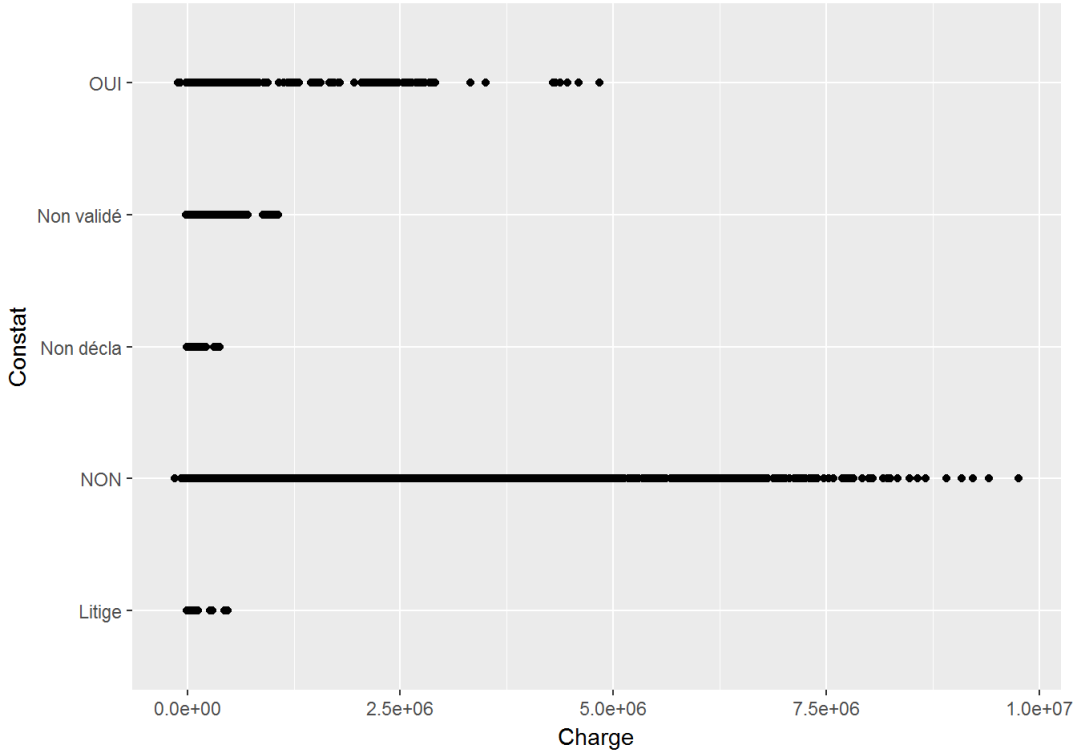

```
ggplot(data =Data ,aes(x=Charge,y=Tiers_domm))+
    geom_point()
```



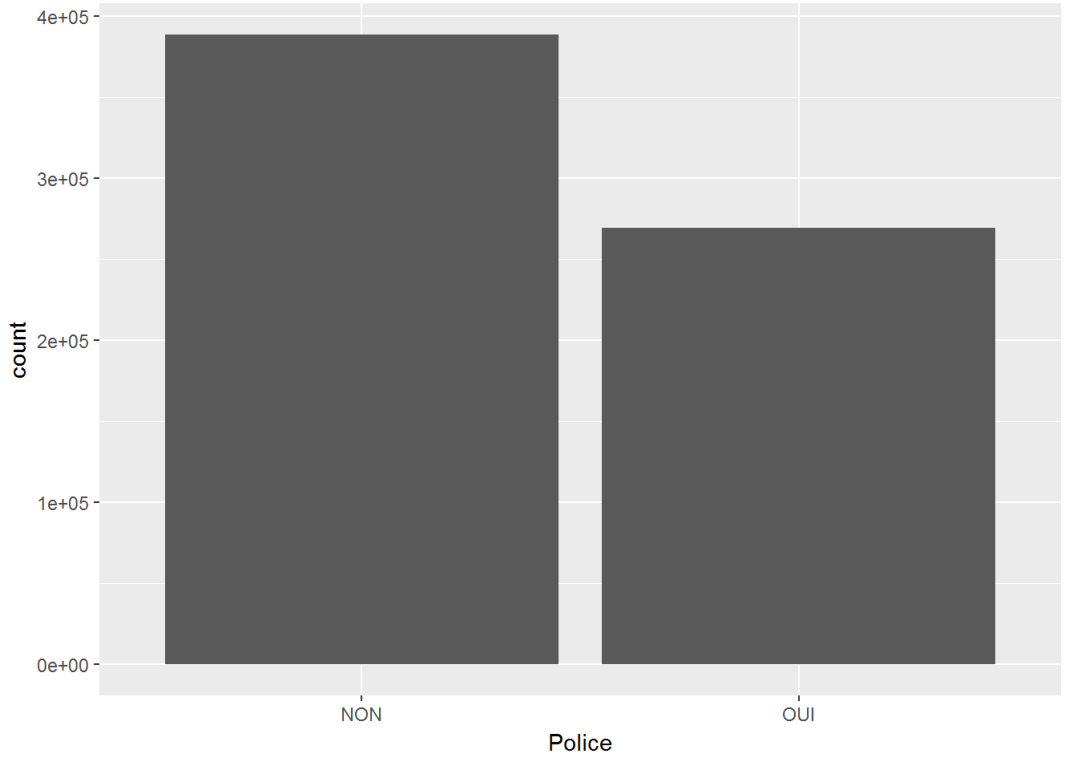

```
ggplot(data =Data ,aes(x=Constat),binwidth =50)+
    geom_bar()
```



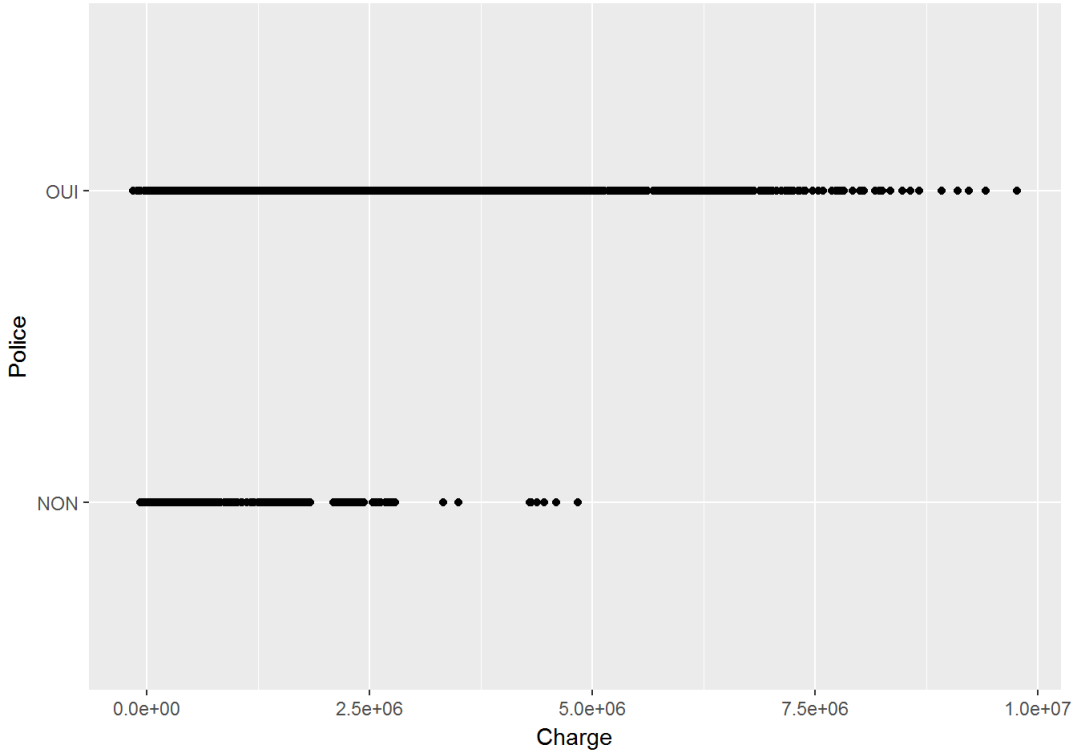

```
ggplot(data =Data ,aes(x=Charge,y=Constat))+
    geom_point()
```



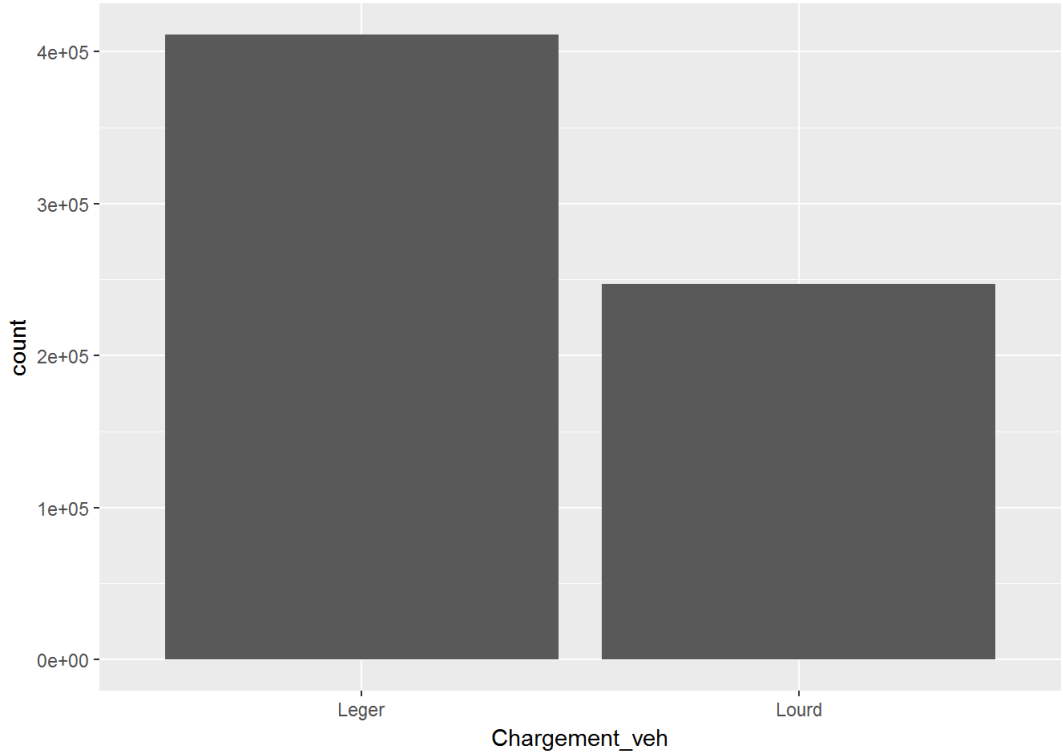

```
ggplot(data =Data ,aes(x=Police),binwidth =50)+
    geom_bar()
```



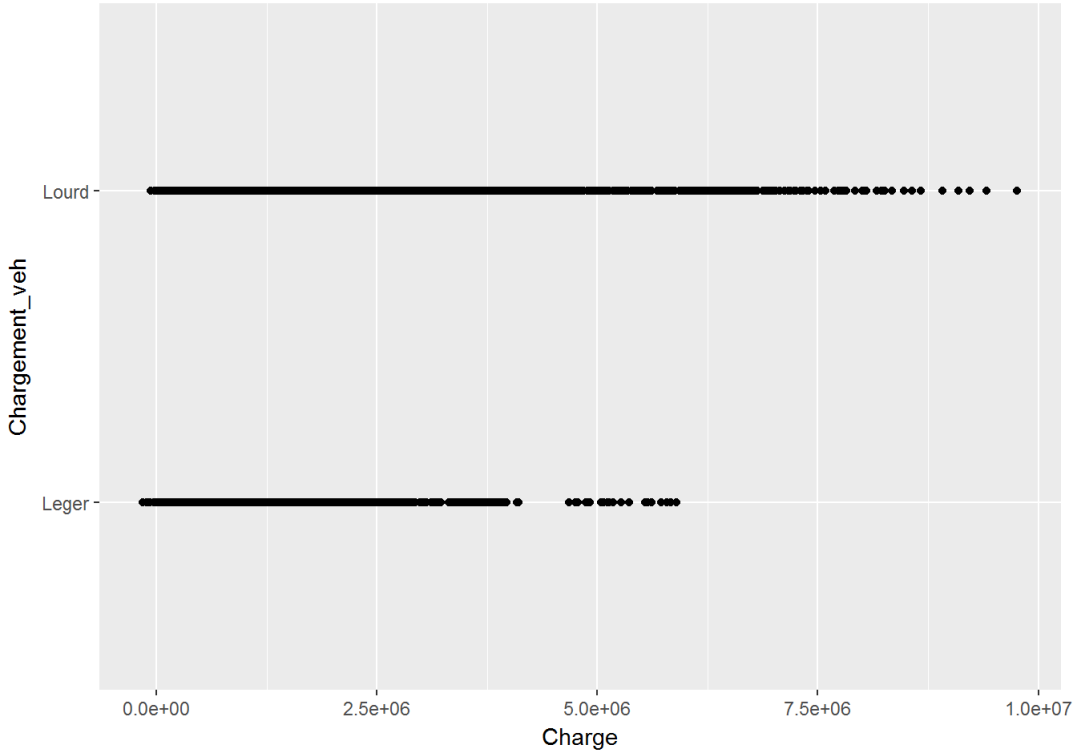

```
ggplot(data =Data ,aes(x=Charge,y=Police))+
    geom_point()
```



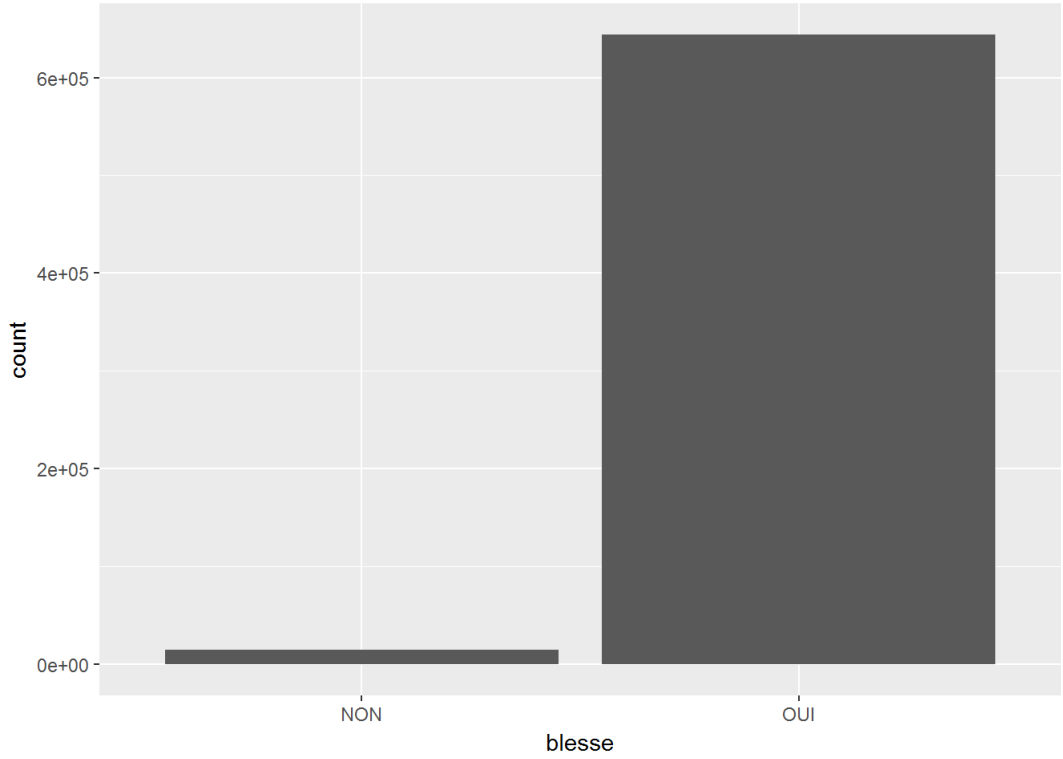

```
ggplot(data =Data ,aes(x=Chargement_veh),binwidth =50)+
    geom_bar()
```



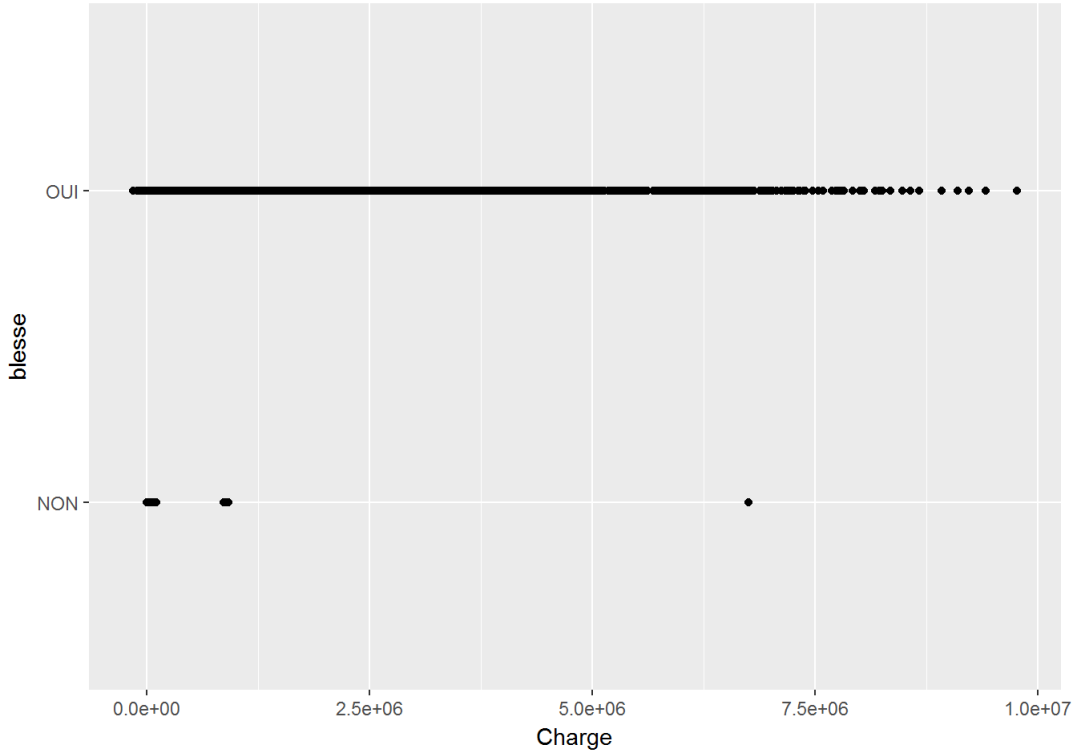

```
ggplot(data =Data ,aes(x=Charge,y=Chargement_veh))+
    geom_point()
```



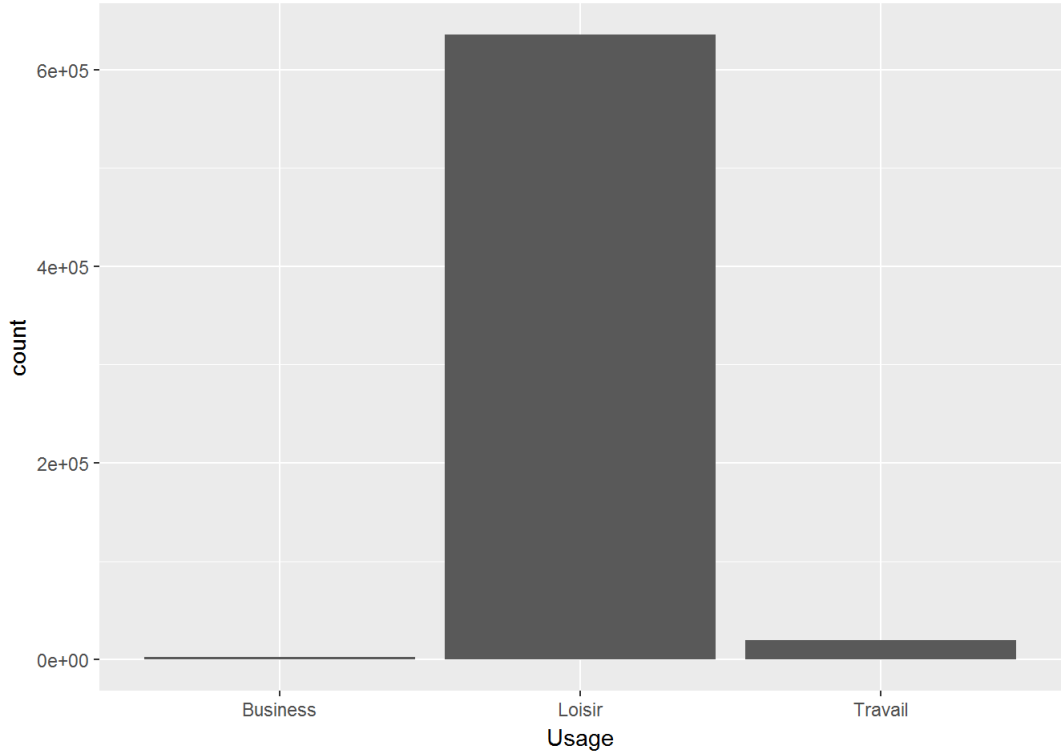

```
ggplot(data =Data ,aes(x=blesse),binwidth =50)+
     geom_bar()
```



