Royaume du Maroc
Haut Commissariat au Plan
Institut National de Statistique
et d'Économie Appliquée
INSEA

Sujet:

Tarification et Provisionnement

Réalisé par : EL KAMLI Adam
Encadré par : Dr.MARRI Fouad

Table des matières

Première partie

Tarification

L'objectif de ce projet est d'appliquer le cours d'assurance non vie 2 en valorisant dans un premier lieu des contrats **MONO** de l'automobile (*RC automobile usage tourisme*) sous **SAS** pour ensuite calculer les provisions d'une branche d'assurance sous **R**.
La problématique qui se pose alors est Comment trouver le modèle optimale pour calculer la Prime de ce contrat ?

1 Importation ET nettoyage des données

La tarification portera sur les contrats MONO et on va utiliser pour ca deux bases de données :

1. **Production.csv** : cette base de donnée contient
   - *Caractéristique de l'assuré* :
     - Sexe du conducteur.
     - DNA : Date de naissance.
     - DOP : Date d'obtention de permis.
     - CRM : coefficient de réduction majoration.
   - *Caractéristique de la véhicule assuré* :
     - Combustion : Essence ou Gasoil.
     - Puissance fiscale.
     - DMC : Date de mise en circulation.
     - DPEF : Date du premier effet.
   - *Information sur la police* :
     - Exposition.
     - Prime acquise.
     - Numéro de Police.
     - exercice : Année de l'assurance.
2. **Sinistre.csv** : cette base de donnée contient :
   - Numéro de sinistre.
   - Charge : montant du sinistre.
   - Police : Numéro de Police.
   - Exercice : année de l'assurance.

Avant de commencer l'importation des donnée on va créer tout d'abord une librairie qu'on nommera *TARIF* on on organisera tous nos bases de données ainsi que les scriptes.

```
1    LIBNAME Tarif "C:\Users\user\OneDrive\Bureau\Tarification Non-Vie";
2    RUN;
```

1.1 Production.csv

On commence par importer la base **Production** dans la librairie TARIF sous le nom *PROD*.

```
PROC IMPORT DATAFILE = "C:\Users\user\OneDrive\Bureau\Tarification
↪ Non-Vie\production.csv"
    DBMS = CSV
    OUT = TARIF.PROD REPLACE;
    DELIMITER = ";";
    GETNAMES = YES;
RUN;
```

On visualise ensuite les 5 première observation par la commande suivante :

```
    PROC PRINT DATA = TARIF.PROD(OBS=5);
        TITLE 'La base PRODUCTION';
    RUN;
```

La base PRODUCTION

Obs	exposition	prime_acquise	Combustion	puissance_fiscale	SEXE	Exercice	CRM	DOP	DNA	DPEF	DMC	NUMERO_POLICE
1	0.1342465753	421.66849315		.		2007	0	.	.	20040219	.	3425
2	0.8712328767	1754.8372603	E	7	M	2007	0.9	19900524	19641217	20040217	20020125	3426
3	0.1287671233	53.414117647		.	M	2007	1	19900524	19641217	20040217	.	3426
4	1	3146.0551314	G	8	F	2008	0.9	19931230	19750906	20040513	20010510	3447
5	0.6136986301	1927.6273973	G	9	F	2010	1	19820525	19601222	20040521	20080717	3449

Comme on peut observer l'importation à été bien faite et on a pu récupérer nos 12 variables.

1.i Renommer les variables

Pour des raisons de clarification et de lisibilité on va renommer les variables suivante :
- DNA : Date de naissance.
- DMC : Date de Mise en Circulation.
- DOP : Date obtention du permis.
- DPEF : Date du premier effet.

```
    DATA TARIF.PROD;
                set TARIF.PROD(rename = (
                DNA = DATE_DE_NAISSANCE
                DMC = DATE_MISE_EN_CIRCULATION
                DOP = DATE_OBTENTION_PERMIS
                DPEF = DATE_PREMIER_EFFET
                ));
       RUN;
```

La base Production renommé

Obs	exposition	prime_acquise	Combustion	puissance_fiscale	SEXE	Exercice	CRM	DATE_OBTENTION_PERMIS	DATE_DE_NAISSANCE	DATE_PREMIER_EFFET	DATE_MISE_EN_CIRCULATION	NUMERO_POLICE
1	0.1342465753	421.66849315		.		2007	0	.	.	20040219	.	3425
2	0.8712328767	1754.8372603	E	7	M	2007	0.9	19900524	19641217	20040217	20020125	3426
3	0.1287671233	53.414117647		.	M	2007	1	19900524	19641217	20040217	.	3426
4	1	3146.0551314	G	8	F	2008	0.9	19931230	19750906	20040513	20010510	3447
5	0.6136986301	1927.6273973	G	9	F	2010	1	19820525	19601222	20040521	20080717	3449

1.ii Les valeurs manquantes

Pour traiter les valeurs manquantes de notre base de donnée on va utiliser la procédure **MEANS**.

```
1  PROC MEANS DATA = TARIF.PROD
2                  NMISS;
3  RUN;
```

Les valeurs manquantes

The MEANS Procedure

Variable	N Miss
exposition	0
prime_acquise	0
puissance_fiscale	15032
Exercice	0
CRM	61
DATE_OBTENTION_PERMIS	10065
DATE_DE_NAISSANCE	10065
DATE_PREMIER_EFFET	0
DATE_MISE_EN_CIRCULATION	15032
NUMERO_POLICE	0