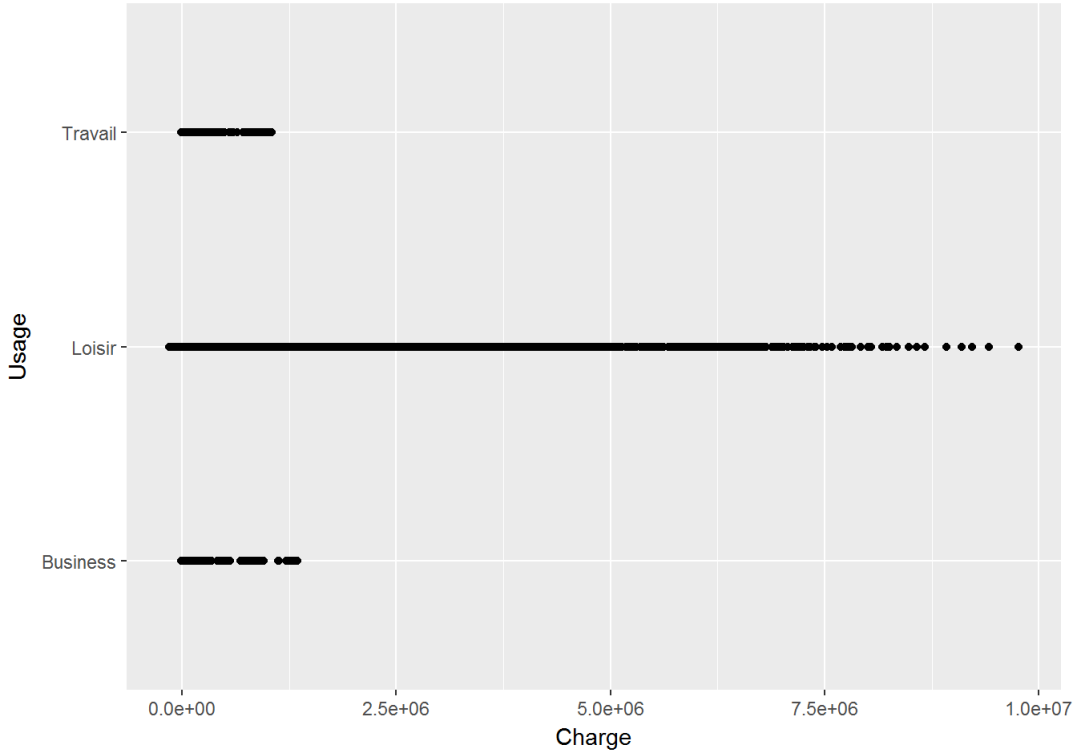

```
ggplot(data =Data ,aes(x=Charge,y=blesse))+
    geom_point()
```



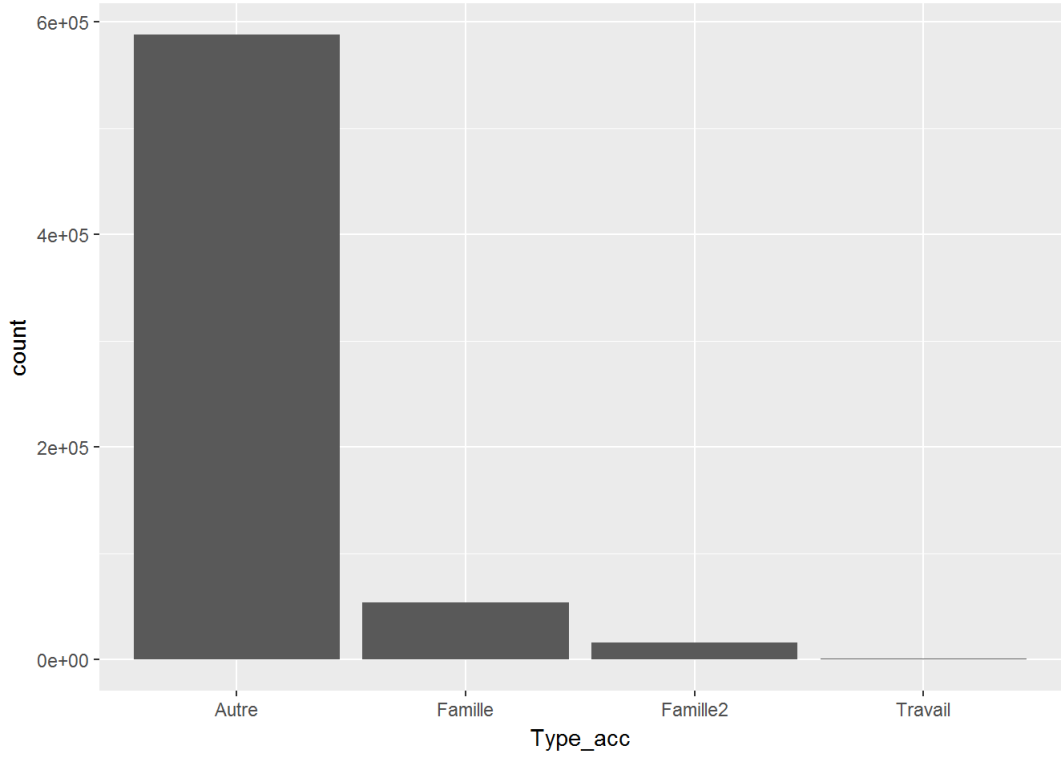

```
ggplot(data =Data ,aes(x=Usage),binwidth =50)+
    geom_bar()
```



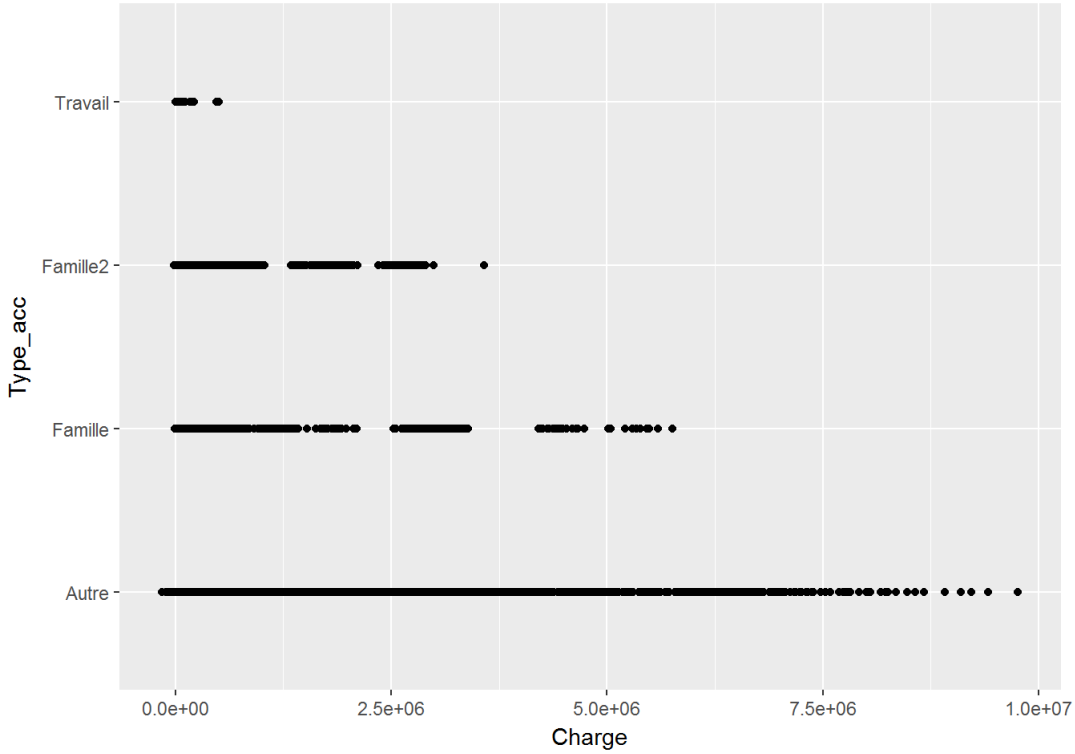

```
ggplot(data =Data ,aes(x=Charge,y=Usage))+
    geom_point()
```



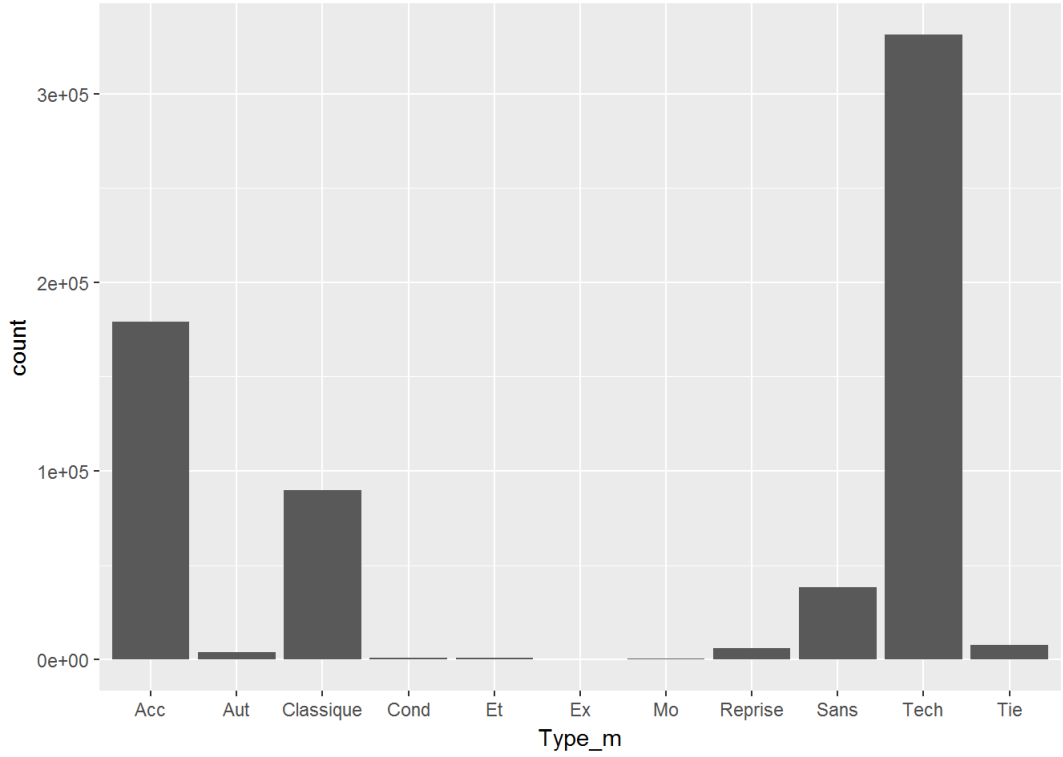

```
ggplot(data =Data ,aes(x=Type_acc),binwidth =50)+
    geom_bar()
```



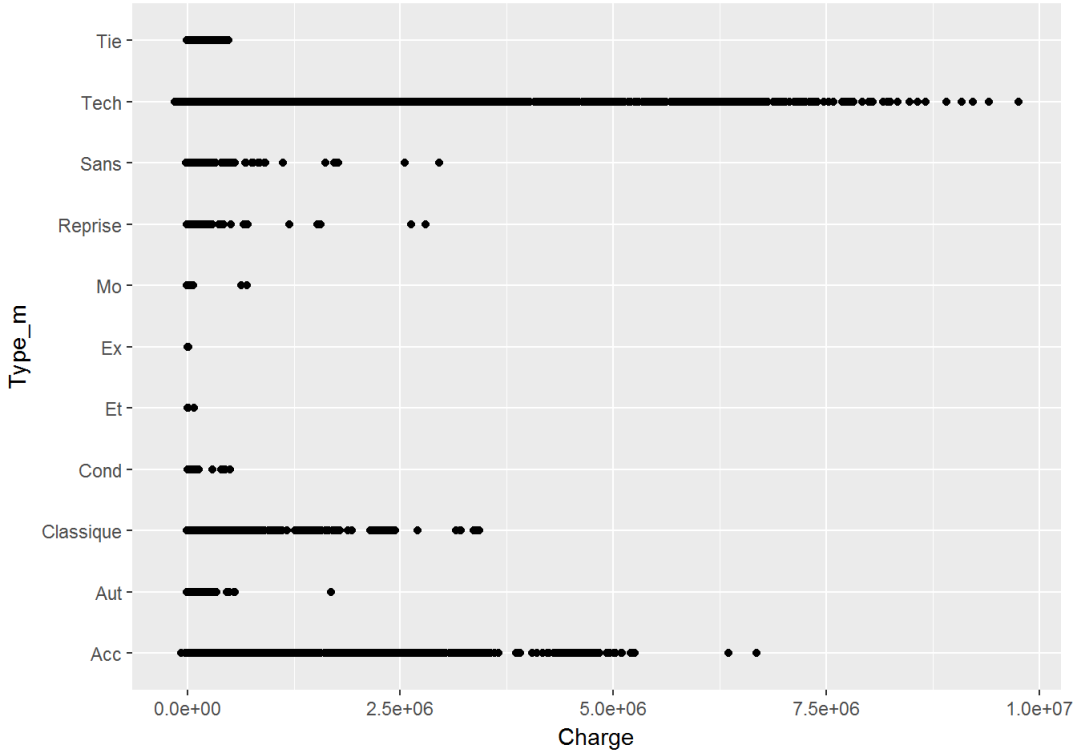

```
ggplot(data =Data ,aes(x=Charge,y=Type_acc))+
    geom_point()
```



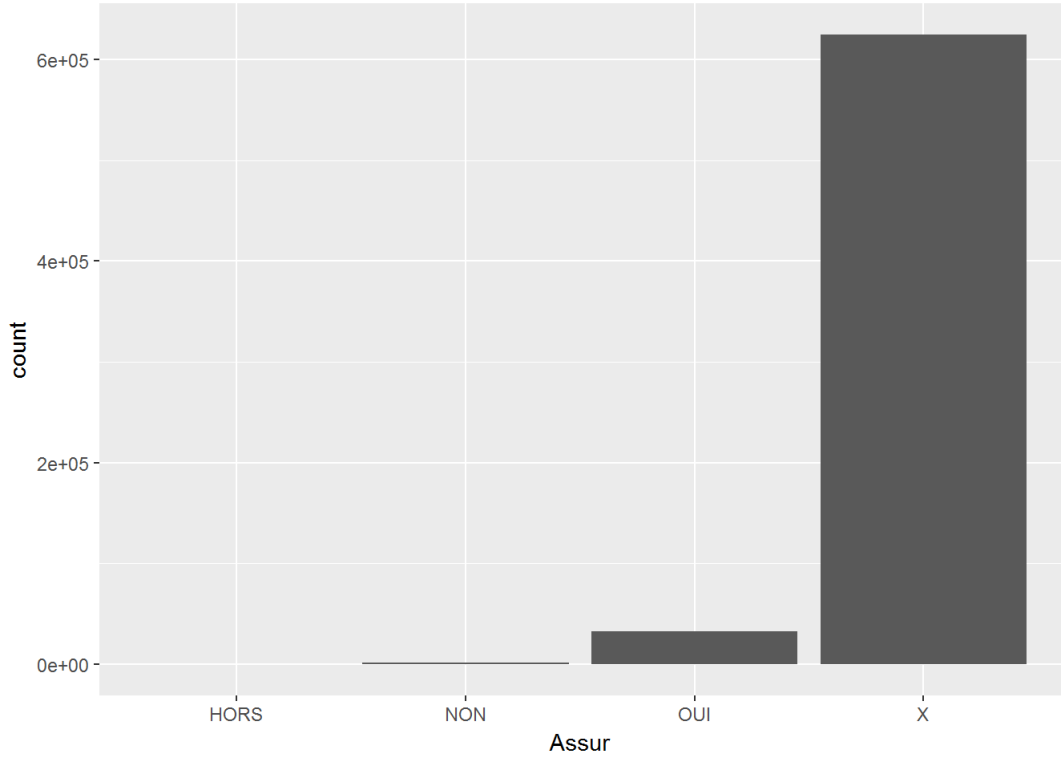

```
ggplot(data =Data ,aes(x=Type_m),binwidth =50)+
    geom_bar()
```



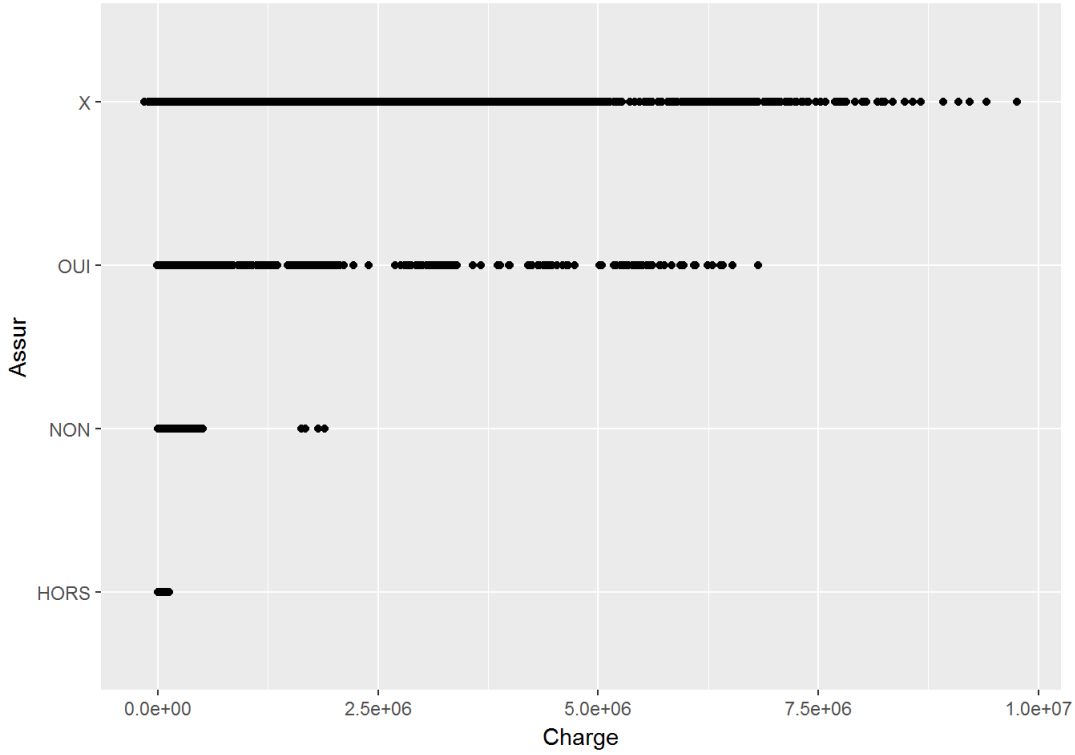

```
ggplot(data =Data ,aes(x=Charge,y=Type_m))+
    geom_point()
```



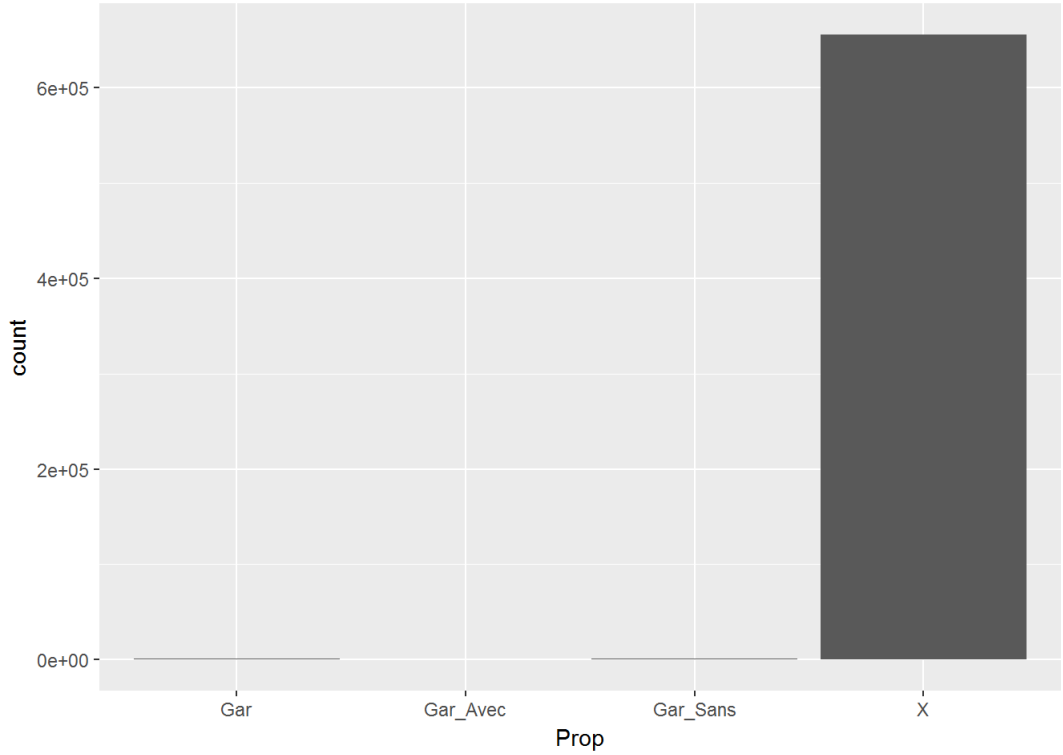

```
ggplot(data =Data ,aes(x=Assur),binwidth =50)+
    geom_bar()
```



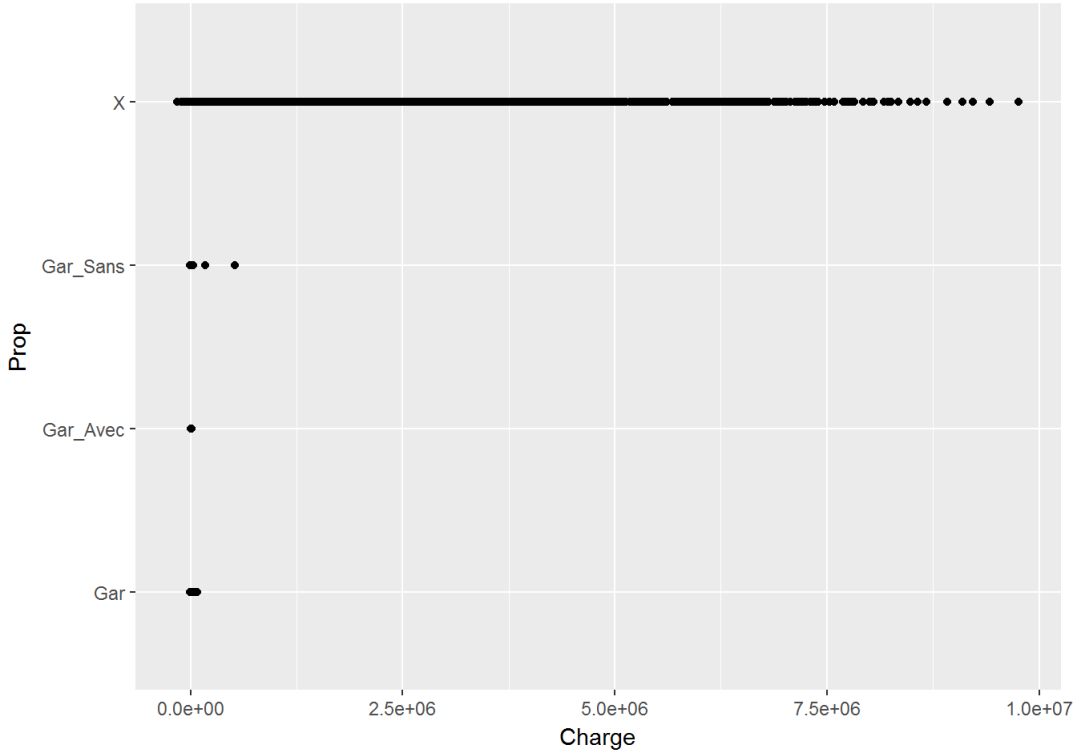

```
ggplot(data =Data ,aes(x=Charge,y=Assur))+
    geom_point()
```



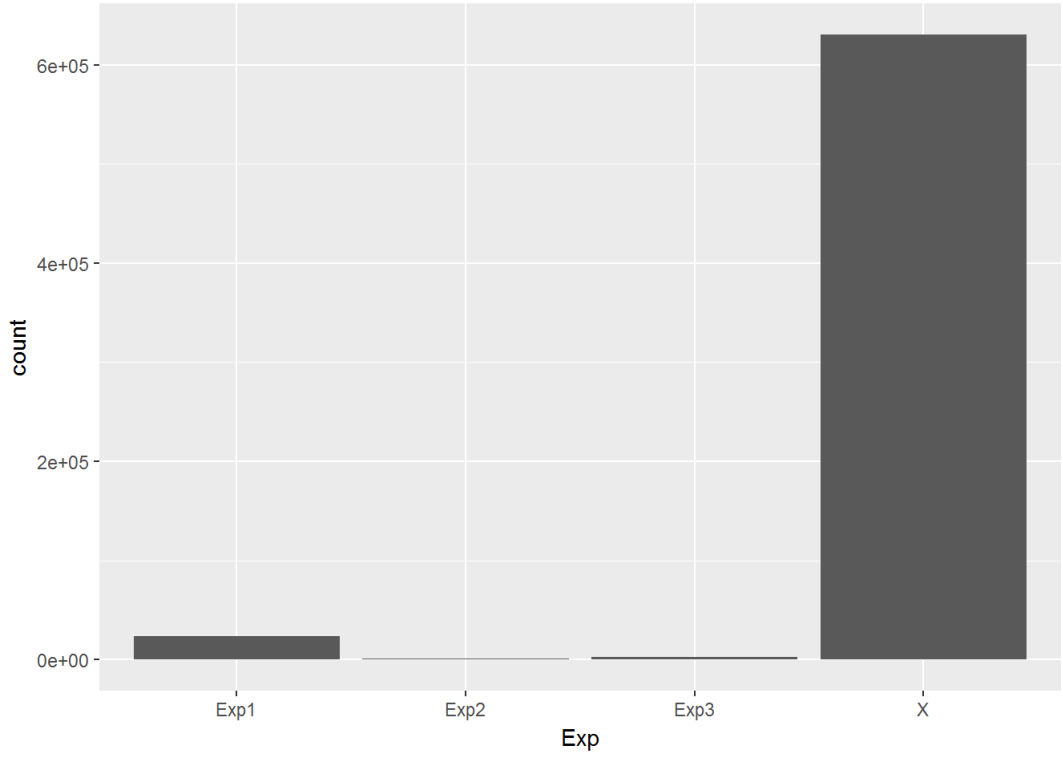

```
ggplot(data =Data ,aes(x=Prop),binwidth =50)+
    geom_bar()
```



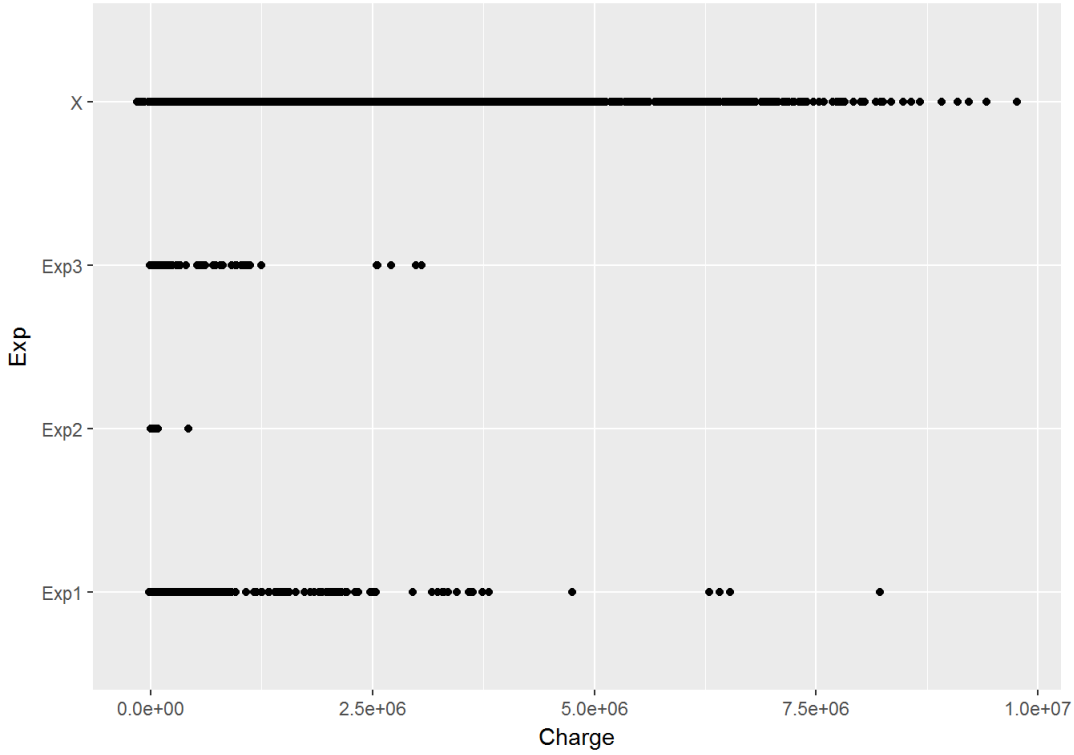

```
ggplot(data =Data ,aes(x=Charge,y=Prop))+
    geom_point()
```



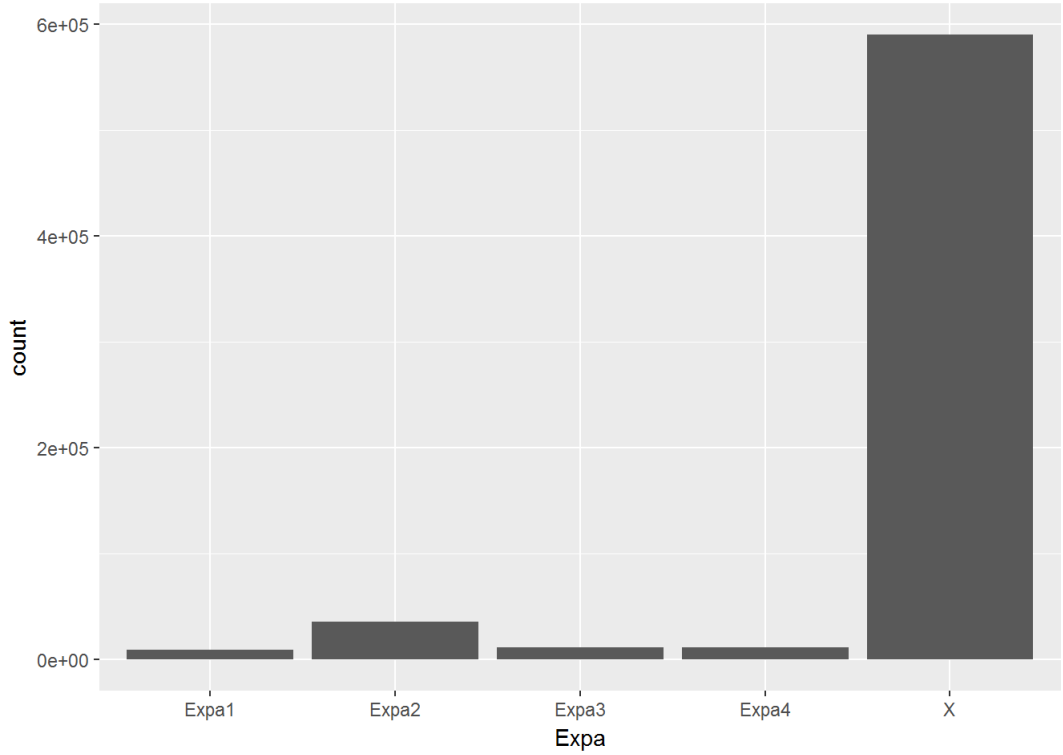

```
ggplot(data =Data ,aes(x=Exp),binwidth =50)+
    geom_bar()
```



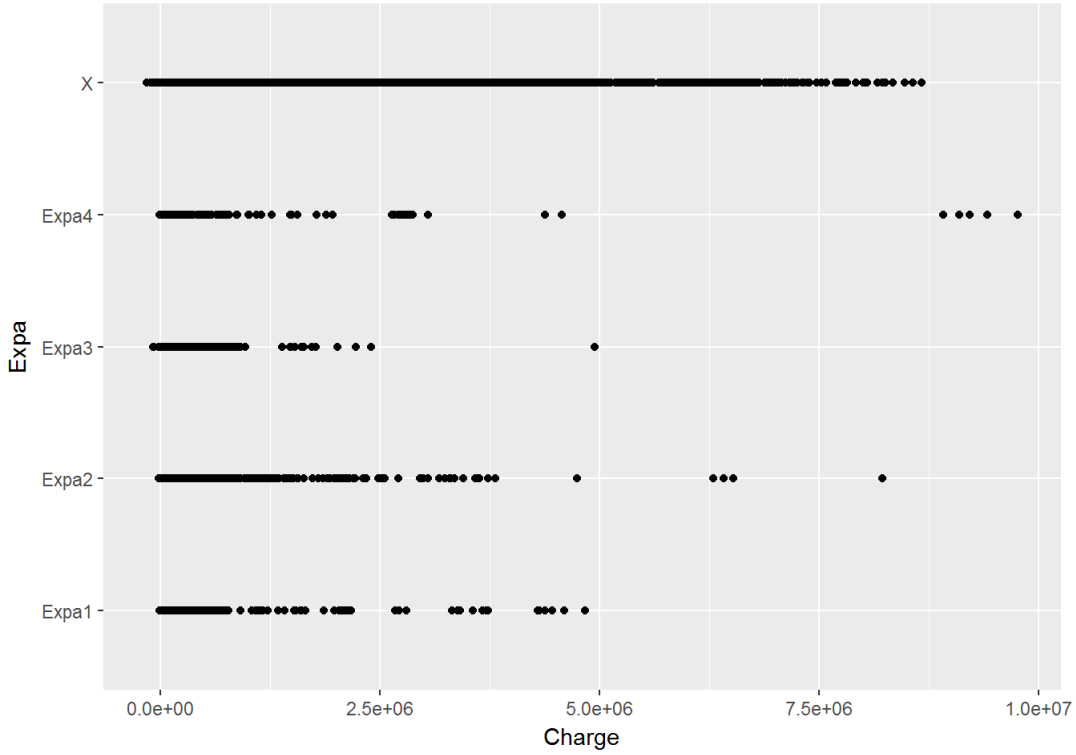

```
ggplot(data =Data ,aes(x=Charge,y=Exp))+
    geom_point()
```



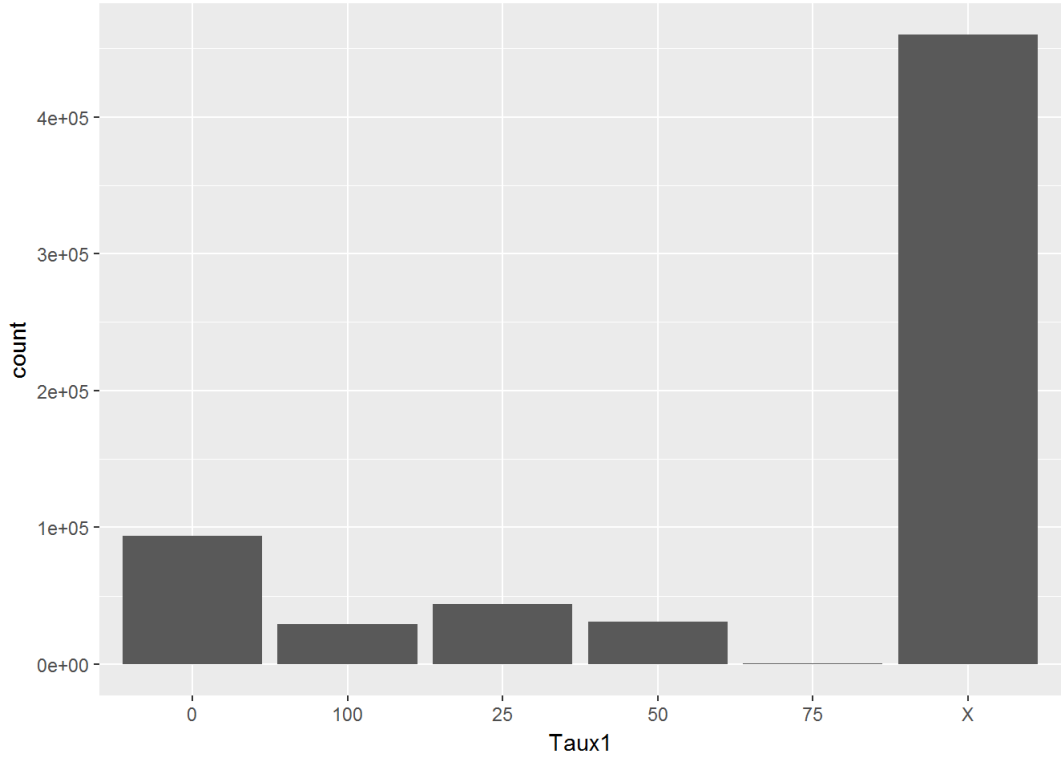

```
ggplot(data =Data ,aes(x=Expa),binwidth =50)+
    geom_bar()
```



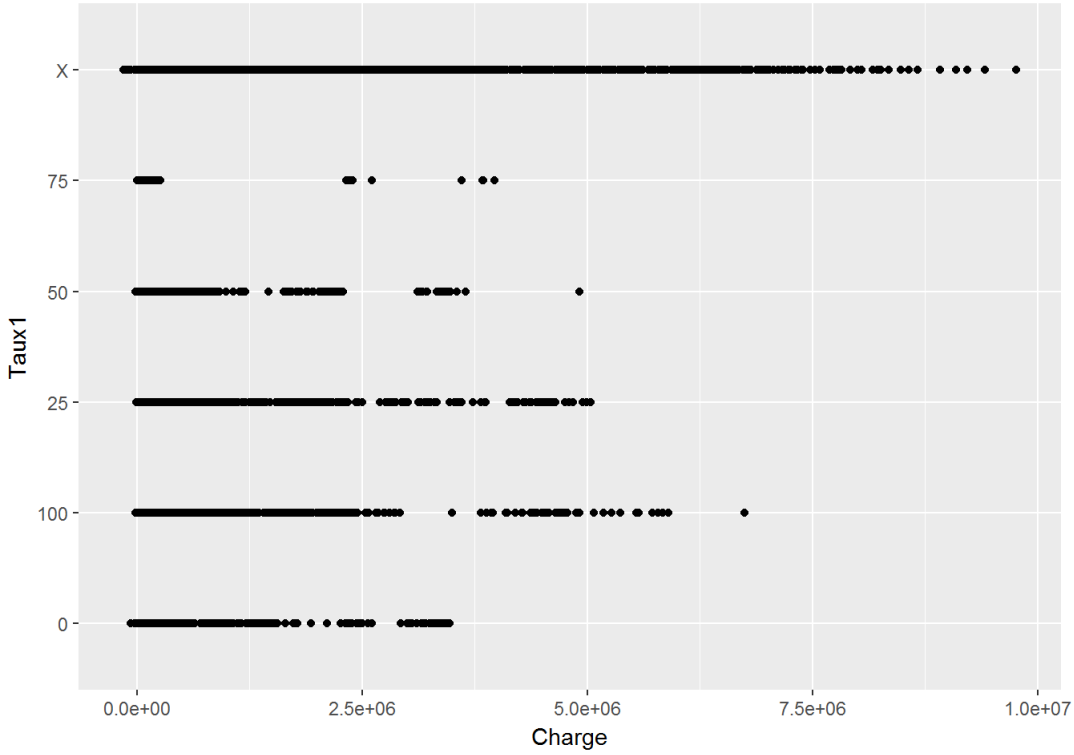

```
ggplot(data =Data ,aes(x=Charge,y=Expa))+
    geom_point()
```



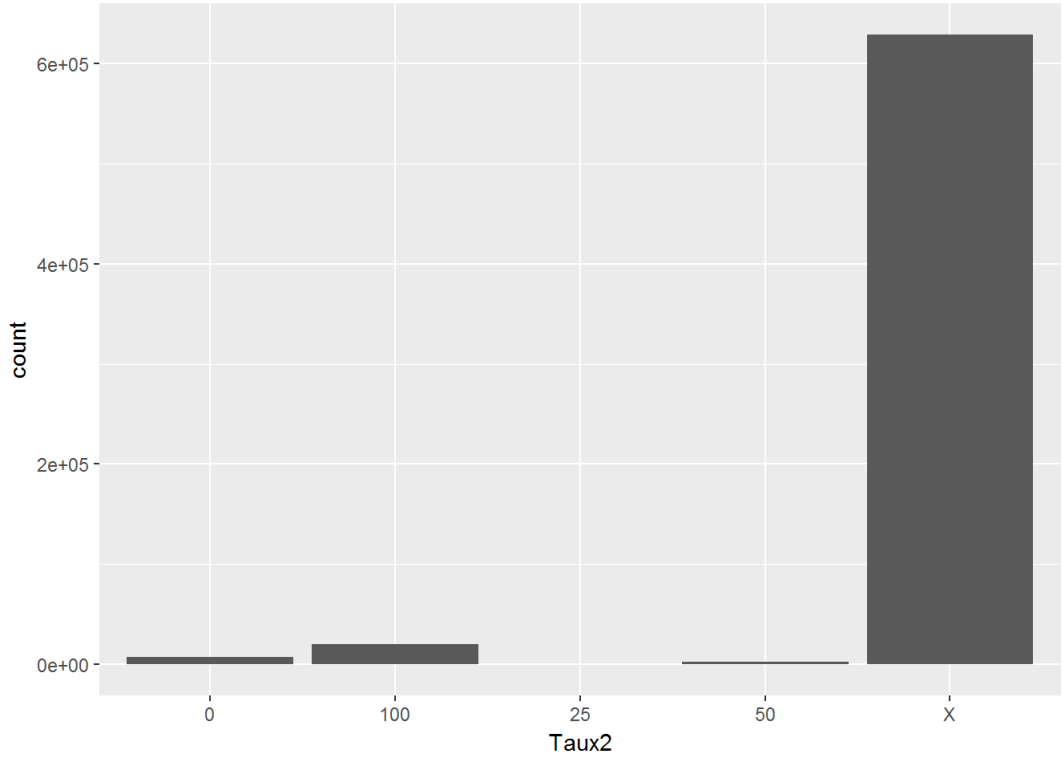

```
ggplot(data =Data ,aes(x=Taux1),binwidth =50)+
    geom_bar()
```