On stock tout d'abord les valeurs de ce tableau dans une table qu'on va nommer *Missing* et qu'on va transposer pour une meilleur lisibilité dans la table *Missing_ trans*.

```
PROC MEANS DATA = TARIF.PROD NMISS N NOPRINT;
    OUTPUT OUT = TARIF.Missing
    nmiss = /autoname;
RUN;
PROC TRANSPOSE DATA = TARIF.MISSING
                              OUT = TARIF.MISSING_TRANS;
RUN;
```

Pour mieux interpréter ces résultats on va calculer la fréquence de ces valeurs manquantes par rapport a toute les observations de la base de donnée PROD.

```
PROC SQL;
    create table TARIF.MISS as
    select * ,
            COL1 / (select count(*) from TARIF.PROD)*100 as FREQ
    from TARIF.MISSING_TRANS;
QUIT;
/*Ce prochain bloque et pour la visualization de la table
 ↪  TARIF.MISS*/
PROC PRINT DATA = tarif.MISS;
    title 'Valeurs manquante et fréquences';
RUN;
```

Valeurs manquante et fréquences

Obs	_NAME_	COL1	FREQ
1	_TYPE_	0	0.000
2	_FREQ_	145344	100.000
3	exposition_NMiss	0	0.000
4	prime_acquise_NMiss	0	0.000
5	puissance_fiscale_NMiss	15032	10.342
6	Exercice_NMiss	0	0.000
7	CRM_NMiss	61	0.042
8	DATE_OBTENTION_PERMIS_NMiss	10065	6.925
9	DATE_DE_NAISSANCE_NMiss	10065	6.925
10	DATE_PREMIER_EFFET_NMiss	0	0.000
11	DATE_MISE_EN_CIRCULATION_NMiss	15032	10.342
12	NUMERO_POLICE_NMiss	0	0.000

On peut tirer plusieurs informations de cette table :

— Les seuls variables avec des valeurs manquante sont : Puissance_fiscale, CRM, Date de l'obtention du permis, date de naissance et la date de mise en circulation

— La puissance fiscale et la date de mise en circulation ont tous les deux 10% de leurs observations qui sont NULL.

— La date d'obtention de permis et la date de naissance ont 6.925% de leurs observations qui sont NULL.

On peut conclure alors que :

— Le traitement de ces valeurs manquantes par suppression ne nous causera pas de problème vu qu'elles ne sont pas très fréquents .

— Il existe peut être une relation entre date de naissance et date d'obtention de permis ainsi qu'entre date de mise en circulation et ma puissance fiscale.

1.iii Nettoyage

On va commencer par supprimer les valeurs NULL ainsi que les valeurs qui n'ont pas de sens, et pour ca on appliquera les conditions suivantes :

— une Date de Mise en Circulation (négative ou égale à une valeur manquante).

— une Date de naissance (négative ou égale à une valeur manquante).

— une Date obtention du permis (négative ou égale à une valeur manquante).

— une Date du premier effet(négative ou égale à une valeur manquante).

— un CRM égal à une valeur manquante

— une exposition ($<= 0$ ou >1).

— une puissance fiscale (négative ou >100).

```
1     PROC SQL;
2         delete from TARIF.PROD
3         where  DATE_OBTENTION_PERMIS < 0 OR DATE_OBTENTION_PERMIS IS
          ↪  NULL OR
4                         DATE_DE_NAISSANCE < 0 OR DATE_DE_NAISSANCE IS
                          ↪  NULL OR
5                         DATE_PREMIER_EFFET < 0 OR DATE_PREMIER_EFFET IS
                          ↪  NULL OR
6                         DATE_MISE_EN_CIRCULATION < 0 OR
                          ↪  DATE_MISE_EN_CIRCULATION IS NULL OR
7                         DATE_MISE_EN_CIRCULATION < 0 OR
                          ↪  DATE_MISE_EN_CIRCULATION IS NULL OR
8                         crm IS NULL OR
9                         exposition > 1 OR exposition <= 0;
10 QUIT;
```

On la suppression de 10065 valeurs de Date de naissance qui est exactement le nombre de valeurs manquante, donc il n'y avait pas de date négative.

En utilisant la PROC MEANS une autre fois on voit bien que les valeurs manquante de la date d'obtention du permis sont devenu Null ce qui confirme la relation enter ces deux, et par conséquent on va uniquement vérifier la condition de positivité pour la date d'obtention de permis.

Variable	N Miss
exposition	0
prime_acquise	0
puissance_fiscale	4985
Exercice	0
CRM	57
DATE_OBTENTION_PERMIS	0
DATE_DE_NAISSANCE	0
DATE_PREMIER_EFFET	0
DATE_MISE_EN_CIRCULATION	4985
NUMERO_POLICE	0

```
PROC SQL;
    delete from TARIF.PROD
    where  DATE_OBTENTION_PERMIS < 0 ;
QUIT;
```

On va vérifier que la condition de négativité pour la date du premier effet car elle n'a pas de valeurs manquantes.

```
PROC SQL;
      delete from TARIF.PROD
      where  DATE_PREMIER_EFFET < 0 ;
    QUIT;
```

```
PROC SQL;
  delete from TARIF.PROD
  where  DATE_MISE_EN_CIRCULATION < 0 OR
   ↪  DATE_MISE_EN_CIRCULATION IS NULL;
QUIT;
```