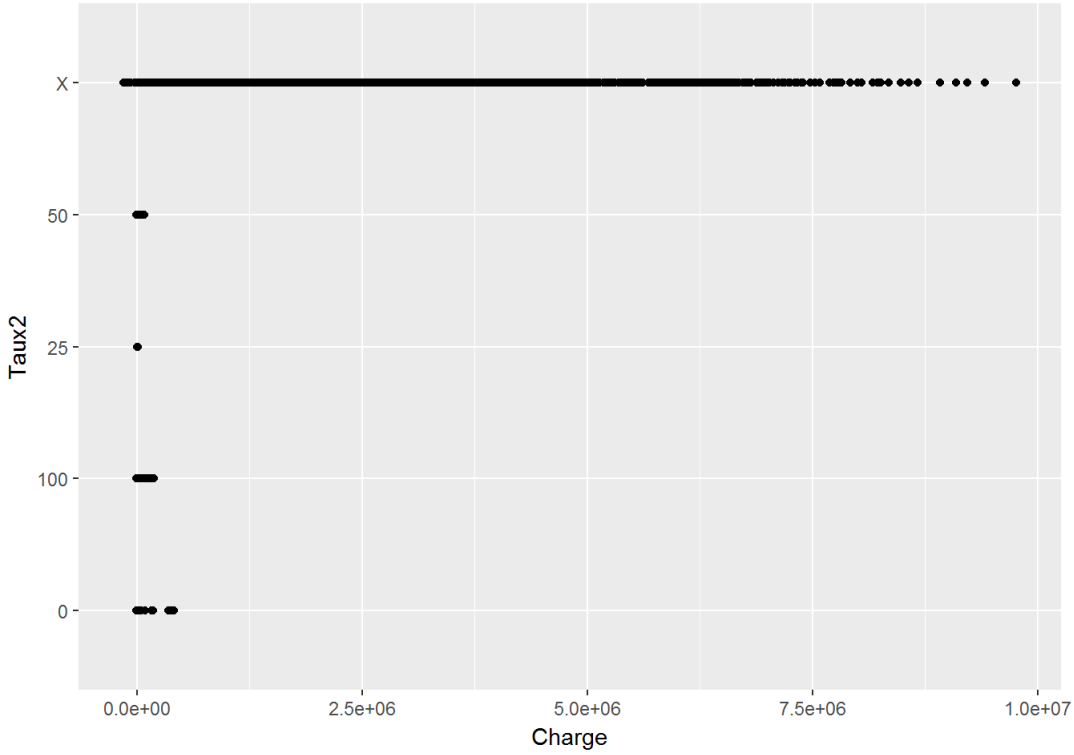

```
ggplot(data =Data ,aes(x=Charge,y=Taux1))+
    geom_point()
```



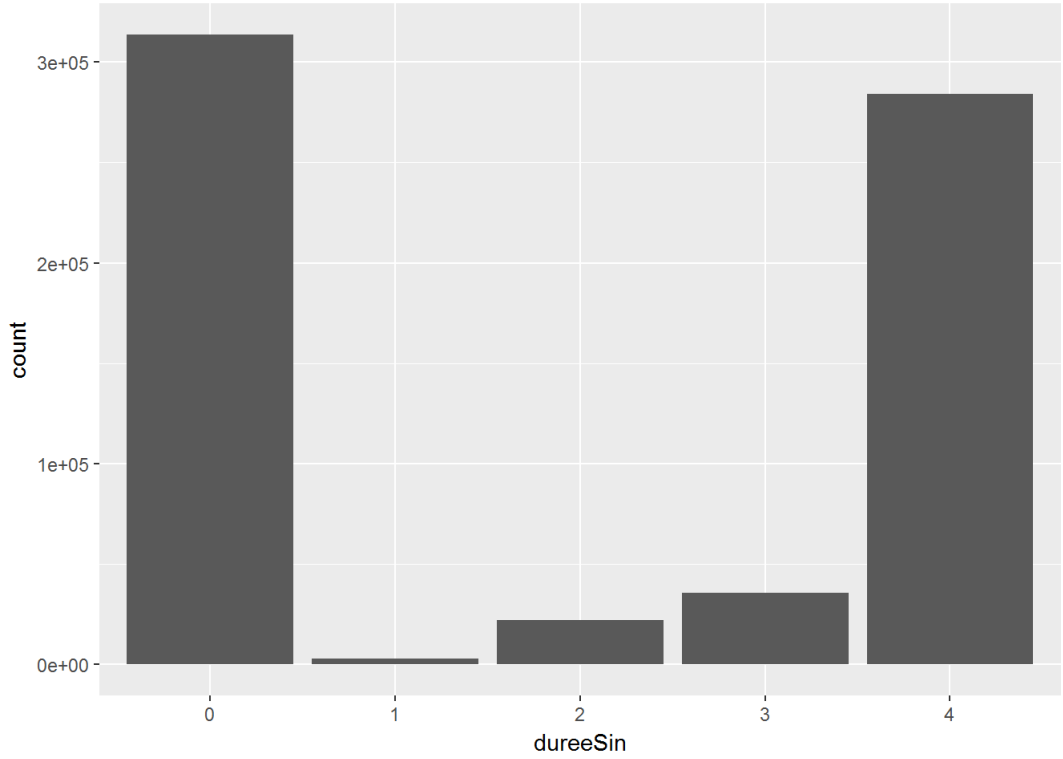

```
ggplot(data =Data ,aes(x=Taux2),binwidth =50)+
    geom_bar()
```



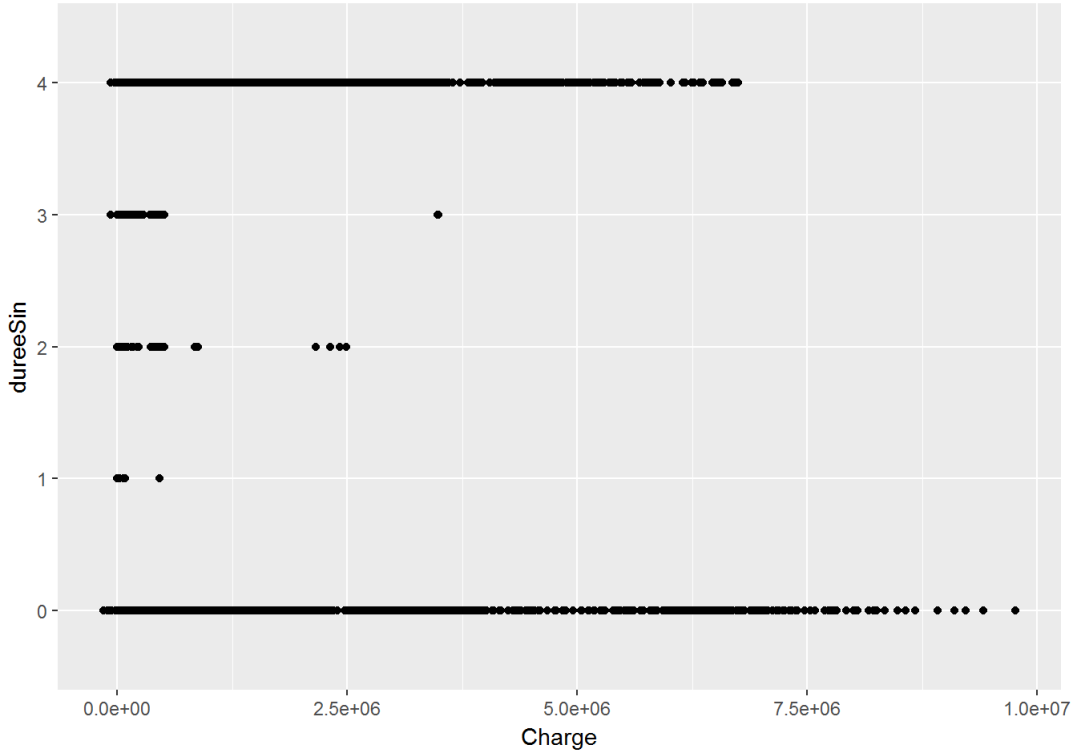

```
ggplot(data =Data ,aes(x=Charge,y=Taux2))+
    geom_point()
```



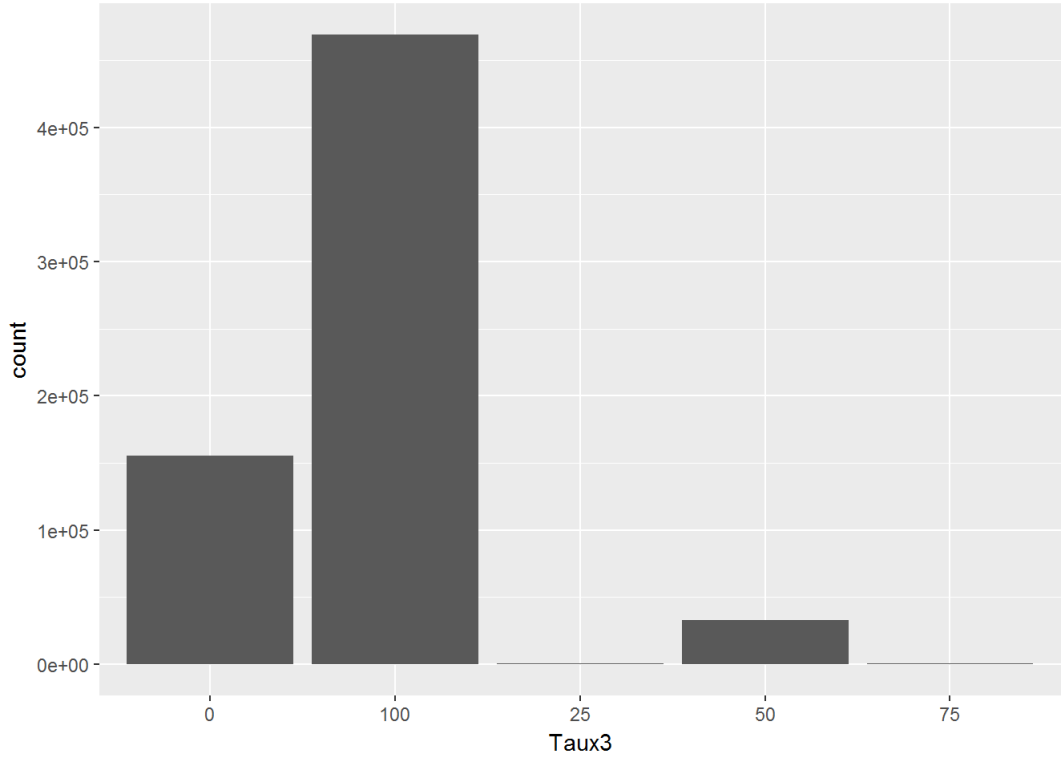

```
ggplot(data =Data ,aes(x=dureeSin),binwidth =50)+
     geom_bar()
```



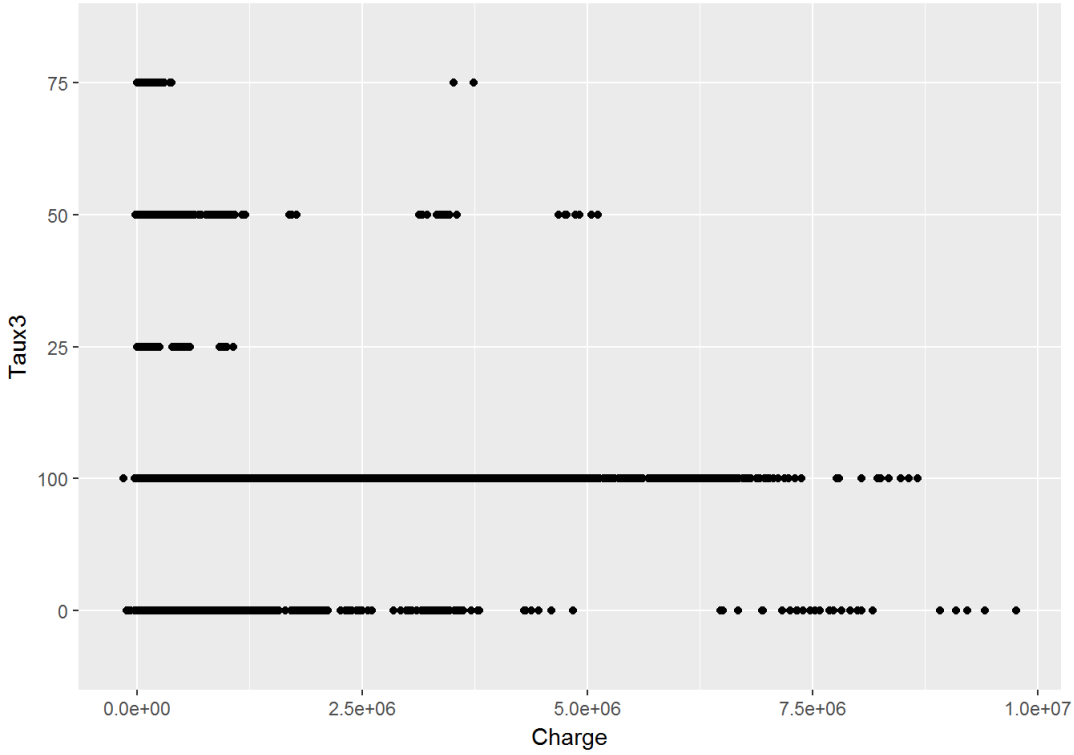

```
ggplot(data =Data ,aes(x=Charge,y=dureeSin))+
    geom_point()
```



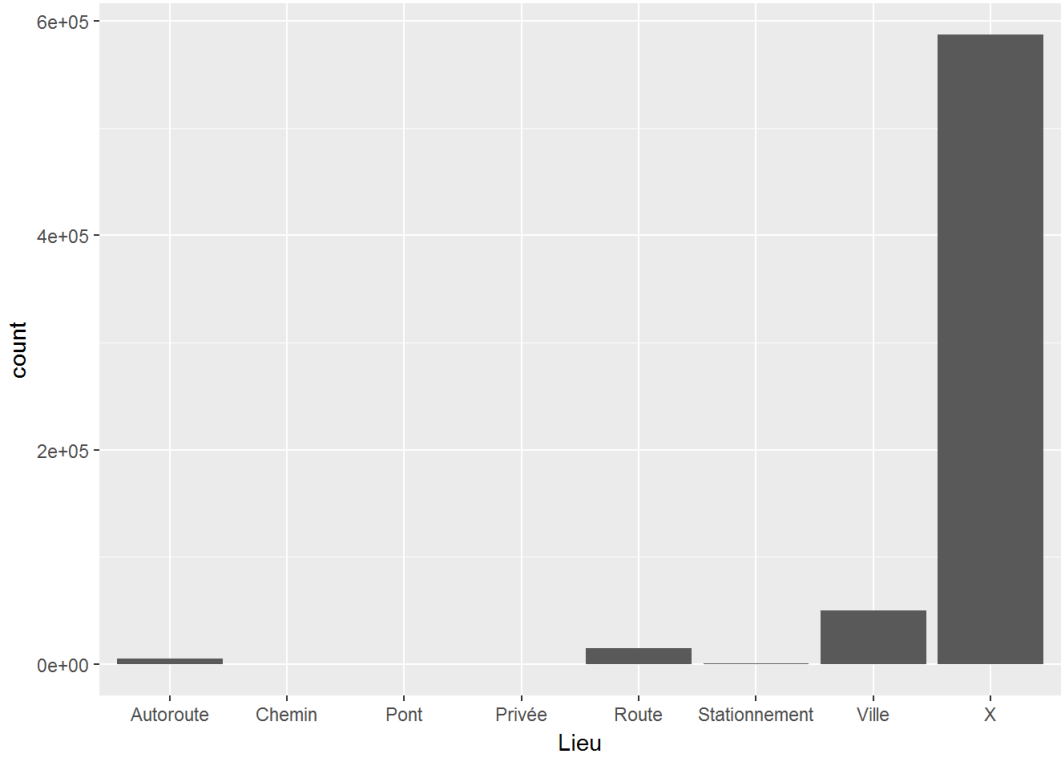

```
ggplot(data =Data ,aes(x=Taux3),binwidth =50)+
    geom_bar()
```



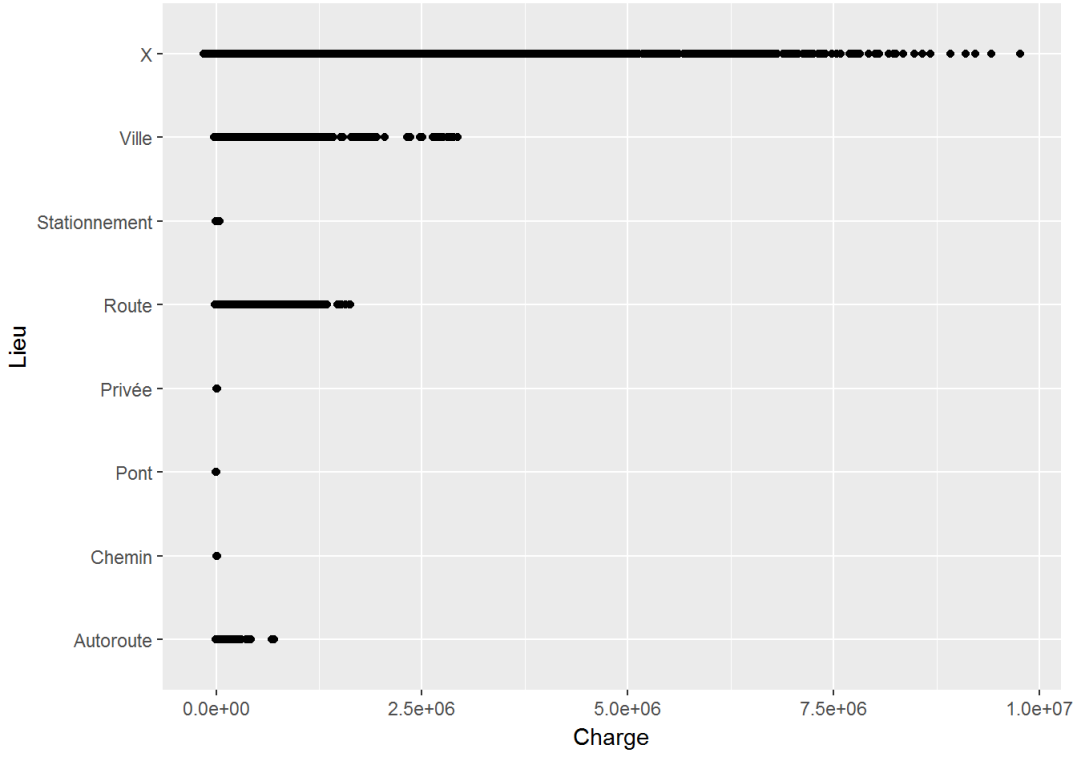

```
ggplot(data =Data ,aes(x=Charge,y=Taux3))+
    geom_point()
```




```
ggplot(data =Data ,aes(x=Lieu),binwidth =50)+
    geom_bar()
```




```
ggplot(data =Data ,aes(x=Charge,y=Lieu))+
    geom_point()
```



```r
#-------------------------------------------------------------------------------------------
#*********************************************************************
**************************Analyse descriptive**********************************

Data = readRDS("/Users/User/Downloads/base_sinistres_indiv.rds")
Data
require(dplyr)
Data%>%glimpse()
Da=filter(Data,Charge>0)
Da
unique(Da$Charge)
count(Da,Numéro_sinistre)
a=unique(Da$Numéro_sinistre)
length(a)
b=NULL
for(i in 1:29){
  b[[i]]=count(Da,Da[,i])
}
b
for(i in 1:29){
  barplot(b[[i]]$n,names=b[[i]]$`Da[, i]`)
}

#*****************************************Visualisation des profils de sinistres **********
a=unique(Data$Numéro_sinistre)
p=NULL
for(i in 1:50){
  p[[i]]=Data$Charge[which(Data$Numéro_sinistre==a[i])]
  plot(p[[i]],type='l')
}
```

Étude des valeurs extrêmes

```
#TRAITEMENT DE LA BASE DE DONNEES

#-----------------------------------------------------------------------------------------------------
library(evd)
par(mfrow=c(1,1))
mrlplot(X1$Charge,nt=100)
grid()
abline(v=1000000,col = "red")
tlim=c(0,7000000)
par(mfrow=c(1,2))
tcplot(X1$Charge,tlim = tlim,nt=100,std.err = FALSE)


#retrait des sinistres ayant plus de 1 000 000 comme charges à un instant donné.
sin_a_enlever =sqldf('select distinct numsin from X1 where Charges> 1000000')
Xdata1 = sqldf("select * from X1 where numsin not in sin_a_enlever")
```