On constate après une autre PROC MEANS effectivement chaque ligne de valeurs manquante pour la puissance fiscale à une date de mise en circulation Null est vice vers

ça ce qui indique qu'il n'est plus important de supprimer les valeurs manquantes pour puissance fiscale.
On applique les conditions qui restent ainsi :

```
        PROC SQL;
                delete from TARIF.PROD
                where  crm IS NULL or
                exposition >1 or exposition <=0 or
                puissance_fiscale >100 or puissance_fiscale <0;
        QUIT;

```

Variable	N Miss
exposition	0
prime_acquise	0
puissance_fiscale	0
Exercice	0
CRM	0
DATE_OBTENTION_PERMIS	0
DATE_DE_NAISSANCE	0
DATE_PREMIER_EFFET	0
DATE_MISE_EN_CIRCULATION	0
NUMERO_POLICE	0

Avant de terminer cette section il est important de noter que jusqu'a ici on traiter les valeurs manquantes des variables quantitative, et par conséquent on va créer pour chaque variable catégorielle une variable quantitative qui prend la valeur 1 si la variable originial est manquante, pour ensuite calculer la somme de cette variable indicatric.

```
/*Creationd des variables*/
    data TARIF.PROD;
    set TARIF.PROD;
    if SEXE = "" then SEXE_missing = 1;
    else SEXE_missing = 0;
```

```
        IF COMBUSTION = "" THEN COM_MISSING = 1;
        ELSE COM_MISSING = 0;
run;
/*Calcul de la somme*/
        proc summary data=tarif.prod;
    var com_missing Sexe_missing;
    output out=work.sumC sum=;
    /*Affichage des resultats*/
run;
proc print data = sumC;
        run;
```

On trouve ainsi :

Valeurs manquante et fréquences

Obs	_TYPE_	_FREQ_	COM_MISSING	SEXE_missing
1	0	125235	0	128

On trouve qu'il reste 128 valeurs manquantes de la variabl SEXE qu'on va supprimer.

```

PROC SQL;
        delete from TARIF.prod
        where Sexe_missing = 1 ;
QUIT;
```

On la supression de 128 lignes !

1.iv Conversion des types

Afin de régler le type des dates on va dans un premier lieu convertir les dates en caractères, en créant des nouvelle variables.

```
/*Conversion des types ey création des variables*/
    DATA TARIF.PROD ;
        SET TARIF.PROD;
        DATE_DE_NAISSANCE_CH = PUT(DATE_DE_NAISSANCE, 8.);
        DATE_MISE_EN_CIRCULATION_CH = PUT(DATE_MISE_EN_CIRCULATION, 8.);
        DATE_OBTENTION_PERMIS_CH = PUT(DATE_OBTENTION_PERMIS, 8.);
        DATE_PREMIER_EFFET_CH = PUT(DATE_PREMIER_EFFET, 8.);
    RUN;
/*Suppression des anciennes variables*/
```

```
    DATA TARIF.PROD(DROP = DATE_DE_NAISSANCE DATE_MISE_EN_CIRCULATION
     ↪  DATE_OBTENTION_PERMIS DATE_PREMIER_EFFET);
        SET TARIF.PROD;
    RUN;
```

DATE_DE_NAISSANCE_CH	DATE_MISE_EN_CIRCULATION_CH	DATE_OBTENTION_PERMIS_CH	DATE_PREMIER_EFFET_CH
19641217	20020125	19900524	20040217
19750906	20010510	19931230	20040513
19601222	20080717	19820525	20040521
19651225	20040528	19881103	20040528
19690128	20060808	19900528	20040603

Finalement, pour chaque date on va la convertir en format date en utilisant *substr(trim(left())„)* qui va crée une nouvelle variable de format date à partir de l'ancienne variable de type caractère qu'on va supprimer ensuite .

```
    DATA TARIF.PROD;
                SET TARIF.PROD ;
  /*Extraction de l'année, mois et jours*/
                YEAR = SUBSTR(TRIM(LEFT(DATE_DE_NAISSANCE_CH)),1,4);
                MOIS = SUBSTR(TRIM(LEFT(DATE_DE_NAISSANCE_CH)),5,2);
                DAY = SUBSTR(TRIM(LEFT(DATE_DE_NAISSANCE_CH)),7,2);
  /* Conversion de date_de_naissance en format date*/
                DATE_DE_NAISSANCE = MDY(MOIS,DAY,YEAR);
                FORMAT DATE_DE_NAISSANCE YYMMDD10.;
                RUN;
  /*Suppression de l'ancienne variable*/
                DATA TARIF.PROD (DROP = YEAR MOIS DAY
                 ↪  DATE_DE_NAISSANCE_CH);
                SET TARIF.PROD;
    RUN;
```

On va répéter la même procédure pour tous les autres dates, et on obtient .