Création de variables explicatives

```
#extraction de l'année de survenance d'un sinistre
Xdata1$an_de_surv = as.numeric(format(Xdata1$Date_de_surv, format = "%Y"))

#Ajout de la variable année de developpement (annedev)
Xdata1$annedev = Xdata1$Vue-Xdata1$an_de_surv

#certains sinistres ont des paiements effectués avant l'année de survenance
levels(as.factor(Xdata1$Vue[which(Xdata1$annedev<0)]))
Xdata1[which(Xdata1$numsin=="AW.AXIC3DEKPZ53M431Y"),]
Xdata1[which(Xdata1$numsin=="AW.AKDE51CMI0ZY5X6317"),]
Xdata1= Xdata1[-which(Xdata1$annedev<0),]

#table synthétisant les données par années de developpement
Xdatasyn1 = sqldf("select * from Xdata1 group by numsin,annedev")

tab1 = sqldf("select * from Xdata1 where Fin_vue = 9990")
tab = tab1[,-c(1,2,4,5,6,7)]

#ajustement des modalité pour quelles soient identique
#jointure pour recupérer l'information de la dernière ligne sur toutes les autres
tab2 = sqldf("select *
             from Xdata1 left join tab
             on Xdata1.numsin = tab.numsin")

tab3 = tab2[,-c(5,7:29,33,56:58)]

#variable permettant de connaitre le dernier règlement dans l'année
#si chgtreg =1 il s'agit du dernier reglement.
tab3$chgtreg =  pmin(rep(1,nrow(tab3)),tab3$Fin_Vue - tab3$Vue)

n = nrow(tab3)
tab3$reglement = c(tab3$Charges[1],tab3$Charges[2:nrow(tab3)]-tab3$Charges[1:(nrow(tab3)-1)])
tab3$reglement[which(tab3$Fin_Vue[-n]==9990)+1]=tab3$Charges[which(tab3$Fin_Vue[-n]==9990)+1]

#création d'une table synthétique avec une ligne correspondant
#a une année de dévéloppement ; le montant de l'année de dévelloppement
#est le dernier montant connu dans l'année considéré
Xdatassyns = sqldf("select * from tab3
                   group by numsin,annedev
                   having chgtreg=1")

Xdatassyns = sqldf("select * from tab3 group by numsin,annedev")

#Ajout de la variable dernière charge connu (Charge_prec)
n = nrow(Xdatassyns)
Xdatassyns$Charge_prec =  c(0,Xdatassyns$Charges[1:(n-1)])
Xdatassyns$Charge_prec[which(Xdatassyns$Fin_Vue[-n]==9990)+1]=0

#existe t-il des charges négatives ou nulles?
Xdatassyn2 =Xdatassyns[which(Xdatassyns$Charges<=0),]
Xdatassyns1 =Xdatassyns[which(Xdatassyns$Charges>0),]
```

Étude des corrélation

```r
#-------------------------------------------------------------------------
#
#     ETUDE DES CORRELATIONS
#
#-------------------------------------------------------------------------

#variables explicatives
colnames(Xdatassyns)

#-----------------------------------------------------
# variables numerique
#-----------------------------------------------------*
data_cor = Xdatassyns

#recherche des variables numériques
facteur = rep(0,ncol(data_cor))
for(i in 1:ncol(data_cor)){
  facteur[i] = !is.numeric(data_cor[,i])
}
colnames(data_cor[,which(facteur==0)])

#affichage des corrélations
library(corrplot)
mcor = cor(data_cor[,which(facteur==0)])
corrplot(mcor)

#-----------------------------------------------------
# variables factorielle
#-----------------------------------------------------

data_cor = Xdatassyns

#transformation en variables numériques
data_cor$Taux3 = as.numeric(data_cor$Taux3)

levels(as.factor(data_cor$Type_de_sinistre))
data_cor$Type_de_sinistre = ifelse(data_cor$Type_de_sinistre =="Accid. 1veh",1,
                                   ifelse(data_cor$Type_de_sinistre =="Accid. 2veh",2,
                                          ifelse(data_cor$Type_de_sinistre =="Accid. 3+veh",3
                                                 ifelse(data_cor$Type_de_sinistre =="Bdg",4,
                                                        ifelse(data_cor$Type_de_sinistre =="I

levels(as.factor(data_cor$Permis))
data_cor$Permis = ifelse(data_cor$Permis=="A",1,
                         ifelse(data_cor$Permis=="E",2,3))

levels(as.factor(data_cor$Tiers_acc))
data_cor$Tiers_acc = ifelse(data_cor$Tiers_acc=="OUI",1,0)

levels(as.factor(data_cor$Tiers_domm))
data_cor$Tiers_domm = ifelse(data_cor$Tiers_domm=="OUI",1,0)

levels(as.factor(data_cor$Police))
data_cor$Police = ifelse(data_cor$Police=="OUI",1,0)
```

```
55
56 levels(as.factor(data_cor$Chargement_veh))
57 data_cor$Chargement_veh = ifelse(data_cor$Chargement_veh=="Leger",1,0)
58
59 levels(as.factor(data_cor$blesse))
60 data_cor$blesse= ifelse(data_cor$blesse=="OUI",1,0)
61
62 levels(as.factor(data_cor$Usage))
63 data_cor$Usage = ifelse(data_cor$Usage=="Business",1,
64                         ifelse(data_cor$Usage=="Loisir",2,3))
65
66
67 data_cor$dureeSin = as.numeric(data_cor$dureeSin )
68
69 levels(as.factor(data_cor$Type_acc))
70 data_cor$Type_acc = ifelse(data_cor$Type_acc =="Famille",1,
71                            ifelse(data_cor$Type_acc =="Famille2",2,
72                                   ifelse(data_cor$Type_acc =="Travail",3,4)))
73
74 levels(as.factor(data_cor$Type_m))
75 data_cor$Type_m = ifelse(data_cor$Type_m=="Tech",1,0)
76
77 levels(as.factor(data_cor$Expa))
78 data_cor$Expa = ifelse(data_cor$Expa=="Expa1",1,
79                        ifelse(data_cor$Expa=="Expa2",2,
80                               ifelse(data_cor$Expa=="Expa3",3,
81                                      ifelse(data_cor$Expa=="Expa4",4,0))))
82 levels(as.factor(data_cor$Exp))
83 data_cor$Exp = ifelse(data_cor$Exp=="Exp1",1,
84                        ifelse(data_cor$Exp=="Exp2",2,
85                               ifelse(data_cor$Exp=="Exp3",3,0)))
86
87
88 levels(as.factor(data_cor$Etat))
89 data_cor$Etat = ifelse(data_cor$Etat=="Ouvert",1,
90                        ifelse(data_cor$Etat=="Clos",0,
91                               ifelse(data_cor$Etat=="Clos Def",-1,2)))
92
93 levels(as.factor(data_cor$Lieu))
94 data_cor$Lieu = ifelse(data_cor$Lieu=="Autoroute",1,
95                        ifelse(data_cor$Lieu=="Chemin",2,
96                               ifelse(data_cor$Lieu=="Pont",3,
97                                      ifelse(data_cor$Lieu=="Privée",5,
98                                             ifelse(data_cor$Lieu=="Route",6,
99                                                    ifelse(data_cor$Lieu=="Stationnement",7,
100                                                           ifelse(data_cor$Lieu=="Ville",8,0))
101
102
103 levels(as.factor(data_cor$type_repar))
104 data_cor$type_repar = ifelse(data_cor$type_repar=="Ret1",1,
105                              ifelse(data_cor$type_repar=="Ret2",2,0))
106
107
108 levels(as.factor(data_cor$Constat))
109 data_cor$Constat = ifelse(data_cor$Constat=="NON",2,
110                           ifelse(data_cor$Constat=="OUI",1,
```

```
                                    ifelse(data_cor$Constat=="Litige",2,
                                           ifelse(data_cor$Constat=="Non décla",3,4))))

levels(as.factor(data_cor$Assur))
data_cor$Assur = ifelse(data_cor$Assur =="NON",2,
                        ifelse(data_cor$Assur =="OUI",1,
                               ifelse(data_cor$Assur =="HORS",3,0)))

levels(as.factor(data_cor$Prop))
data_cor$Prop = ifelse(data_cor$Prop=="Gar",1,
                       ifelse(data_cor$Prop=="Gar_Avec",2,
                              ifelse(data_cor$Prop=="Gar_Sans",3,0)))

levels(as.factor(data_cor$Taux1))
data_cor$Taux1[which(data_cor$Taux1=="X")]="-1"
data_cor$Taux1 = as.numeric(data_cor$Taux1)

levels(as.factor(data_cor$Taux2))
data_cor$Taux2[which(data_cor$Taux2=="X")]="-1"
data_cor$Taux2 = as.numeric(data_cor$Taux2)

levels(as.factor(data_cor$Degat))
data_cor$Degat = ifelse(data_cor$Degat=="Degat1",1,
                        ifelse(data_cor$Degat=="Degat2",2,
                               ifelse(data_cor$Degat=="Degat3",3,
                                      ifelse(data_cor$Degat=="Degat4",4,
                                             ifelse(data_cor$Degat=="Degat5",5,
                                                    ifelse(data_cor$Degat=="Degat6",6,
                                                           ifelse(data_cor$Degat=="Degat7",7,

#mise à jour des variables numérique
facteur = rep(0,ncol(data_cor))
for(i in 1:ncol(data_cor)){
  facteur[i] = !is.numeric(data_cor[,i])
}
colnames(data_cor[,which(facteur==1)])

#affichage des correlarions
mcor = cor(data_cor[,which(facteur==0)])
par(mfrow=c(1,1))
corrplot(mcor,type="upper",tl.cex = 0.7)
```

Implémentation des méthodes usuelles

```r
#-----------------------------------------------------------------
#IMPLEMENTATION CHAINLADDER
#-----------------------------------------------------------------

ChainLadder = function (Donnee){

  Donneef = Donnee
  nl = nrow(Donnee)
  nc = ncol(Donnee)

  #calcul des ratios
  lambdaChap = rep(0,nc-1)
  for(j in 1: (nc-1)){
    lambdaChap[j] =  sum(Donnee[,j+1]) /sum((Donnee[-((nl-j+1):nl),j]))
  }

  #extention du modèle à la table
  for(j in 2:(nc)){
    for(i in (nl+2-j):nl){
      #Donneef[i,j] = prod(lambdaChap[(j-1):(nl+1-j)]) * Donneef[i,nl+1-j]
      Donneef[i,j] = lambdaChap[j-1]*Donneef[i,j-1]
    }
  }
  return(Donneef)
}

#-----------------------------------------------------------------
#IMPLEMENTATION MACK
#-----------------------------------------------------------------
Mack = function(Donnee,nban=0){

  Donneef = Donnee
  nl = nrow(Donnee)
  nc = ncol(Donnee)

  ##########################################################################
  #---------------------- Création des estimateurs de Mack ----------------------#
  ##########################################################################


  #calcul des ratios lambdaChap
  lambdaChap = rep(0,nc-1)
  for(j in 1: (nc-1)){
    lambdaChap[j] =  sum(Donnee[,j+1]) /sum((Donnee[-((nl-j+1):nl),j]))
  }

  #-----------------calcul des estimations de sigma2------------------------------
  sigma2chap = rep(0,nc-1)          #initialisation

  Cij_1 = Donnee[,-1]
  Cij = Donnee[,-nc]
```

```
52
53    coeff = (Cij_1/Cij)-lambdaChap
54    C = Cij * coeff^2
55    C[is.nan(C)] = 0                  # On remplace par 0 les Nan
56    j = c(1: (nc-2),1,1)              #j va de 1 à n-2.
57                                      #on rajoute les 1 pour des soucis de longueur du vecteur
58    sigma2chap = (1/(nl-j-1))*apply(C,1,sum)
59    sigma2chap[nc-1]=min((sigma2chap[nc-2]^2)/sigma2chap[nc-3],min(sigma2chap[nc-3],sigma2chap[
60
61
62    ##############################################################################
63    #--------------------------- Application du modèle --------------------------#
64    ##############################################################################
65
66    for(j in 2:(nc)){
67      for(i in (nl+2-j):nl){
68        #Donneef[i,j] = prod(lambdaChap[(j-1):(nl+1-j)]) * Donneef[i,nl+1-j]
69        Donneef[i,j] = lambdaChap[j-1]*Donneef[i,j-1]
70      }
71    }
72    #-----------------Calcul de la Reserve ----------------------------------
73    if(nban == 0){
74      R = 0
75      for(i in 1:nl){
76        R = R + (Donneef[i,nc]-Donnee[i,i])
77      }
78    }else{
79      R = 0
80      for(i in (nc-nban+1):nl){
81        R = R + (Donneef[i,nban]-Donnee[i,i])
82      }
83    }
84
85    #-----------------calcul des estimations de MSEP-------------------------
86
87    MSEP = rep(0,nl)
88    for( i in 2:nl){
89      coeff = 0
90      for (j in (nc-i+1):(nc-1)){
91              coeff = coeff + (sigma2chap[j]/(lambdaChap[j]^2))*(1/Donneef[i,j]  +  1/sum(Donne
92      }
93      MSEP[i] =(Donneef[i,nc]^2) *coeff
94    }
95    ##############################################################################
96    #--------------------------- Étude de l'erreur ------------------------------#
97    ##############################################################################
98    #Ci,j+1 = lambdaj*Ci,j+sigmaj*sqrt(Cij)*eps(i,j)
99
100   #calcul des résidus
101   Cij_1 = Donneef[,-1]
102   Cij = Donneef[,-nc]
103
104   eps = (Cij_1- lambdaChap *Cij)/(sigma2chap*sqrt(Cij))
105
106   return(list(Reserve=R,Prov_Cacul = Donneef,MSEP = MSEP,residus = eps,ratio = lambdaChap))
107 }
```

Création du triangle et application MACK

```
X = Data[,c(1,2,3,4,7)]

# utilisation du package sqldf qui permet d'effectuer des requêtes sql dan R
library(sqldf)


montant_max=1000000
sin_a_enlever =sqldf('select distinct numsin from X where charge> 1000000')
Xdata = sqldf("select * from X where numsin not in sin_a_enlever")

#extraction de l'année de survenance d'un sinistre
Xdata$an_de_surv = as.numeric(format(Xdata$Date_de_surv, format = "%Y"))

#extraction des annees de developpement
Xdata$annee_dev = Xdata$Vue-Xdata$an_de_surv

#extraction du dernier règlement dans l'année
Xdata$chgtreg =  pmin(rep(1,nrow(Xdata)),Xdata$Fin_Vue - Xdata$Vue)


X1 = sqldf("select * from Xdata
           group by numsin,annee_dev
           having chgtreg=1")

#extraction des règlements par année de dev
n = nrow(X1)
X1$reglement = c(X1$Charge[1],X1$Charge[2:nrow(X1)]-X1$Charge[1:(nrow(X1)-1)])
X1$reglement[which(X1$Fin_Vue[-n]==9990)+1]=X1$Charge[which(X1$Fin_Vue[-n]==9990)+1]

#extraction des annees de survenance
annee_surv = sqldf('select distinct an_de_surv
              from X1
              order by an_de_surv')


#----------------------- Creation du triangle ------------------------------{#}
ni =length(annee_surv$an_de_surv) #nombre d'années d'observations
nj = max(X1$annee_dev) #nombres d'années de développement

#initialisation
Tri = matrix(rep(0,ni*(nj+1)),nrow = ni)
Triangle = matrix(rep(0,ni*(nj+1)),nrow = ni)

#on crée la matrice conntenant l'ensembles des reglements effectués
#en fonction de l'année de survenance et de la vue considéré
for (i in 1:ni){
  for (j in 1:(nj+1)){
    Tri[i,j]= sum(X1$reglement[which((X1$annee_dev==(j-1))&(X1$an_de_surv==annee_surv[i,1]))])
  }
}

#on réalise le cumul des reglements pour avoir un triangles de charges cumulées
for(i in 1:19){
  Triangle[i,]=cumsum(Tri[i,])
```

```
  if(i!=1){Triangle[i,(19-i+2):19]=rep(0,length((19-i+2):19))
  }else{Triangle[i,]=cumsum(Tri[i,])}

}

#----------------- APPLICATION MACK  -------------------------
a=Mack(Triangle)
a$Reserve

#----------------------comparaison avec le package chain ladder ----------------------------
Tr =Triangle
Tr[which(Tr==0)]=NA
essai = MackChainLadder(Triangle = Tr, est.sigma="Mack")
sum(abs(essai$FullTriangle - Mack(Triangle)$Prov_Cacul))
mean((essai$FullTriangle - Mack(Triangle)$Prov_Cacul)^2)
#différence de l'ordre de 10^-7

#----------------------visualisation des erreurs---------------------------------
mt = Mack(Triangle)
u = data.frame(x = as.vector(mt$Prov_Cacul[,-1]),y =as.vector(mt$residus))
ggplot(data=u,aes(x,y))+
  geom_point()+
  labs(title="Résidus en fonction des Charges",x="Cij",y="Residus")

u2 = data.frame(x = 1:19,y=mt$MSEP)
ggplot(data=u2,aes(x,y))+
  geom_point()+
  labs(title ="Evolution du MSEP",x ="année de survenance",y ="MSEP")

u3 = data.frame(x = 1:19,y=mt$s2)
ggplot(data=u3,aes(x,y))+
  geom_point()+
  labs(title ="Evolution du sig^2",x ="année de dévéloppement",y ="sig^2")
```

Implémentation des modèles d'apprentissage du montant des sinistres

Fonctions de comparaisons

```
##########################################"
#
#   FONCTIONS AUXILIAIRES DE COMPARAISON
#
##########################################

#############################################################
# Calcul des critères des proximités de Chain Ladder
#############################################################
D_M = function(A){
  S = 0
  for(i in 1: length(A)){
    S = S + (A[[i]]$CL[19] -A[[i]]$model[19] )^2
  }
  return(sqrt(S/length(A)))
}
D_M(Resultat_svm)
################################################

C_M = function(A){
  S = 0
  for(i in 1: length(A)){
    S = S + cor(A[[i]]$CL,A[[i]]$model)
  }
  return((S/length(A)))
}
C_M(Resultat_svm)
################################################

R_M = function(A){
  R=0 ;RM=0; nc = length(A[[i]]$model)
  for(i in 1:18){
    RM = RM + (A[[i]]$model[nc]-A[[i]]$model[nc-i])
    R = R + (A[[i]]$CL[nc]-A[[i]]$CL[nc-i])
  }
  return(cbind(R,RM,(RM-R)/R))
}
R_M(Resultat_svm)

################################################
comp_CL = function(result){
  DM=D_M(result);ERM=R_M(result)[3];CM=C_M(result)
  score = log(DM)+abs(ERM)-CM
  return(data.frame(DM=DM, ERM =ERM,CM=CM,score = round(score,3)))
}
comp_CL(Resultat_svm)

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

```

```r
#*************************************************************
# Comparaison entre modèle
#*************************************************************
comp_modele = function(Xreel , Xpred){

  C= cor(Xreel,Xpred)
  RMSE = sqrt(mean((Xreel-Xpred)^2))
  #RMSLE = sqrt(mean((log((Xpred +1)/(Xreel+1))^2)))
  fit = lm(Xpred ~ Xreel)
  pente = fit$coefficients[2]
  score = (RMSE)/mean(Xreel) - pente - C
  return(cbind(Correlation = C, pente =pente, RMSE=RMSE,score = round(score,3)))
}

comp_modele(Xdatas_svm$Charges[-Bapp],predtest)

#*************************************************************
# interval de confiance pour plusieurs observations
#*************************************************************
Int_pred = function(x){
  n = length(x)
  x_ = mean(x)
  s = sd(x)
  a = 0.05
  t = qt(1-(a/2),n-1)
  IC = x_ +c(-1,1)*t*s/sqrt(n)
  return(IC)
}

#*************************************************************
# Affichage des résultats d'un modèle
#*************************************************************

affiche_res = function(u){
  ggplot(data=u,aes(x=Charges,y=t))+
    geom_point()+
    geom_smooth(method = lm,aes(colour = "droite de regression"))+
    geom_abline(color="red",size=2,slope=1,intercept=0,show.legend=TRUE,
                aes(colour = "première bissectrice")) +
    labs(title ="RF-performances prédictives du modèle sur la base de Test" ,
         y="valeurs prédites",x="valeur réelles")+
    scale_colour_manual("",values = c("blue","red"))+
    theme(legend.position="bottom")
}
```