DATE_DE_NAISSANCE	DATE_OBTENTION_PERMIS	DATE_MISE_EN_CIRCULATION	DATE_PREMIER_EFFET
1964-12-17	1990-05-24	2002-01-25	2004-02-17
1975-09-06	1993-12-30	2001-05-10	2004-05-13
1960-12-22	1982-05-25	2008-07-17	2004-05-21
1965-12-25	1988-11-03	2004-05-28	2004-05-28
1969-01-28	1990-05-28	2006-08-08	2004-06-03

1.v Création des nouvelles variables

On va créer deux variables importante dans la tarification qui sont :
— Âge du conducteur : Exercice - date de naissance
— Âge du véhicule : Exercice - date de mise en circulation

Mais avant de faire ca on va tout d'abord convertir la variable exercice au type date.

```
    DATA TARIF.PROD;
                SET TARIF.PROD;
                EXERCICE = MDY(1,1,EXERCICE);
                FORMAT EXERCICE YYMMDD10.; /*On transforme exercice à la même format de date que les autres dates YYMMDD10.*/
    RUN;
```

On a alors :

```
DATA TARIF.PROD;
        SET TARIF.PROD;

                AGE_CONDUCTEUR = INTCK('YEAR', DATE_DE_NAISSANCE,EXERCICE);
                AGE_VEHICULE = INTCK('YEAR', DATE_MISE_EN_CIRCULATION,EXERCICE);
                ANCIENNETE_PERMIS = INTCK('YEAR', DATE_OBTENTION_PERMIS, EXERCICE);
                ANCIENNETE_POLICE = INTCK('YEAR', DATE_PREMIER_EFFET, EXERCICE);
run;
```

AGE_CONDUCTEUR	AGE_VEHICULE	ANCIENNETE_PERMIS	ANCIENNETE_POLICE
43	5	17	3
33	7	15	4
50	2	28	6
46	7	23	7
38	1	17	3

1.vi Les doublons

Pour traiter les doublons on va tout d'abord une table *TEST* qui va regrouper les doublons .

```
    proc sql;
                create table TARIF.test as
                select numero_police,exercice, count(*) as repetition
                from TARIF.PROD
                group by numero_police,exercice
```

```
                order by repetition desc;
    QUIT;
```

Et on va éliminer maintenant les doublons par :

```
                PROC SORT DATA = TARIF.PROD NODUPKEY OUT = TARIF.PROD;
                  BY NUMERO_POLICE EXERCICE ;
                RUN;
```

On obtient alors :

Doublons

Obs	NUMERO_POLICE	Exercice	repetition
1	580034	2008-01-01	3
2	985483	2008-01-01	3
3	1013493	2009-01-01	2
4	927595	2008-01-01	2
5	1024408	2011-01-01	2

Maintenant qu'on éliminer les doublons on va supprimer les observations suivantes :
- un Age du véhicule (négative, supérieur à 100 ou égal à une valeur manquante)
- un Age du conducteur (<18 , supérieur à 100 ou égale à une valeur manquante)
- une Ancienneté du permis (négative, supérieur à 80 ou égale à une valeur manquante)
- une Ancienneté de la police (négative ou égale à une valeur manquante)

```

                PROC SQL;
                DELETE FROM TARIF.PROD
                WHERE AGE_CONDUCTEUR <18 OR AGE_CONDUCTEUR >100 OR
                 ↪  AGE_CONDUCTEUR IS NULL;
                RUN;
                proc sql ;
                DELETE FROM TARIF.PROD
                WHERE AGE_VEHICULE <0 OR AGE_VEHICULE>100 OR
                 ↪  AGE_VEHICULE IS NULL ;
                RUN;

                PROC SQL ;
                DELETE FROM TARIF.PROD
                WHERE ANCIENNETE_PERMIS <0 OR ANCIENNETE_PERMIS>80 OR
                 ↪  ANCIENNETE_PERMIS IS NULL;
```

```
RUN;

PROC SQL ;
DELETE FROM TARIF.PROD
WHERE ANCIENNETE_POLICE <0 OR ANCIENNETE_POLICE IS NULL;
RUN;
```

A ce stade pour vérifier davantage la cohérance de nos donnée on va créer une nouvelle variable qu'on va nommer DIFF et elle va représenter l'âge du conducteur moins l'ancienneté de permis ce qui nous donnera l'âge d'obtention du permis de chaque police, ensuite on va vérifier les valeurs de cette variable .

```
/*Creation de la variable*/
PROC SQL;
    CREATE TABLE TARIF.PROD1 AS
    SELECT *,
    AGE_CONDUCTEUR - ANCIENNETE_PERMIS AS DIFF
    FROM TARIF.PROD;
QUIT;
/*Statistique descriptive sur la variable DIFF*/
PROC UNIVARIATE DATA = TARIF.PROD1;
    VAR DIFF;
RUN;
```

On obtient alors :

Extreme Observations			
Lowest		Highest	
Value	Obs	Value	Obs
13	121924	84	51070
16	120858	84	72666
17	109773	85	24872
17	72621	87	68924
17	38402	89	121448

On remarque qu'il existe des polices qu'ils ont obtenu leurs permis a 13 ans ou même 17 ans ce qui n'est pas possible car l'âge légal pour l'obtention du permis d'automobile (Permis type **B**) est de 18 ans au Maroc.
Alors on va supprimer les observations qui ont une DIFF $<$ 18 ans .

```
PROC SQL;
    DELETE FROM TARIF.PROD
    WHERE DIFF < 18;
QUIT;
```

On ainsi supprimer 5 observations !

1.2 Sinistre.csv

On commence par l'importation de la base de donnée

```
PROC IMPORT DATAFILE = "C:\Users\user\OneDrive\Bureau\Tarification
 ↪  Non-Vie\sinistre.csv"
            DBMS = CSV
            OUT = TARIF.SIN REPLACE;
            DELIMITER = ";";
            GETNAMES = YES;
RUN;
```

base SINISTRE

Obs	numero_sinistr	Charge	exercice	Police
1	2.008E14	-5380	2008	3396
2	2.008E14	-3996	2008	3397
3	2.01E13	-1593.48	2010	3398
4	2.007E14	975.18	2007	3401
5	2.011E14	8839.12	2011	3404

On commence par éliminer les sinistres qui ont une charge négative.

```
PROC SQL ;
  DELETE FROM TARIF.SIN
  WHERE CHARGE <0;
RUN;
```

On groupe les variables selon la police et l'exercice en créant une variable comme la somme des charges et une autres qui la somme des nombre de sinistres selon la police et l'exercice.

```
proc sql;
     create table TARIF.SIN as
    select police ,exercice, count(*) as nombre , sum(CHARGE) as
     ↪  somme
    FROM TARIF.SIN
    GROUP BY EXERCICE,POLICE;
QUIT;
```

2.i Valeurs manquantes

On vérifie la présence des valeurs manquantes

```
PROC MEANS DATA = TARIF.SIN
    NMISS;
RUN;
```

base SINISTRE

The MEANS Procedure

Variable	N Miss
numero_sinistr	0
Charge	0
exercice	0
Police	0

1.3 Jointure

Après avoir nettoyer nos bases de données on va procéder par une jointure.
On va faire une jointure Gauche ***LEFT JOIN***, avec la base de production a gauche et on va remplacer les valeurs manquantes du sinistre par 0 car ca veut dire que cette police n'a pas fait d'accident .
Mais avant cette jointure on supprime les variables qui ne sont plus utile de la base production.

```
DATA TARIF.PROD (DROP =  sexe_missing DIFF com_missing
 ↪  DATE_DE_NAISSANCE DATE_OBTENTION_PERMIS DATE_MISE_EN_CIRCULATION
            DATE_PREMIER_EFFET);
        SET TARIF.PROD;
RUN;
```

```
    PROC SQL ;
        CREATE TABLE TARIF.DATA AS
        SELECT P.*, S.*
        FROM TARIF.PROD AS P
        LEFT JOIN TARIF.SIN AS S
        ON P.NUMERO_POLICE = S.POLICE;
        ALTER TABLE TARIF.DATA DROP VAR1;
        UPDATE TARIF.DATA
                SET SOMME=0, NOMBRE=0
                WHERE SOMME = .;
    QUIT;
```

On supprime les valeurs de sinistres qui dépassent la quantile 99%.

```
    proc sql;
        delete from TARIF.DATA
        where somme >66900.0 ;
quit;
```

On supprime la variable Police car elle est répétée. et on obtient alors la table :

Jointure de Prod et Sin

Obs	exposition	prime_acquise	Combustion	puissance_fiscale	SEXE	Exercice	CRM	NUMERO_POLICE	AGE_CONDUCTEUR	AGE_VEHICULE	ANCIENNETE_PERMIS	ANCIENNETE_POLICE	nombre	somme
1	0.671232877	1754.8372603	E	7	M	2007-01-01	0.9	3428	43	5	17	3	0	0.00
2	1	3146.0551314	G	8	F	2008-01-01	0.9	3447	33	7	15	4	0	0.00
3	0.613698631	1927.6273973	G	9	F	2010-01-01	1	3449	50	2	28	6	0	0.00
4	0.452739726	741.04109589	E	6	F	2011-01-01	1	3451	46	7	23	7	1	46990.32
5	0.580821918	1266.2650738	G	7	M	2007-01-01	0	3453	38	1	17	3	0	0.00

On ajoute ensuite les deux variables réponses qui nous intéressent qui sont :

—
$$Frequence = \frac{\text{Nombre de sinistre}}{\text{Exposition}}$$

—
$$Severite = \frac{\text{Somme des sinistres}}{\text{Nombre de sinistres}}$$

```
    PROC SQL;
                ALTER TABLE TARIF.DATA ADD FREQUENCE FLOAT(4);
                ALTER TABLE TARIF.DATA ADD SEVERITE FLOAT (4);

                UPDATE TARIF.DATA
                SET FREQUENCE = NOMBRE / EXPOSITION,
                SEVERITE = SOMME / NOMBRE ;
                UPDATE TARIF.DATA
                SET SEVERITE = 0
                WHERE SEVERITE = .;
    QUIT;
```

2 Exploration des données

2.1 Introduction aux variables

On trace quelque courbe descriptives des variables en exécutant le code suivant :

```
    PROC SGPLOT DATA = TARIF.DATA;
        VBAR  SEXE /STAT = PERCENT
        FILLATTRS = (COLOR = CX4682B4);
RUN;
PROC SGPLOT DATA = TARIF.DATA;
        VBAR  PUISSANCE_FISCALE /STAT = PERCENT
        FILLATTRS = (COLOR = CX4682B4);
RUN;
PROC SGPLOT DATA = TARIF.DATA;
        VBAR  COMBUSTION /STAT = PERCENT
        FILLATTRS = (COLOR = CX4682B4);
RUN;
PROC SGPLOT DATA = TARIF.DATA;
        HISTOGRAM  AGE_VEHICULE /
        FILLATTRS = (COLOR = CX4682B4);
RUN;
PROC SGPLOT DATA = TARIF.DATA;
```

```
        VBAR  AGE_VEHICULE /STAT = PERCENT
        FILLATTRS = (COLOR = CX4682B4);
RUN;
PROC SGPLOT DATA = TARIF.DATA;
        VBAR  NOMBRE /STAT = PERCENT
        FILLATTRS = (COLOR = CX4682B4);
RUN;
PROC SGPLOT DATA = TARIF.DATA;
        histogram  somme /
        FILLATTRS = (COLOR = CX4682B4);
RUN;
```