Implémentation GLM

```
Xdata_glm = Xdatassyns1[,-c(32)]
Xdata_glm$Taux3 = as.factor(Xdata_glm$Taux3)
Xdata_glm$Type_de_sinistre = as.factor(Xdata_glm$Type_de_sinistre )
Xdata_glm$Permis = as.factor(Xdata_glm$Permis)
Xdata_glm$Tiers_acc = as.factor(Xdata_glm$Tiers_acc)
Xdata_glm$Tiers_domm = as.factor(Xdata_glm$Tiers_domm)
Xdata_glm$Police = as.factor(Xdata_glm$Police)
Xdata_glm$Chargement_veh = as.factor(Xdata_glm$Chargement_veh )
Xdata_glm$blesse = as.factor(Xdata_glm$blesse )
Xdata_glm$Usage = as.factor(Xdata_glm$Usage )
#Xdata_glm$dureeSin = as.factor(Xdata_glm$dureeSin )
Xdata_glm$Type_acc = as.factor(Xdata_glm$Type_acc )
Xdata_glm$Type_m = as.factor(Xdata_glm$Type_m )
Xdata_glm$Expa = as.factor(Xdata_glm$Expa )

#-------------------------------------------------------------------------------------------#
#
#                                Création d'un modèle
#
#-------------------------------------------------------------------------------------------#

  Xt = Xdata_glm

  set.seed(1)
  Bapp = sample(1:nrow(Xt),floor(0.8 * nrow(Xt)))
  colnames(Xt)


  train.h2o = as.h2o(Xt[Bapp,])
  test.h2o = as.h2o(Xt[-Bapp,])

  #-----------------------------------------------------------
  modele_simple_lnormal = h2o.glm(y = "Charges",
                                  x = colnames(Xt)[-7],
                                  training_frame = train.h2o,
                                  validation_frame = test.h2o,
                                  family = "gaussian",
                                  link = "log",lambda=0,nfolds=10,
                                  remove_collinear_columns =TRUE,
                                  keep_cross_validation_predictions = TRUE,
                                  compute_p_values=TRUE)

  #coefficients
  co1 = modele_simple_lnormal@model$coefficients
  co2 = modele_simple_lnormal@model$coefficients_table

  (co1[which(abs(co1)>10e-10)])
  (co2$names[which(co2$p_value<0.05)])

  #variables non significatives
  (co2[which(co2$p_value>0.05),c(1,5)])
  (co2$names[which(co2$p_value>0.05)])

  print(modele_simple_lnormal)
```

```r
pp = modele_simple_lnormal@parameters$x

#AIC et BIC du modèle
aic = h2o.aic(modele_simple_lnormal)
logL = -(aic-2*length(pp))/2 #log vraisemblance du modèle
bic = aic-2*length(pp) + log(nrow(Xdata_glm))*length(pp)
bic


#R2 ajuste
p = length(pp) ; n= nrow(Xt)
r2 = h2o.r2(modele_simple_lnormal)
r2_ajust = 1-((1-r2)*(n-1))/(n-p-1)

#*****************************************************************
#                          Selection BIC
#*****************************************************************
selection = function(sup){

  train.h2o = as.h2o(Xt[Bapp,-sup])
  test.h2o = as.h2o(Xt[-Bapp,-sup])

  #-------------------------------------------------------
  modele_simple_lnormal = h2o.glm(y = "Charges",
                                  x = colnames(Xt)[-c(7,sup)],
                                  training_frame = train.h2o,
                                  validation_frame = test.h2o,
                                  family = "gaussian",
                                  link = "log",lambda=0,nfolds=10,
                                  remove_collinear_columns =TRUE,
                                  keep_cross_validation_predictions = TRUE,
                                  compute_p_values=TRUE)

  #coefficients
  co2 = modele_simple_lnormal@model$coefficients_table

  #variables non significatives
  print((co2[which(co2$p_value>0.05),c(1,5)]))

  pp = modele_simple_lnormal@parameters$x

  #AIC et BIC du modèle
  aic = h2o.aic(modele_simple_lnormal)
  logL = -(aic-2*length(pp))/2 #log vraisemblance du modèle
  bic = aic-2*length(pp) + log(nrow(Xdata_glm))*length(pp)
  return(list(BIC = bic,logL = logL,model=modele_simple_lnormal))

}
colnames(Xt)

#processus de selection BIC
selection(9)$BIC
selection(19)$BIC
selection(12)$BIC
selection(c(12,9))$BIC
selection(c(12,19))$BIC
```

```
selection(c(12,13))$BIC
selection(c(12,13,9))$BIC
selection(c(12,13,19))$BIC


#test de deviance
a = selection(c(12,13))
D = -2*(a$logL - logL)
qchisq(0.95,2) #quantile du khi-2 à p2-p1 degré de liberté
pvalue = 1-pchisq(D,1)
pp2 = pp[-c(8,9)]

#-----------------------------------------------------------------
#modèle retenu
ModelGLM =a$model
train.h2o = as.h2o(Xt[Bapp,-c(12,13)])
test.h2o = as.h2o(Xt[-Bapp,-c(12,13)])

#AIC et BIC du modèle
aic = h2o.aic(ModelGLM)
bic = aic-2*(length(pp2)) + log(nrow(Xdata_glm))*length(pp2)
bic

#R2 ajuste
p = length(pp2) ; n= nrow(Xt)
r2 = h2o.r2(ModelGLM)
r2_ajust = 1-((1-r2)*(n-1))/(n-p-1)

#---------------------------------------
# etude des residus
#---------------------------------------
pred_app =as.data.frame (h2o.predict(ModelGLM,train.h2o))
eps = log(Xt$Charges[Bapp])-log(pred_app$predict)
eps_pearson = eps/sqrt(pred_app$StdErr)

#centree?
par(mfrow=c(1,2))
plot(eps,type="p")
abline(h=0,col="red")
plot(eps_pearson,type="p")
abline(h=0,col="red")

#loi normal?
par(mfrow=c(1,2))
sim = rnorm(length(eps))
qqplot(eps,sim)
f =lm(sort(sim)~sort(eps))
lines(eps,f$coefficients[2]*eps + f$coefficients[1],col="red")

f =lm(sort(sim)~sort(eps_pearson))
qqplot(eps_pearson,sim)
lines(eps_pearson,f$coefficients[2]*eps_pearson + f$coefficients[1],col="red")

par(mfrow=c(1,2))
hist(eps,probability = "TRUE")
curve(dnorm(x,0,1),col="red",add=TRUE)
```

```
167     hist(eps_pearson,probability = "TRUE")
168     curve(dnorm(x,0,1),col="red",add=TRUE)
169
170     #résidus  en fonction des prédictions
171     plot(pred_app$predict,eps,type="p")
172     abline(h=2,col="red")
173     abline(h=-2,col="red")
174     plot(pred_app$predict,eps_pearson,type="p")
175     abline(h=2,col="red")
176     abline(h=-2,col="red")
177
178
179     par(mfrow=c(1,1))
```

Implémentation ARBRE DE DECISION

```
require(rpart)
fit5=rpart(charge~.,data=Data)
plot(fit5)
text(fit5)
rsq.rpart(fit5)
names(fit5)
summary(fit5)
fit5$cptable
fit55=rpart(annedev~.,data=Xdaa[,-c(12,14)],control =rpart.control(minsplit = 3,cp=0.01 ))
plot(fit55)    #Optimisation des paramètres
text(fit55)
fit55
```

Implémentation SVM

```
#-----------------------------------------------------------------
#                MISE EN FORME DE LA BASE
#fonction qui permet d'avoir les variables explicatives dans le bon format
transforme = function(u){
  new = u[,-c(1,2,3,4,5,10,14,21:23,25,26,29,30)]

  new$Taux3 = as.factor(new$Taux3)
  levels(new$Taux3) =c(levels(new$Taux3), levels(as.factor(Xdatassyns$Taux3)))

  new$Type_de_sinistre = as.factor(new$Type_de_sinistre )
  levels(new$Type_de_sinistre) =c(levels(new$Type_de_sinistre),
                                  levels(as.factor(Xdatassyns$Type_de_sinistre)))

  new$Permis = as.factor(new$Permis)
  levels(new$Permis) =c(levels(new$Permis), levels(as.factor(Xdatassyns$Permis)))

  new$Tiers_acc = as.factor(new$Tiers_acc)
  levels(new$Tiers_acc) = c(levels(new$Tiers_acc) ,levels(as.factor(Xdatassyns$Tiers_acc)))

  new$Tiers_domm = as.factor(new$Tiers_domm)
  levels(new$Tiers_domm) = c(levels(new$Tiers_domm),levels(as.factor(Xdatassyns$Tiers_domm)))

    new$Police = as.factor(new$Police)
  levels(new$Police) = c(levels(new$Police) ,levels(as.factor(Xdatassyns$Police)))

  new$Chargement_veh = as.factor(new$Chargement_veh )
  levels(new$Chargement_veh) = c(levels(new$Chargement_veh),
                                 levels(as.factor(Xdatassyns$Chargement_veh)))

  new$blesse = as.factor(new$blesse )
  levels(new$blesse) = c(levels(new$blesse),levels(as.factor(Xdatassyns$blesse)))

  new$Usage = as.factor(new$Usage )
  levels(new$Usage) = c(levels(new$Usage), levels(as.factor(Xdatassyns$Usage)))

  new$dureeSin = as.factor(new$dureeSin )
  levels(new$dureeSin) = c(levels(new$dureeSin),levels(as.factor(Xdatassyns$dureeSin)))

  new$Type_acc = as.factor(new$Type_acc )
  levels(new$Type_acc) = c(levels(new$Type_acc),levels(as.factor(Xdatassyns$Type_acc)))

  new$Type_m = as.factor(new$Type_m )
  levels(new$Type_m) = c(levels(new$Type_m),levels(as.factor(Xdatassyns$Type_m)))

  new$Expa = as.factor(new$Expa )
  levels(new$Expa) = c(levels(new$Expa),levels(as.factor(Xdatassyns$Expa)))

  return(new)
}
Xdatas_svm = transforme(Xdatassyns1)
str(Xdatas_svm)

#-------------------------------------------------------------------------------#
#
```

```r
#                                  Création d'un modèle
#
#-------------------------------------------------------------------------------------#


library(e1071)

Xt = Xdatas_svm

#separation en base de test et base d'apprentissage
set.seed(1)# on veut que la base de test soit la même pour tous les modèle
Bapp = sample(1:nrow(Xt),floor(0.8 * nrow(Xt)))
modelssvm1 = svm(Charges~.,subset = Bapp,data = Xt,kernel ="radial",type="eps-regression")

#valeur de gamma retenu pour le svm radial
modelssvm1$gamma

#-------------------------------------------------
#qualités prédictives
#-------------------------------------------------

predapp = predict(modelssvm1,newdata=Xdatas_svm[Bapp,],type="response")
errqua=mean((predapp-Xdatas_svm$Charges[Bapp])^2)
RMSEsapp= sqrt(errqua)

predtest = predict(modelssvm1,newdata=Xdatas_svm[-Bapp,],type="response")
errqua=mean((predtest-Xdatas_svm$Charges[-Bapp])^2)
RMSEstest= sqrt(errqua)


par(mfrow=c(1,1))
u = cbind(Xdatas_svm[-Bapp,],predtest)
#affichage des résultats sur la base de test
affiche_res(u)
comp_modele(Xdatas_svm$Charges[-Bapp],predtest)
```

Implémentation DEEP LEARNING

```
# choix et mise en forme des variables explicatives
Xdatas_deep = Xdatassyns1[,-c(32)]
Xdatas_deep$Taux3 = as.factor(Xdatas_deep$Taux3)
Xdatas_deep$Type_de_sinistre = as.factor(Xdatas_deep$Type_de_sinistre )
Xdatas_deep$Permis = as.factor(Xdatas_deep$Permis)
Xdatas_deep$Tiers_acc = as.factor(Xdatas_deep$Tiers_acc)
Xdatas_deep$Tiers_domm = as.factor(Xdatas_deep$Tiers_domm)
Xdatas_deep$Police = as.factor(Xdatas_deep$Police)
Xdatas_deep$Chargement_veh = as.factor(Xdatas_deep$Chargement_veh )
Xdatas_deep$blesse = as.factor(Xdatas_deep$blesse )
Xdatas_deep$Usage = as.factor(Xdatas_deep$Usage )
Xdatas_deep$dureeSin = as.factor(Xdatas_deep$dureeSin )
Xdatas_deep$Type_acc = as.factor(Xdatas_deep$Type_acc )
Xdatas_deep$Type_m = as.factor(Xdatas_deep$Type_m )
Xdatas_deep$Expa = as.factor(Xdatas_deep$Expa )

str(Xdatas_deep)
#-------------------------------------------------------------------------------------#
#
#                                Création d'un modèle
#
#-------------------------------------------------------------------------------------#
# library(adabag)
library(h2o)

h2o.init()

Xt = Xdatas_deep

#on fixe la graine pour une répétabilité des résultats pour vérification
set.seed(1)

#Base d'apprentissage composé de  80 % des observations du tableau Xdata
Bapp = sample(1:nrow(Xt),floor(0.8 * nrow(Xt)))

#implémentation d'un random forest
h2o_data = as.h2o(Xt[Bapp,])
h2o_validation_data = as.h2o(Xt[-Bapp,])

model_dl_h2o = h2o.deeplearning(x=colnames(Xdatas_deep)[-7],y="Charges",
                                training_frame =h2o_data ,
                               validation = h2o_validation_data)

summary(model_dl_h2o)

#Importance des variables (en pourcentage)
dl_import = model_dl_h2o@model$variable_importances
barplot(dl_import[,4],names.arg = dl_import[,1] ,
        main = "Importance des variables",horiz = TRUE,cex.names = 0.5)

#-------------------------------------------------
#qualités prédictives
#-------------------------------------------------

```

```
predapph2o = h2o.predict(model_dl_h2o,h2o_data)
predtesth2o = h2o.predict(model_dl_h2o,h2o_validation_data)

test_dl = as.data.frame(predtesth2o)
app_dl = as.data.frame(predapph2o)

par(mfrow=c(1,1))
plot((Xt$Charges[-Bapp]),test_dl$predict)
abline(a=0,b=1,col="red")

plot((Xt$Charges[Bapp]),app_dl$predict)
abline(a=0,b=1,col="red")

cor(Xt$Charges[Bapp],app_dl$predict)
cor(Xt$Charges[-Bapp],test_dl$predict)

u = cbind(Xdatas_deep[-Bapp,],t=test_dl$predict)
affiche_res(u)

comp_modele(Xdatas_deep$Charges[-Bapp],test_dl$predict)
```

Implémentation RANDOM FOREST

```
Xdatas_rf = Xdatassyns1[,-32]
Xdatas_rf$Taux3 = as.factor(Xdatas_rf$Taux3)
Xdatas_rf$Type_de_sinistre = as.factor(Xdatas_rf$Type_de_sinistre )
Xdatas_rf$Permis = as.factor(Xdatas_rf$Permis)
Xdatas_rf$Tiers_acc = as.factor(Xdatas_rf$Tiers_acc)
Xdatas_rf$Tiers_domm = as.factor(Xdatas_rf$Tiers_domm)
Xdatas_rf$Police = as.factor(Xdatas_rf$Police)
Xdatas_rf$Chargement_veh = as.factor(Xdatas_rf$Chargement_veh )
Xdatas_rf$blesse = as.factor(Xdatas_rf$blesse )
Xdatas_rf$Usage = as.factor(Xdatas_rf$Usage )
Xdatas_rf$dureeSin = as.factor(Xdatas_rf$dureeSin )
Xdatas_rf$Type_acc = as.factor(Xdatas_rf$Type_acc )
Xdatas_rf$Type_m = as.factor(Xdatas_rf$Type_m )
Xdatas_rf$Expa = as.factor(Xdatas_rf$Expa )

#-------------------------------------------------------------------------------------#
#
#                                Création d'un modèle
#
#-------------------------------------------------------------------------------------#
library(randomForest)
library(h2o)

h2o.init()

Xt = Xdatas_rf

#on fixe la graine pour une répétabilité des résultats pour vérification
set.seed(1)
#Base d'apprentissage composé de  80 % des observations du tableau Xdata
Bapp = sample(1:nrow(Xt),floor(0.8 * nrow(Xt)))
Btest = setdiff(Bapp,1:nrow(Xt))

#*************************************************************************************
#*************************************************************************************
# MODELE COMPLEXE POUR RECHERCHER DES INTERVALLES DE CONFIANCE
# NB : ce principe peut être réutilisé pour toutes les autres méthodes
#*************************************************************************************
#*************************************************************************************


#*************************************************************************************
nbsubd = 10
Model_subd = function(nbsubd,Xt){

  model_rf = list()
  res_rf = list()
  pred_rf = matrix(0,nrow=nrow(Xt[-Bapp,]),ncol = nbsubd)

  for(i in 1:nbsubd){
    Btest_subd= sample(Bapp,floor(length(Bapp)/nbsubd) )
    Bapp_subd= setdiff(Bapp,Btest_subd)
    h2o_data_subd = as.h2o(Xt[Bapp_subd,])
    h2o_validation_data= as.h2o(Xt[-Bapp,])
```

```
55
56      model_rf[[i]] = h2o.randomForest(x=h2o.colnames(h2o_data_subd)[-which(h2o.colnames(h2o_da
57                                       y="Charges",
58                                       training_frame =h2o_data_subd ,
59                                       ntrees=100,
60                                       validation = h2o_validation_data)
61
62
63      predtesth2o_subd = h2o.predict(model_rf[[i]],h2o_validation_data)
64      test_rf_subd = as.data.frame(predtesth2o_subd)
65      res_rf[[i]] = comp_modele(Xt$Charges[-Bapp],test_rf_subd$predict)
66      pred_rf[,i] = test_rf_subd$predict
67
68
69    }
70    return(list(model_rf = model_rf,res_rf = res_rf,pred_rf = pred_rf))
71  }
72  M = Model_subd(nbsubd,Xdatas_rf)
73
74
75  #performances global du model
76  IC = apply(M$pred_rf,1,Int_pred)
77  Int_pred(M$pred_rf[2,])
78  affiche_res(cbind(Xdatas_rf[-Bapp,],t=apply(M$pred_rf,1,mean)))
79  comp_modele(Xdatas_rf$Charges[-Bapp],apply(M$pred_rf,1,mean))
80
81
82  val_pred = apply(M$pred_rf,1,mean)
83  comp_modele(Xdatas_rf$Charges[-Bapp],val_pred)
84  affiche_res(cbind(Xdatas_rf[-Bapp,],t=val_pred))
85
86  #-------------------------------------------------------------------------
87  #plot des intervalles de predictions
88  nb = 1000
89  bte = setdiff(1:nrow(Xdatas_rf),Bapp)
90  plot(Xdatas_rf$Charges[bte[1:nb]],val_pred[1:nb],pch=4,
91       xlab = "valeurs reelles",ylab = "valeurs predites",
92       main = paste("valeurs réelles en fonction des valeurs \n prédites de",
93                    nb,"individus pour n =",nbsubd))
94  abline(0,1,col="red")
95  for(i in 1:nb){
96    segments(x0 = Xdatas_rf$Charges[bte[i]],y0 = IC[1,i],x1 = Xdatas_rf$Charges[bte[i]],y1 = IC
97  }
98  legend(x = "bottomright",col = c("blue"),lty = 1,
99         legend = c("interval de prediction"))
```