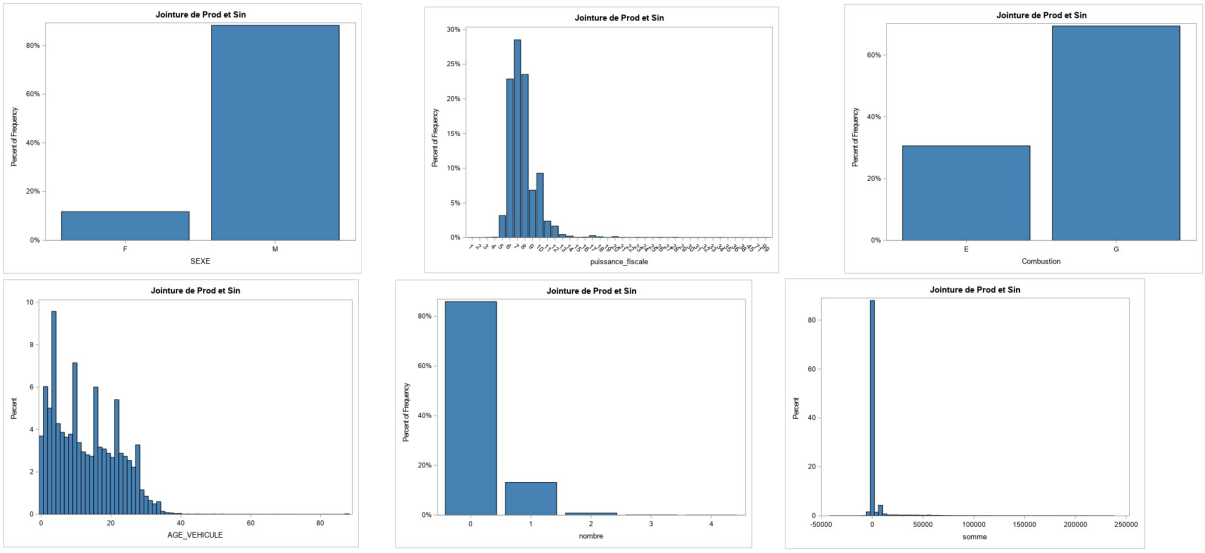


Figure 2.1 – Fréquence des variables dans les deux bases de données

3 Modélisation de la fréquence

3.1 Division de la base de donnée

Pour un meilleur test de la performance de chaque modèle on va diviser notre base de données en deux bases, une pour l'entraînement des modèles et l'autre pour tester la performance.
Vu l'importance de la dominantion du 0 dans la modélisation on va procéder par un échantillonnage stratifié pour garder la même fréquence de 0 .

```
    PROC SORT DATA = TARIF.DATA OUT = TARIF.DATA2;
BY NOMBRE;
RUN;
```

```
proc surveyselect data=tarif.data2 rate=0.8 outall out=TARIF.DATA3
 ↪  seed=1234;
strata NOMBRE;
run;
data TARIF.DATA_train TARIF.DATA_test;
set TARIF.DATA3;
if selected =1 then output TARIF.DATA_train;
else output TARIF.DATA_test;
drop selected;
run;
```

Pour s'assurer de l'échantillonnage on calcul la fréquence des variables dans chaque base de donnée :

```
    proc freq data=tarif.DATA_train;
table NOMBRE;
run;

proc freq data=TARIF.DATA_test;
table NOMBRE;
run;
```

nombre	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	88899	89.24	88899	89.24
1	10202	10.24	99101	99.48
2	488	0.49	99589	99.97
3	25	0.03	99614	100.00
4	4	0.00	99618	100.00

0.5 0.5

nombre	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	22224	89.25	22224	89.25
1	2550	10.24	24774	99.49
2	121	0.49	24895	99.97
3	6	0.02	24901	100.00
4	1	0.00	24902	100.00

Figure 3.1 – Fréquence des variables dans les deux bases de données

3.2 Variable OFFSET

Vu que la modéllisation du nombre de sinistre n'est qu'une étape vers la modélisation de fréquence qui divise le nombre sur l 'exposition, il est tout a fait naturel de considérer l'exposition dans notre régression ce qui est fait par l'introduction de la variable offset qui est dans notre cas l'exposition.
Vu que dans les modélisation qu'on va utiliser on va utiliser la fonction $t \leftarrow ln(t)$ on va caculer le log(exposition).

```
1    PROC SQL ;
2        alter table TARIF.DATA ADD LOGEXPO FLOAT(4);
3        UPDATE TARIF.DATA SET LOGEXPO = log(EXPOSITION);
4        QUIT;
```

3.3 Modèle de Poisson

Le modèle de Poisson est l'un des modèle les plus utiliser pour modéliser les variables de comptage qui est dans notre le nombre de sinistre.

Pour une variable de comptage Y_i qu'on veut expliquer par des variables explicatives $X_1, \ldots, X_p$, le modèle de Poisson se présente sous la forme suivante :

$$Y_i \sim P(\lambda_i), \quad g(\lambda_i) = \beta_0 + \sum_{i=1}^{p} \beta_i X_i$$

Avec :

- $g(.)$: est la fonction *link*.
- $\lambda_i = \mathbb{E}(Y_i|X_i)$

3.i Premier modèle

Pour le premier modèle on va choisir :

- **Variable de comptage** : Nombre de sinistres
- **Variables explicative** :
  - *Variables qualitatives* : Sexe et Combustion
  - *Variables quantitatives* : Puissance Fiscale, Age du conducteur, Age du conducteur, anciennete du permis et le CRM.

On applique le modèle sur notre base d'entraînement avec le code suivant :

```
1      PROC GENMOD DATA = TARIF.DATA_train;
2                   CLASS SEXE COMBUSTION  ;
3                  MODEL NOMBRE = SEXE COMBUSTION PUISSANCE_FISCALE
                    ↪  AGE_VEHICULE AGE_CONDUCTEUR ANCIENNETE_PERMIS
                    ↪  ANCIENNETE_POLICE PUISSANCE_FISCALE CRM/
4                  DIST = POISSON LINK = LOG OFFSET = LOGEXPO;
5                  TITLE ' POISSON1';
6                  OUTPUT out = TARIF.P1 predicted = nombre_predi;
7          STORE TARIF.P1;
8  RUN;
```

On obtient le résultat suivant :

Analysis Of Maximum Likelihood Parameter Estimates

Parameter		DF	Estimate	Standard Error	Wald 95% Confidence Limits		Wald Chi-Square	Pr > ChiSq
Intercept		1	-0.8980	0.0520	-0.9998	-0.7961	298.45	<.0001
SEXE	F	1	0.1713	0.0256	0.1211	0.2215	44.65	<.0001
SEXE	M	0	0.0000	0.0000	0.0000	0.0000	.	.
Combustion	E	1	-0.1036	0.0218	-0.1463	-0.0610	22.65	<.0001
Combustion	G	0	0.0000	0.0000	0.0000	0.0000	.	.
puissance_fiscale		1	0.0150	0.0018	0.0114	0.0186	67.16	<.0001
AGE_VEHICULE		1	-0.0530	0.0013	-0.0556	-0.0504	1552.56	<.0001
AGE_CONDUCTEUR		1	-0.0126	0.0012	-0.0149	-0.0102	105.49	<.0001
ANCIENNETE_PERMIS		1	0.0105	0.0013	0.0080	0.0130	66.72	<.0001
ANCIENNETE_POLICE		1	0.0323	0.0024	0.0275	0.0371	175.00	<.0001
CRM		1	-0.0200	0.0276	-0.0740	0.0340	0.53	0.4670
Scale		0	1.0000	0.0000	1.0000	1.0000		

On constate que toute les variables on une p-valeur inférieure à 0.001 à l'exception de CRM. ce q'uon va essayer de régler dans le deuxième modèle.

3.ii Deuxième modèle

On constate que tout les variables on une *p-valeur* inférieure a un seuil de risque de 1% à l'exception de la variable CRM.
Un premier doute sera la présence d'une multi colinéarité dans le modèle et alors on va calculer la matrice de corrélation entre ces variables.

Pour s'informer mieux de la performance du modèle on a :

```
proc corr data=TARIF.DATA outp=TARIF.corr_matrix ;
    var PUISSANCE_FISCALE AGE_VEHICULE AGE_CONDUCTEUR ANCIENNETE_POLICE
     ↪  ANCIENNETE_PERMIS CRM;
run;
```

Pearson Correlation Coefficients, N = 124520
Prob > |r| under H0: Rho=0

	puissance_fiscale	AGE_VEHICULE	AGE_CONDUCTEUR	ANCIENNETE_POLICE	ANCIENNETE_PERMIS	CRM
puissance_fiscale	1.00000	-0.03423 <.0001	0.06515 <.0001	0.01780 <.0001	0.08957 <.0001	-0.02235 <.0001
AGE_VEHICULE	-0.03423 <.0001	1.00000	-0.05385 <.0001	-0.02508 <.0001	-0.14019 <.0001	0.13721 <.0001
AGE_CONDUCTEUR	0.06515 <.0001	-0.05385 <.0001	1.00000	0.25431 <.0001	0.71683 <.0001	-0.12916 <.0001
ANCIENNETE_POLICE	0.01780 <.0001	-0.02508 <.0001	0.25431 <.0001	1.00000	0.29190 <.0001	-0.31295 <.0001
ANCIENNETE_PERMIS	0.08957 <.0001	-0.14019 <.0001	0.71683 <.0001	0.29190 <.0001	1.00000	-0.16226 <.0001
CRM	-0.02235 <.0001	0.13721 <.0001	-0.12916 <.0001	-0.31295 <.0001	-0.16226 <.0001	1.00000

On remarque que CRM est le plus corrélée avec l'ancienneté de permis.Une solution qu'on peut essayer dans ce cas ca sera de remplacer ces deux variables dans le modèle par le produit des deux, ce qui nous donne :

```

 PROC GENMOD DATA = TARIF.DATA_train ;
                  CLASS SEXE COMBUSTION  ;
                 MODEL NOMBRE = SEXE COMBUSTION PUISSANCE_FISCALE
                  ↪  AGE_VEHICULE AGE_CONDUCTEUR ANCIENNETE_POLICE*CRM
                  ↪  ANCIENNETE_PERMIS /
                 DIST = POISSON LINK = LOG OFFSET = LOGEXPO;
                 TITLE ' POISSON2';
                 OUTPUT OUT = TARIF.P2 predicted=predicted;
RUN;
```