Étude de la durée restante des sinistres

```r
#on ne considère que les sinistre définitivement clos pour réaliser l'étude
Xdatas_duree = Xdatassyns[which(Xdatassyns$Fin_Vue == "9990" & Xdatassyns$Etat=="Clos Def"),]

# choix et mise en forme des variables explicatives
Xdatas_duree$Taux3 = as.factor(Xdatas_duree$Taux3)
Xdatas_duree$Type_de_sinistre = as.factor(Xdatas_duree$Type_de_sinistre )
Xdatas_duree$Permis = as.factor(Xdatas_duree$Permis)
Xdatas_duree$Tiers_acc = as.factor(Xdatas_duree$Tiers_acc)
Xdatas_duree$Tiers_domm = as.factor(Xdatas_duree$Tiers_domm)
Xdatas_duree$Police = as.factor(Xdatas_duree$Police)
Xdatas_duree$Chargement_veh = as.factor(Xdatas_duree$Chargement_veh )
Xdatas_duree$blesse = as.factor(Xdatas_duree$blesse )
Xdatas_duree$Usage = as.factor(Xdatas_duree$Usage )
Xdatas_duree$dureeSin = as.factor(Xdatas_duree$dureeSin )
Xdatas_duree$Type_acc = as.factor(Xdatas_duree$Type_acc )
Xdatas_duree$Type_m = as.factor(Xdatas_duree$Type_m )
Xdatas_duree$Expa = as.factor(Xdatas_duree$Expa )

Xdatas_duree$duree = as.numeric(Xdatas_duree$Vue) - as.numeric(Xdatas_duree$an_de_surv)

#randomisation
duree = Xdatas_duree$duree+runif(nrow(Xdatas_duree))

#etude prelinaire de la duree
summary(duree)
sd(duree)^2
mean(duree)

library(ggplot2)
q1 = ggplot(data =as.data.frame(duree) ,aes(x=duree),binwidth =10)+
  geom_histogram(col="grey")+
  geom_density()

#hist(duree,probability = TRUE)
n = nrow(Xdatas_duree)

#-----------------------------------------------------------------
# loi des durées restantes
#-----------------------------------------------------------------

str(Xdatas_duree)
colnames(Xdatas_duree)
don = Xdatas_duree[,c(3,32)]
Xt_duree1 = sqldf("select *
                  from Xdatas left join don
                  where Xdatas.numsin = don.numsin")

Xt_duree1$ecartduree = Xt_duree1$duree-Xt_duree1$annedev
Xt_duree = Xt_duree1[,-c(2,3,4,5,7,31,32,33)]#on retire les variables
qu'on juge inutile pour l'étude
summary(Xt_duree$ecartduree)
```

```r


Xt_duree2 = Xt_duree
Xt_duree2$ecartduree = Xt_duree2$ecartduree +runif(nrow(Xt_duree2))


q2 = ggplot(data = as.data.frame(Xt_duree2),aes(x=ecartduree),binwidth =10)+
   geom_density(fill = "black",size = 0.5,linetype="dashed",alpha=0.5)+
q2

#-----------------------------------------
#loi exponentielle ?
loi_exp = fitdistr(Xt_duree2$ecartduree,"exponential")
x =seq(0,15,0.01)
q= q2+ stat_function(fun=function(x) dexp(x,loi_exp$estimate),
                     aes(colour = "loi exponentielle"),n=1000,size=1)
q
ks.test(duree,"pexp",loi_exp$estimate)

#-----------------------------------------
#loi gamma?
loi_gamma = fitdistr(Xt_duree2$ecartduree,"gamma")
curve(expr = dgamma(x,shape = loi_gamma$estimate[1],rate = loi_gamma$estimate[2]),col="blue",
ks.test(duree,"pgamma",loi_gamma$estimate[1],loi_gamma$estimate[2])
q= q+ stat_function(fun=function(x) dgamma(x,shape = loi_gamma$estimate[1],
                                           rate = loi_gamma$estimate[2]),
                    aes(colour = "loi gamma"),n=1000,size=1)
q

#-----------------------------------------
#loi poisson?
loi_poiss = fitdistr(Xt_duree2$ecartduree,"poisson")
lines(dpois(1:15,lambda = loi_poiss$estimate),col="green")
ks.test(duree,"ppois",loi_poiss$estimate)

#-----------------------------------------
#loi log-normale?
loi_lnormal = fitdistr(Xt_duree2$ecartduree,"lognormal")
curve(expr =dlnorm(x,meanlog  = loi_lnormal$estimate[1],sdlog = loi_lnormal$estimate[2]),col=
ks.test(Xt_duree2$ecartduree,"plnorm",meanlog  = loi_lnormal$estimate[1],sdlog = loi_lnormal$
q= q+ stat_function(fun=function(x) dlnorm(x,meanlog  = loi_lnormal$estimate[1],
                                           sdlog = loi_lnormal$estimate[2]),
                    aes(colour = "loi log-normale"),n=1000,size=1)
q
#-----------------------------------------
#loi Weibull?
loi_weibull = fitdistr(Xt_duree2$ecartduree,"weibull")
curve(expr =dweibull(x,shape  = loi_weibull$estimate[1],scale = loi_weibull$estimate[2]),col=
ks.test(Xt_duree2$ecartduree,"pweibull",shape  = loi_weibull$estimate[1],scale = loi_weibull$
q= q+ stat_function(fun=function(x) dweibull(x,shape  = loi_weibull$estimate[1],
                                             scale = loi_weibull$estimate[2]),
                    aes(colour = "loi de weibull"),n=1000,size=1)
q + scale_colour_manual("Lois ajustées", values = c("orange","greenyellow", "coral2", "deepsk


#-----------------------------------------------------------------------
```

```r
#-----------------------QQPLOT-------------------------------------
#------------------------------------------------------------------
a=data.frame(x=quantile(Xt_duree2$ecartduree,probs = seq(0,1,length.out = nrow(Xt_duree2))),
             y=qlnorm(seq(0,1,length.out = nrow(Xt_duree2)),meanlog  = loi_lnormal$estimate[1]
l=ggplot(data = a,aes(y=x,x=y))+
  geom_point()+
  geom_smooth(method="lm",linetype="dashed",colour="red",alpha=0.5)
l + labs( title ="qq-plot loi log-normal",y="réels",x="théoriques")


#------------------------------------------------------------------

b=data.frame(x=quantile(Xt_duree2$ecartduree,probs = seq(0,1,length.out = nrow(Xt_duree2))),
             y=qgamma(seq(0,1,length.out = nrow(Xt_duree2)),shape = loi_gamma$estimate[1],
                      rate = loi_gamma$estimate[2]))
l2=ggplot(data = b,aes(y=x,x=y))+
  geom_point()+
  geom_smooth(method="lm",linetype="dashed",colour="red",alpha=0.5)
l2 + labs( title ="qq-plot loi gamma",y="réels",x="théoriques")
```

Implémentation des modèles d'apprentissage de la durée

```
#-------------------------------------------------------------------------------
#------------------------------------ GLM --------------------------------------
#-------------------------------------------------------------------------------

colnames(Xt_duree1)
n = nrow(Xt_duree)
set.seed(1)
app = sample(1:n,0.5*n)

train.h2o = as.h2o(Xt_duree[app,])
test.h2o = as.h2o(Xt_duree[-app,])

#-----------------------------------------------------------
modele_duree_lnormal = h2o.glm(y = "ecartduree",
                        x = colnames(Xt_duree)[-27],
                        model_id = "duree",
                        training_frame = train.h2o,
                        validation_frame = test.h2o,
                        family = "gaussian",
                        link = "log",lambda=0,
                        remove_collinear_columns =TRUE,
                        compute_p_values=TRUE)

#Paramètres retenus
modele_duree_lnormal@parameters$x

#inforation sur le modèle
s_glm_duree = summary(modele_duree_lnormal)
duree_norm_aic = h2o.aic(modele_duree_lnormal)
param = modele_duree_lnormal@parameters$x
duree_norm_aic - 2*length(param) + log(nrow(Xt_duree)) *length(param)
h2o.r2(modele_duree_lnormal)
h2o.rmse(modele_duree_lnormal)

#prediction de la base de test
pred_glm =as.data.frame (h2o.predict(modele_duree_lnormal,test.h2o))
p_glm = round(pred_glm$predict)

#affichage des performances
u = data.frame(y = round(pred_glm$predict),duree = Xt_duree$ecartduree[-app])
ggplot(data=u,aes(x=duree,y=y))+
  geom_point()+
  geom_abline(color="red",linetype="dashed",size=1.5,slope=1,intercept=0,show.legend=TRUE,
              aes(colour = "première bissectrice")) +
  labs(title ="performances prédictives du modèle sur la base de Test" ,
       y="valeurs prédites",x="valeur réelles")+
  scale_colour_manual("",values = "red")+
  theme(legend.position="bottom")

#-------------------------------------------------------------------------------
#---------------------------------- RANDOM FOREST ------------------------------
```

```r
#-------------------------------------------------------------------------------
#-----------------------------------------------------------
X = Xt_duree
#X$ecartduree = as.factor(X$ecartduree)
train.h2o.rf = as.h2o(X[app,])
test.h2o.rf = as.h2o(X[-app,])
#-----------------------------------------------------------
modele_duree_rf = h2o.randomForest(x=colnames(Xt_duree)[-26],y="ecartduree",
                                   training_frame =train.h2o.rf ,
                                   ntree=500,
                                   mtries = -1,
                                   keep_cross_validation_predictions=TRUE,
                                   validation = test.h2o.rf)

#Paramètres retenus
modele_duree_rf@parameters$x

#prediction de la base de test
pred_rf =as.data.frame (h2o.predict(modele_duree_rf,test.h2o.rf))
p_rf = round(as.numeric(pred_rf$predict))
c = data.frame(y=round(as.numeric(pred_rf$predict)),x=Xt_duree$ecartduree[-app])

#affichage des performances
ggplot(data = c,aes(x,y))+
  geom_point()+
  geom_smooth(method="lm",colour="cyan3")+
  geom_abline(colour="red",linetype="dashed",intercept = 0,slope=1)+
  labs(title="RF performance du modèle sur la base de test",
       x="valeurs réelles",y="valeurs prédites")

cor(p_rf,Xt_duree$ecartduree[-app])
comp_modele(Xt_duree$ecartduree[-app],p_rf)


#-------------------------------------------------------------------------------
#----------------------------------- DEEP LEARNING -----------------------------
#-------------------------------------------------------------------------------
#-----------------------------------------------------------
X = Xt_duree
#X$ecartduree = as.factor(X$ecartduree)
train.h2o.deep = as.h2o(Xt_duree[app,])
test.h2o.deep = as.h2o(Xt_duree[-app,])
#-----------------------------------------------------------
modele_duree_deep = h2o.deeplearning(x=colnames(Xt_duree)[-27],y="ecartduree",
                                     training_frame =train.h2o.deep ,
                                     validation = test.h2o.deep)

#Paramètres retenus
modele_duree_deep@parameters$x

pred_deep =as.data.frame (h2o.predict(modele_duree_deep,test.h2o.deep))
p_deep = round(as.numeric(pred_deep$predict))

#affichage des performances
c = data.frame(y=p_deep,x=Xt_duree$ecartduree[-app])
ggplot(data = c,aes(x,y))+
```

```
  geom_point()+
  geom_smooth(method="lm",colour="cyan3")+
  geom_abline(colour="red",linetype="dashed",intercept = 0,slope=1)+
  labs(title="deep performance du modèle sur la base de test",
       x="valeurs réelles",y="valeurs prédites")

cor(p_deep,Xt_duree$ecartduree[-app])
comp_modele(Xt_duree$ecartduree[-app],p_deep)
```

Prédictions de l'évolution des charges avec les différents modèles calibrés

Prédictions GLM

```
#Prédiction pour 1 sinistre sin pendant d années
prediction_glm = function(sin,d){

    val = sin[nrow(sin),]
    val$Charge_prec = val$Charges

    fin  =sin$annedev[nrow(sin)] + d
    nban = min(18 - sin$annedev[nrow(sin)],max(fin- sin$annedev[nrow(sin)],0))

    #sinistre connu
    connu = rep(0,max(sin$annedev)+1)
    connu[sin$annedev+1]=sin$Charges
    #mise en forme pour les sinistres n'ayant pas ete paye pendant quelques annees
    for(i in which(connu==0)){
      connu[i]= ifelse((i<max(sin$annedev)+1) & (i!=1),connu[i-1],0)
    }

    if(nban == 0){
      complete_fin = rep(connu[length(connu)],ifelse((18-fin)>0,max(0,(18-fin)),0))
      return(c(connu,complete_fin))
    }else{
      pred = rep(0,nban)
      new = val[which(colnames(val) %in%  ModelGLM@parameters$x)]
      for(j in 1:nban){
        new$annedev = new$annedev + 1
        test.h2o = as.h2o(new)
        pred[j]  = as.data.frame(h2o.predict(ModelGLM,test.h2o))$predict
        new$Charge_prec = pred[j]
      }
      complete_fin = rep(pred[length(pred)],ifelse((18-fin)>0,18-fin,0))
      return(c(connu,pred,complete_fin))
    }

  }

t = Xdata_glm[which(Xdata_glm$numsin =="AW.A333YZDECP4KX87316"),]
prediction_glm(t,1)

#*********************************************************************************************
#Prédiction pour tout une année de développement
test_an_glm = function(annee,taille_echan=1000000){

  a_pred = Xdatassyns1[which((Xdatassyns1$Fin_Vue =="9990") & (Xdatassyns1$Etat != "Clos Def"
  table =a_pred[which(a_pred$an_de_surv==annee),]
  echantilon = sample(1:nrow(table),min(taille_echan,nrow(table)))
  test = table[echantilon,]

  test.h2o = as.h2o(test[,which(colnames(test) %in% colnames(Xt_duree))])
  duree = round(as.data.frame (h2o.predict(modele_duree_lnormal,test.h2o))$predict)
  tab_pred_an = rep(0,19)
```

```
  Tri_test_an = rep(0,19)

    for(i in 1:nrow(test)){
      t = Xdata_glm[which(Xdatassyns1$numsin == test$numsin[i]),]
      t2 = Xdatassyns1[which(Xdatassyns1$numsin == test$numsin[i]),]
      pred = prediction_glm(t,duree[i])
      tab_pred_an = tab_pred_an + pred
      i=i+1
      #recherche des charges connus pour ce sinistre afin de
      #remplir le vecteur et faire un chain ladder
      charge_conn = rep(0,19)
      charge_conn[t2$annedev+1] = t2$Charges
      ind_dif0 = max(t2$annedev)+1#indice de l'ann?e ? partir de laquelle on peut
      resultat_CL = c(charge_conn[1:ind_dif0],
                      charge_conn[ind_dif0] *cumprod(coeff_CL[ind_dif0:18]))

      Tri_test_an = Tri_test_an + resultat_CL
    }


    plot(Tri_test_an, main=annee,type = "o",
         ylim=c(min(tab_pred_an,Tri_test_an),max(tab_pred_an,Tri_test_an)))
    lines(tab_pred_an,col="red",lty=2)
    legend("bottomright",col =c("black","red"),
           legend = c("ChainLadder","Modele-GLM"),
           lty=1:2)

    return(list(model = tab_pred_an, CL =Tri_test_an))

  }


test_an_glm(1997,10)


Resultat_glm=list()
for(i in 1:length(ans)){
  cat(" ################################## \n","### ",ans[i],"### \n","########################
  Resultat_glm[[i]] = test_an_glm(ans[i])
  }
D_M(Resultat_glm)
C_M(Resultat_glm)
R_M(Resultat_glm)
```

Prédictions SVM

```
prediction = function(sin,d){

  val = sin[nrow(sin),]
  val$Charge_prec = val$Charges


  fin  =sin$annedev[nrow(sin)] + d
  nban = min(18 - sin$annedev[nrow(sin)],max(fin- sin$annedev[nrow(sin)],0))

  #sinistre connu
  connu = rep(0,max(sin$annedev)+1)
  connu[sin$annedev+1]=sin$Charges
  #mise en forme pour les sinistres n'ayant pas ete paye pendant quelques annees
  for(i in which(connu==0)){
    connu[i]= ifelse((i<max(sin$annedev)+1) & (i!=1),connu[i-1],0)
  }

  if(nban == 0){
    complete_fin = rep(connu[length(connu)],ifelse((18-fin)>0,max(0,(18-fin)),0))
    return(c(connu,complete_fin))
  }else{
    pred = rep(0,nban)
    new = transforme(val)
    for(j in 1:nban){
      new$annedev = new$annedev + 1
      pred[j]  = predict(modelssvm1,newdata=new)
      new$Charge_prec = pred[j]
    }
    complete_fin = rep(pred[length(pred)],ifelse((18-fin)>0,18-fin,0))
    return(c(connu,pred,complete_fin))
  }

}

n=nrow(Xdatas)
test = Xdatassyns1[which(Xdatassyns1$numsin =="AW.A333YKZI176PC1316"),]
test = Xdatassyns1[which(Xdatassyns1$numsin =="AW.A3PIZ60K6MDE299316"),]
test = Xdatassyns1[which(Xdatassyns1$numsin =="AW.A333YZDECP4KX87316"),]
essai = prediction(test,5)
length(essai)
essai


#-------------------------------------------------------------------------------
#-------------------------------------------------------------------------------
#-------------------------------------------------------------------------------

coeff_CL = Mack(Triangle)$ratio #coefficients de chain ladder

test_an = function(annee,taille_echan=1000){

  a_pred = Xdatassyns1[which((Xdatassyns1$Fin_Vue =="9990") & (Xdatassyns1$Etat != "Clos Def"
  table =a_pred[which(a_pred$an_de_surv==annee),]
  test = table[1:min(taille_echan,nrow(table)),]
```

```
55
56    test.h2o = as.h2o(test[which(colnames(test) %in% colnames(Xt_duree))])
57    duree = round(as.data.frame (h2o.predict(modele_duree_lnormal,test.h2o))$predict)
58
59    tab_pred_an = rep(0,19)
60    Tri_test_an = rep(0,19)
61
62    for(i in 1:nrow(test)){
63      t2 = Xdatassyns1[which(Xdatassyns1$numsin == test$numsin[i]),]
64      pred = prediction(t2,duree[i])
65      tab_pred_an = tab_pred_an + pred
66      #recherche des charges connus pour ce sinistre afin de
67      #remplir le vecteur et faire un chain ladder
68      charge_conn = rep(0,19)
69      charge_conn[t2$annedev+1] = t2$Charges
70      ind_dif0 = max(t2$annedev)+1 #indice de l'annee a partir de laquelle on peut
71      resultat_CL = c(charge_conn[1:ind_dif0],
72                      charge_conn[ind_dif0] *cumprod(coeff_CL[ind_dif0:18]))
73
74      Tri_test_an = Tri_test_an + resultat_CL
75    }
76
77
78     plot(Tri_test_an, main=annee,type = "o",
79          ylim=c(min(tab_pred_an,Tri_test_an),max(tab_pred_an,Tri_test_an)))
80     lines(tab_pred_an,col="red",lty=2)
81     legend("bottomright",col =c("black","red"),
82            legend = c("ChainLadder","Modele"),
83            lty=1:2)
84
85     return(list(model = tab_pred_an, CL =Tri_test_an))
86
87  }
88  test_an(1996,10)
89
90  Resultat_svm = list()
91  ans = 1989:2006
92  for(i in 1:length(ans)){
93    cat("################################## \n","### ",ans[i],"### \n","########################
94    Resultat_svm[[i]] = test_an(ans[i])
95  }
96  #-------------------------------------------------------------------------------------
97  #-------------------------------------------------------------------------------------
98  #-------------------------------------------------------------------------------------
99
100 D_M(Resultat_svm)
101 C_M(Resultat_svm)
102 R_M(Resultat_svm)
```

Prédictions DEEP LEARNING

```
#-------------------------------------------------
# Prédiction du triangle
#-------------------------------------------------

prediction_deep = function(sin,d){

  val = sin[nrow(sin),]
  val$Charge_prec = val$Charges

  fin  =sin$annedev[nrow(sin)] + d
  nban = min(18 - sin$annedev[nrow(sin)],max(fin- sin$annedev[nrow(sin)],0))

  #sinistre connu
  connu = rep(0,max(sin$annedev)+1)
  connu[sin$annedev+1]=sin$Charges
  #mise en forme pour les sinistres n'ayant pas ete paye pendant quelques annees
  for(i in which(connu==0)){
    connu[i]= ifelse((i<max(sin$annedev)+1) & (i!=1),connu[i-1],0)
  }

  if(nban == 0){
    complete_fin = rep(connu[length(connu)],ifelse((18-fin)>0,max(0,(18-fin)),0))
    return(c(connu,complete_fin))
  }else{
    pred = rep(0,nban)
    new = val
    for(j in 1:nban){
      new$annedev = new$annedev + 1
      test = as.h2o(new)
      pred[j]  = as.data.frame(h2o.predict(model_dl_h2o,test))$predict
      new$Charge_prec = pred[j]
    }
    complete_fin = rep(pred[length(pred)],ifelse((18-fin)>0,18-fin,0))
    return(c(connu,pred,complete_fin))
  }

}

t = Xdatas_deep[which(Xdatas_deep$numsin =="AW.A333YZDECP4KX87316"),]
prediction_deep(t,5)


test_an_deep = function(annee,taille_echan=1000){

  a_pred = Xdatassyns1[which((Xdatassyns1$Fin_Vue =="9990") & (Xdatassyns1$Etat != "Clos Def"
  table =a_pred[which(a_pred$an_de_surv==annee),]
  test = table[1:min(taille_echan,nrow(table)),]

  test.h2o = as.h2o(test[which(colnames(test) %in% colnames(Xt_duree))])
  duree = round(as.data.frame (h2o.predict(modele_duree_lnormal,test.h2o))$predict)
  tab_pred_an = rep(0,19)
  Tri_test_an = rep(0,19)

  for(i in 1:nrow(test)){
```

```
    t = Xdatas_deep[which(Xdatassyns1$numsin == test$numsin[i]),]
    t2 = Xdatassyns1[which(Xdatassyns1$numsin == test$numsin[i]),]
    pred = prediction_deep(t,duree[i])
    tab_pred_an = tab_pred_an + pred
    #recherche des charges connus pour ce sinistre afin de
    #remplir le vecteur et faire un chain ladder
    charge_conn = rep(0,19)
    charge_conn[t2$annedev+1] = t2$Charges
    ind_dif0 = max(t2$annedev)+1#indice de l'ann?e ? partir de laquelle on peut
    resultat_CL = c(charge_conn[1:ind_dif0],
                    charge_conn[ind_dif0] *cumprod(coeff_CL[ind_dif0:18]))

    Tri_test_an = Tri_test_an + resultat_CL
  }


  plot(Tri_test_an, main=annee,type = "o",
       ylim=c(min(tab_pred_an,Tri_test_an),max(tab_pred_an,Tri_test_an)))
  lines(tab_pred_an,col="red",lty=2)
  legend("bottomleft",col =c("black","red"),
         legend = c("ChainLadder","Modele-deepLearning"),
         lty=1:2)

  return(list(model = tab_pred_an, CL =Tri_test_an))

}


test_an_deep(1996,10)


Resultat_deep=list()
for(i in 1:length(ans)){
   cat(" ################################## \n","### ",ans[i],"### \n","#####################
   Resultat_deep[[i]] = test_an_deep(ans[i],100000)
 }
D_M(Resultat_deep)
C_M(Resultat_deep)
R_M(Resultat_deep)

#-------------------------------------------------------------------------------------------
#----------------------prediction avec un modèle de durée deep-learning---------------------
#-------------------------------------------------------------------------------------------

test_an_deep_deep = function(annee,taille_echan=1000000){

  a_pred = Xdatassyns1[which((Xdatassyns1$Fin_Vue =="9990") & (Xdatassyns1$Etat != "Clos Def"
  table =a_pred[which(a_pred$an_de_surv==annee),]
  test = table[1:min(taille_echan,nrow(table)),]

  test.h2o = as.h2o(test[,which(colnames(test) %in% colnames(Xt_duree))])
  duree = round(as.data.frame (h2o.predict(modele_duree_deep,test.h2o))$predict)

  tab_pred_an = rep(0,19)
  Tri_test_an = rep(0,19)

```

```
111   for(i in 1:nrow(test)){
112     t = Xdatas_deep[which(Xdatassyns1$numsin == test$numsin[i]),]
113     t2 = Xdatassyns1[which(Xdatassyns1$numsin == test$numsin[i]),]
114     pred = prediction_deep(t,duree[i])
115     tab_pred_an = tab_pred_an + pred
116     #recherche des charges connus pour ce sinistre afin de
117     #remplir le vecteur et faire un chain ladder
118     charge_conn = rep(0,19)
119     charge_conn[t2$annedev+1] = t2$Charges
120     ind_dif0 = max(t2$annedev)+1#indice de l'ann?e ? partir de laquelle on peut
121     resultat_CL = c(charge_conn[1:ind_dif0],
122                     charge_conn[ind_dif0] *cumprod(coeff_CL[ind_dif0:18]))
123 
124     Tri_test_an = Tri_test_an + resultat_CL
125     i=i+1
126   }
127 
128 
129   plot(Tri_test_an, main=annee,type = "o",
130        ylim=c(min(tab_pred_an,Tri_test_an),max(tab_pred_an,Tri_test_an)))
131   lines(tab_pred_an,col="red",lty=2)
132   legend("bottomleft",col =c("black","red"),
133          legend = c("ChainLadder","Modele deep-deep"),
134          lty=1:2)
135 
136   return(list(model = tab_pred_an, CL =Tri_test_an))
137 
138 }
139 
140 
141 test_an_deep_deep(1996,10)
142 test_an_deep(1996,10)
143 
144 
145 Resultat_deep_deep=list()
146 for(i in 1:length(ans)){
147   cat(" ################################## \n","### ",ans[i],"### \n","#######################
148   Resultat_deep_deep[[i]] = test_an_deep_deep(ans[i])
149 }
150 D_M(Resultat_deep_deep)
151 C_M(Resultat_deep_deep)
152 R_M(Resultat_deep_deep)
153 comp_CL(Resultat_deep_deep)
154 
155 #-------------------------------------------------------------------------------------------
156 #----------------------prediction avec un modèle de durée random forest---------------------
157 #-------------------------------------------------------------------------------------------
158 
159 test_an_rf_deep = function(annee,taille_echan=1000000){
160 
161   a_pred = Xdatassyns1[which((Xdatassyns1$Fin_Vue =="9990") & (Xdatassyns1$Etat != "Clos Def"
162   table =a_pred[which(a_pred$an_de_surv==annee),]
163   test = table[1:min(taille_echan,nrow(table)),]
164 
165   test.h2o = as.h2o(test[,which(colnames(test) %in% colnames(Xt_duree))])
166   duree = round(as.data.frame (h2o.predict(modele_duree_rf,test.h2o))$predict)
```

```
167
168   tab_pred_an = rep(0,19)
169   Tri_test_an = rep(0,19)
170
171   for(i in 1:nrow(test)){
172     t = Xdatas_deep[which(Xdatassyns1$numsin == test$numsin[i]),]
173     t2 = Xdatassyns1[which(Xdatassyns1$numsin == test$numsin[i]),]
174     pred = prediction_deep(t,duree[i])
175     tab_pred_an = tab_pred_an + pred
176     #recherche des charges connus pour ce sinistre afin de
177     #remplir le vecteur et faire un chain ladder
178     charge_conn = rep(0,19)
179     charge_conn[t2$annedev+1] = t2$Charges
180     ind_dif0 = max(t2$annedev)+1#indice de l'ann?e ? partir de laquelle on peut
181     resultat_CL = c(charge_conn[1:ind_dif0],
182                     charge_conn[ind_dif0] *cumprod(coeff_CL[ind_dif0:18]))
183
184     Tri_test_an = Tri_test_an + resultat_CL
185     i=i+1
186   }
187
188
189   plot(Tri_test_an, main=annee,type = "o",
190        ylim=c(min(tab_pred_an,Tri_test_an),max(tab_pred_an,Tri_test_an)))
191   lines(tab_pred_an,col="red",lty=2)
192   legend("bottomleft",col =c("black","red"),
193          legend = c("ChainLadder","Modele deep-deep"),
194          lty=1:2)
195
196   return(list(model = tab_pred_an, CL =Tri_test_an))
197
198 }
199
200
201 test_an_deep_deep(1996,10)
202 test_an_rf_deep(1996,10)
203 test_an_deep(1996,10)
204
205
206 Resultat_rf_deep=list()
207 for(i in 1:length(ans)){
208   cat(" ################################## \n","### ",ans[i],"### \n","#########################
209   Resultat_rf_deep[[i]] = test_an_rf_deep(ans[i])
210 }
211 D_M(Resultat_rf_deep)
212 C_M(Resultat_rf_deep)
213 R_M(Resultat_rf_deep)
214 comp_CL(Resultat_rf_deep)
```

Prédictions RANDOM FOREST avec intervalles de confiance

```
#-------------------------------------------------------------------------
#---------------------------------------------------------
#Prediction associée à ce modèle
Predire = function(h2o_data){
  pred = rep(0,nbsubd)
  for(i in 1:nbsubd){
    predtesth2o_subd = h2o.predict(M$model_rf[[i]],h2o_data)
    test_rf_subd = as.data.frame(predtesth2o_subd)
    pred[i] = test_rf_subd$predict
  }
  val_pred = mean(pred)

  return(list(predict = val_pred,Int_C = Int_pred(pred)))
}
#---------------------------------------------------------
Predire(as.h2o(test_rf[1,]))


#---------------------------------------------------------
#pREDICYIONS G2N2RALES
prediction_rf_int = function(sin,d){

  val = sin[nrow(sin),]
  val$Charge_prec = val$Charges


  fin  =sin$annedev[nrow(sin)] + d
  nban = min(18 - sin$annedev[nrow(sin)],max(fin- sin$annedev[nrow(sin)],0))

  #sinistre connu
  connu = rep(0,max(sin$annedev)+1)
  connu[sin$annedev+1]=sin$Charges
  #mise en forme pour les sinistres n'ayant pas ete paye pendant quelques annees
  for(i in which(connu==0)){
    connu[i]= ifelse((i<max(sin$annedev)+1) & (i!=1),connu[i-1],0)
  }
  IC_connu = cbind(connu,connu)


  if(nban == 0){
    complete_fin = rep(connu[length(connu)],ifelse((18-fin)>0,max(0,(18-fin)),0))
    IC_fin = cbind(complete_fin,complete_fin)
    IC = rbind(IC_connu,IC_fin)
    return(list(prediction = c(connu,complete_fin) , IC = IC))


  }else{
    pred = rep(0,nban)
    IC_borne_sup = rep(0,nban)
    IC_borne_inf = rep(0,nban)
    new = val
    for(j in 1:nban){
      new$annedev = new$annedev + 1
      test.h2o = as.h2o(new)
```

```
      p = Predire(test.h2o)
      pred[j]  = p$predict
      IC_borne_inf[j] = p$Int_C[1]
      IC_borne_sup[j] = p$Int_C[2]
      new$Charge_prec = pred[j]
    }
    complete_fin = rep(pred[length(pred)],ifelse((18-fin)>0,18-fin,0))
    fin_IC_sup  = rep(IC_borne_sup[nban],length(complete_fin))
    fin_IC_inf  = rep(IC_borne_inf[nban],length(complete_fin))
    IC_fin = cbind(fin_IC_inf,fin_IC_sup)

    IC = rbind(IC_connu,cbind(IC_borne_inf,IC_borne_sup),IC_fin)

    return(list(prediction = c(connu,pred,complete_fin),IC= IC))
  }

}

#test_rf = Xdatas_rf[which(Xdatas_rf$numsin =="AW.A333YKZI176PC1316"),]
#test_rf  = Xdatas_rf[which(Xdatas_rf$numsin =="AW.A3PIZ60K6MDE299316"),]
test_rf  = Xdatas_rf[which(Xdatas_rf$numsin =="AW.A333YZDECP4KX87316"),]
temp = Sys.time()
essai = prediction_rf_int(test_rf ,1)
Sys.time()-temp

plot(essai$prediction,type="b",ylim = c(min(essai$IC[,1]),max(essai$IC[,2])),main = "sinistre
lines(essai$IC[,1] ,col = "red")
lines(essai$IC[,2] ,col = "blue")

#-------------------------------------------------------------------------------
#-------------------------------------------------------------------------------
#-------------------------------------------------------------------------------
a_connu = Xdatassyn2[which((Xdatassyn2$Fin_Vue =="9990") & (Xdatassyn2$Etat == "Clos Def")),]

#-------------------------------------------------------------------------------
#-------------------------------------------------------------------------------
#-------------------------------------------------------------------------------

coeff_CL = Mack(Triangle)$ratio #ratio de chain ladder déja calculé pour cette base

test_an_rf_int = function(annee,taille_echan=10000000){

  set.seed(1)
  a_pred = Xdatassyns1[which((Xdatassyns1$Fin_Vue =="9990") & (Xdatassyns1$Etat != "Clos Def"
  table =a_pred[which(a_pred$an_de_surv==annee),]
  echantilon = sample(1:nrow(table),min(taille_echan,nrow(table)))
  test = table[echantilon,]

  test.h2o = as.h2o(test[which(colnames(test) %in% colnames(Xt_duree))])
  duree = round(as.data.frame (h2o.predict(modele_duree_lnormal,test.h2o))$predict)
  tab_pred_an = rep(0,19)
  Tri_test_an = rep(0,19)
  ic_pred_an_inf = rep(0,19)
  ic_pred_an_sup = rep(0,19)

  for(i in 1:nrow(test)){
```

```
    t2 = Xdatas_rf[which(Xdatassyns1$numsin == test$numsin[i]),]
    p = prediction_rf_int(t2,duree[i])
    pred = p$predict
    ic_pred_an_inf = ic_pred_an_inf + p$IC[,1]
    ic_pred_an_sup = ic_pred_an_sup + p$IC[,2]

    tab_pred_an = tab_pred_an + pred

    #recherche des charges connus pour ce sinistre afin de
    #remplir le vecteur et faire un chain ladder
    charge_conn = rep(0,19)
    charge_conn[t2$annedev+1] = t2$Charges
    ind_dif0 = max(t2$annedev)+1#indice de l'ann?e ? partir de laquelle on peut
    resultat_CL = c(charge_conn[1:ind_dif0],
                    charge_conn[ind_dif0] *cumprod(coeff_CL[ind_dif0:18]))

    Tri_test_an = Tri_test_an + resultat_CL
  }


  plot(Tri_test_an, main=annee,type = "o",
       ylim=c(min(tab_pred_an,Tri_test_an),max(tab_pred_an,Tri_test_an)))
  lines(tab_pred_an,col="red",lty=2)
  lines(ic_pred_an_inf,col = "green",lty = 3)
  lines(ic_pred_an_sup,col = "green",lty = 3)
  legend("bottomright",col =c("black","red","green"),
         legend = c("ChainLadder","Modele","Interval de prediction"),
         lty=1:3)

  return(list(model = tab_pred_an, CL =Tri_test_an,IC = cbind(ic_pred_an_inf,ic_pred_an_sup))

}
test_an_rf_int(1996,10)
test_an(1996,10)

#-------------------------------------------------------------------------
#PREDICTIONS GLOBALES
Resultat_rf_int = list()
ans = 1989:2006
for(i in 1:length(ans)){
  cat("################################## \n","### ",ans[i],"### \n","########################
  Resultat_rf_int[[i]] = test_an_rf_int(ans[i])
}

D_M(Resultat_rf_int)
C_M(Resultat_rf_int)
R_M(Resultat_rf_int)
comp_CL(Resultat_rf_int)
```

Comparaison des modèles

```r
#*****************************************************************************************

#            ETUDE ET COMPARAISON DES MODELES

#*****************************************************************************************
#Nombre de sinistre par années de survenance
#*****************************************************************************************
a_pred = Xdatassyns1[which((Xdatassyns1$Fin_Vue =="9990") & (Xdatassyns1$Etat != "Clos Def")),]
etude = cbind(a_pred$Charges,a_pred$an_de_surv)
et =NULL
for(i in 1:length(ans)){
  et[i] =  length(which(a_pred$an_de_surv==ans[i]))
}
barplot(et,ans,axisnames = TRUE,names.arg = ans,main = "nombre de sinistre par année de survei

#*****************************************************************************************
#Comparaison des modèles de prédiction des charges
#*****************************************************************************************

c1 = comp_modele(Xdatas_svm$Charges[-Bapp],predtest)#svm
c2 = comp_modele(Xdatas_rf$Charges[-Bapp],val_pred)#rf
c3 = comp_modele(Xdata_glm$Charges[-Bapp],pred$predict[-1])#glm
c4 = comp_modele(Xdatas_deep$Charges[-Bapp],test_dl$predict)#deep

res = data.frame(modele = c("svm","rf","glm","deepL"),rbind(c1,c2,c3,c4))

ggplot(data=res, aes(y=score,x = modele,fill=modele) )+
  geom_bar(stat="identity",width=0.5)+
  geom_text(aes(label=score), vjust=-0.3, size=3.5)+
  scale_fill_brewer(palette="Blues")+
  scale_x_discrete(limits=res$modele[order(res[,5])])+
  labs(title="Classement des modèles par rapport à leurs qualités prédictives")

#*****************************************************************************************
#*******************************MODELES DE DUREE******************************************
#*****************************************************************************************
d1 = comp_modele(Xt_duree$ecartduree[-app],p_rf)#rf
d2 = comp_modele(Xt_duree$ecartduree[-app],p_glm)#glm
d3 = comp_modele(Xt_duree$ecartduree[-app],p_deep)#deep

resd = data.frame(modele = c("glm","rf","deepL"),rbind(d1,d2,d3))

ggplot(data=resd, aes(y=score,x = modele,fill=modele) )+
  geom_bar(stat="identity",width=0.5)+
  geom_text(aes(label=score), vjust=-0.3, size=3.5)+
  scale_fill_brewer(palette="Blues")+
  scale_x_discrete(limits=resd$modele[order(resd[,5])])+
  labs(title="Classement des modèles par rapport à leurs qualités prédictives")

#*****************************************************************************************
#********************MODELE GENERAL DE PREDICTION DE L'EVOLUTION**************************
```

```r
#########################################################################################
cl1 =comp_CL(Resultat_svm)
cl2 =comp_CL(Resultat_rf_int)
cl3 =comp_CL(Resultat_glm)
cl4 =comp_CL(Resultat_deep)
cl5 =comp_CL(Resultat_deep_deep)
cl6 =comp_CL(Resultat_rf_deep)

mod = c("glm-svm" ,"glm-rf","glm-glm" ,"glm-dl","dl-dl","rf-dl")
resl = data.frame(modele = mod,rbind(cl1,cl2,cl3,cl4,cl5,cl6))
ggplot(data=resl, aes(y=score,x = modele,fill=modele) )+
  geom_bar(stat="identity",width=0.5)+
  geom_text(aes(label=score), vjust=-0.3, size=3.5)+
  scale_fill_brewer(palette="Blues")+
  scale_x_discrete(limits=resl$modele[order(resl[,5])])+
  labs(title="Classement des modèles par rapport à leur proximité à chain Ladder")


#########################################################################################
#################################TEMPS D'EXECUTIONS######################################
#########################################################################################

t = Xdata_glm[which(Xdata_glm$numsin =="AW.A333YZDECP4KX87316"),]
temp = Sys.time()
prediction_glm(t,1)
Sys.time()-temp

test = Xdatassyns1[which(Xdatassyns1$numsin =="AW.A333YZDECP4KX87316"),]
temp = Sys.time()
prediction(test,1)
Sys.time()-temp

temp = Sys.time()
prediction_deep(t,1)
Sys.time()-temp

temp = Sys.time()
prediction_rf_int(test_rf ,1)
Sys.time()-temp
```

Etude de la distribution des réserves

```
#*********************************************************************
#   Etude de la distribution des réserves :Méthode de boostrap
#*********************************************************************
#GLMboot
install.packages("ChainLadder")
require(ChainLadder)
GLMboot=glmReserve(as.triangle(TRI_NA),mse.method="bootstrap")
summary(GLMboot)
GLMboot$model
confint(GLMboot$model)
plot(GLMboot$model)
plot(GLMboot$FullTriangle,main="processus de paiement des sinistres")
plot(GLMboot$FullTriangle,lattice=TRUE,main="processus de paiement des sinistres")
a=GLMboot$sims.reserve.mean
b=GLMboot$sims.reserve.pred

#calculer les quantiles de la réserve de pertes prévues
t(apply(GLMboot$sims.reserve.pred,2,quantile,c(0.025,0.25,0.5,0.75,0.975)))

#en moyenne:
t(apply(GLMboot$sims.reserve.mean,2,quantile,c(0.025,0.25,0.5,0.75,0.975)))

#grphe de distribution des réserves:
plot(GLMboot,which=3)
```

Prédiction directe de la charge à l'ultime

```r
#####################Ajout du nombre d'ajustement du sinistre comme variable***************
require(dplyr)
A=count(Xdata1,numsin)
A
b=NULL
C=NULL
for(i in 1:nrow(A)){
  b[[i]]=1:A$n[i]
  C=c(C,b[[i]])
}
Xdata1=mutate(Xdata1,n_ajus=C)
xdat=Xdata1[cumsum(A$n),c(3,32,33)]
xdat
rownames(xdat)=xdat$numsin
xdat=xdat[,-1]
xdat

#############Modèle pour le nombre d'années de développemnt*****************
##########################Le modèle deeolearning de H20****************
library(h2o)
h2o.init()
hex <- Xdaa[,-c(12,14)]
a=sample(1:78206,40000,replace=F)
hexa=as.h2o(hex[a,]);hext=as.h2o(hex[-a,])
fit7 <- h2o.deeplearning(x = 1:11, y = 12, training_frame = hexa, seed=123456)
fit7
summary(fit7)
########################Importance des variables du modèle*******************
h2o.varimp_plot(fit7)
performance = h2o.performance(model = fit7)
print(performance)        #affiche les indicateurs de performance du modèle


############################Prédiction*********************************
pred7=predict(fit7,hexa)
pred7
mean((pred7-hexa$annedev)^2)
plot(as.vector(pred7),as.vector(hexa$annedev))
abline(0,1,col='blue')
length(as.vector(pred7))
length(hexa$annedev)
##############################Le Modèle Random Forest******************
dl <- h2o.randomForest(1:11, y = 12, training_frame = hexa, seed=123456)
dl
summary(dl)
plot(dl)
names(dl)
############################Importance des variables sur la charges*********************
h2o.varimp_plot(dl)
pred8=predict(dl,hexa)
mean((pred8-hexa$annedev)^2)
```

```r
abline(0,1,col='red')
#################ajustement du nouveau modele avec les variables détectées (variables les plu
newrandaom=randomForest(annedev~t2+td2+po2+b2+us3+dureeSin,data=Xdaa,ntree=200)
newrandaom
plot(newrandaom)

#############################Modèle pour le nombre d'ajustement du sinistre*****************
#############################Avec deeplearning de H20*****************************
library(h2o)
h2o.init()
hex <- Xdaa[,-14]
a=sample(1:78206,40000,replace=F)
hexa=as.h2o(hex[a,]);hext=as.h2o(hex[-a,])
ajus7 <- h2o.deeplearning(x = 1:11, y = 12, training_frame = hexa, seed=123456)
ajus7
summary(ajus7)
h2o.varimp_plot(ajus7)
performance = h2o.performance(model = ajus7)
print(performance)
########################Prédiction*********************************
pred=predict(ajus7,hexa)
pred
mean((pred-hexa$ajus)^2)
plot(as.vector(pred),as.vector(hexa$ajus))
abline(0,1,col='blue')
length(as.vector(pred))
length(hexa$ajus)

#####################################Le modèle Random Forest*****************************
dl_ajus <- h2o.randomForest(1:11, y = 13, training_frame = hexa, seed=123456)
dl_ajus
summary(dl_ajus)
plot(dl_ajus)
names(dl_ajus)
h2o.varimp_plot(dl_ajus)
predd=predict(dl_ajus,hexa)
mean((predd-hexa$ajus)^2)    #MSE
abline(0,1,col='red')

#############################Modèle deeplearning pour la charge************************
library(h2o)
h2o.init()
hex <- cbind(Xdaa,Xdaa$Charges)
hex=hex[,-12]
hex=as.h2o(hex)
deep_model <- h2o.deeplearning(x =1:13, y = 14, training_frame = hex, seed=123456)
deep_model
summary
h2o.varimp_plot(deep_model)
performance = h2o.performance(model = deep_model)
print(performance)
```