Analysis Of Maximum Likelihood Parameter Estimates

Parameter		DF	Estimate	Standard Error	Wald 95% Confidence Limits		Wald Chi-Square	Pr > ChiSq
Intercept		1	-0.9279	0.0459	-1.0178	-0.8380	409.15	<.0001
SEXE	F	1	0.1720	0.0256	0.1217	0.2222	45.02	<.0001
SEXE	M	0	0.0000	0.0000	0.0000	0.0000	.	.
Combustion	E	1	-0.0975	0.0218	-0.1402	-0.0548	20.02	<.0001
Combustion	G	0	0.0000	0.0000	0.0000	0.0000	.	.
puissance_fiscale		1	0.0151	0.0018	0.0116	0.0187	70.90	<.0001
AGE_VEHICULE		1	-0.0533	0.0013	-0.0560	-0.0507	1585.20	<.0001
AGE_CONDUCTEUR		1	-0.0124	0.0012	-0.0148	-0.0101	103.75	<.0001
ANCIENNETE_POLIC*CRM		1	0.0456	0.0029	0.0398	0.0513	240.86	<.0001
ANCIENNETE_PERMIS		1	0.0109	0.0013	0.0084	0.0134	72.89	<.0001
Scale		0	1.0000	0.0000	1.0000	1.0000		

On constate que maintenant toutes les variables contribue significativement à la régression, à l'addition d'une amélioration de la performance du modèle de la régression.

3.iii Troisième modèle

L'une des caractéristiques du modèle de Poisson est que l'espérance de lal variable de comptage est une fonction montone par rapport a chaque variable, pour d'assurer de ce comportement on va tracer la fréquence par rapport a nos variables explicative pour étudier la monotonie de leurs relations.

D'aprés ces graphes on va essayer de grouper l'âge du conducteur en 3 catégorie :

1. Entre 18 et 30.
2. Entre 30 et 80.
3. supérieure à 80.

On va faire ca avec le code suivant :

```
data TARIF.DATA_TRAIN_G;
    set TARIF.DATA_TRAIN;

    if age_conducteur >= 18 AND  age_conducteur <30 then age_group =
     ↪  '[18-30[';
    else if age_conducteur >= 30 and age_conducteur < 80 then age_group
     ↪  = '[30-80[';
    else  age_group = '>+80';
run;
```

Ensuite en applique le modèle de Poisson et obtient :

```
     PROC GENMOD DATA = TARIF.DATA_train_g plots = all;
                 CLASS SEXE COMBUSTION age_group    ;
                MODEL NOMBRE = SEXE COMBUSTION PUISSANCE_FISCALE
                 ↪  anciennete_permis anciennete_police age_vehicule
                 ↪  age_group CRM /
                DIST = POISSON LINK = LOG OFFSET = LOGEXPO;
```

	Poisson 1	Poisson 2	Poisson 3
AIC	69590.4647	69549.2038	69590.1312
AICC	69590.4665	69549.2052	69590.1330
BIC	69676.0466	69625.2766	69675.7131
Déviance	51 148.9282	47765.8563	47804.7837

Table 3.1 – Caption

```
5                   TITLE ' POISSON3';
6                   OUTPUT OUT = TARIF.P3 predicted=predicted;
7       RUN;
```

Analysis Of Maximum Likelihood Parameter Estimates

Parameter		DF	Estimate	Standard Error	Wald 95% Confidence Limits		Wald Chi-Square	Pr > ChiSq
Intercept		1	-1.3836	0.0288	-1.4401	-1.3270	2301.28	<.0001
SEXE	F	1	0.1813	0.0256	0.1311	0.2315	50.09	<.0001
SEXE	M	0	0.0000	0.0000	0.0000	0.0000	.	.
Combustion	E	1	-0.0983	0.0218	-0.1411	-0.0556	20.33	<.0001
Combustion	G	0	0.0000	0.0000	0.0000	0.0000	.	.
puissance_fiscale		1	0.0154	0.0018	0.0119	0.0189	73.99	<.0001
ANCIENNETE_PERMIS		1	0.0037	0.0009	0.0020	0.0054	17.99	<.0001
AGE_VEHICULE		1	-0.0542	0.0013	-0.0568	-0.0516	1645.74	<.0001
age_group	>+80	1	-0.4582	0.1393	-0.7313	-0.1852	10.82	0.0010
age_group	[18-30[	1	0.2874	0.0364	0.2160	0.3587	62.34	<.0001
age_group	[30-80[	0	0.0000	0.0000	0.0000	0.0000	.	.
CRM*ANCIENNETE_POLIC		1	0.0447	0.0029	0.0390	0.0505	233.35	<.0001
Scale		0	1.0000	0.0000	1.0000	1.0000		

On peut Voir que toutes les variables sont statistiquement différente de 0 à un seuil de 1% à l'exception de la derière catégorie d'âge groupe et que le changement de monotonie est effectivement apparent dans la différence de signe des estimateurs des classes de age_group

3.iv Comparaison des modèles

On peut lire sur la sortie de chacun des codes précédent le AIC, AICC, BIC et la déviance du modèle qu'on regroupera dans le tableau suivant :

d'après la comparaison entre ces 3 modèle on trouve que le meilleur modèle est Poisson 2 avec la plus petite AIC, BIC et Déviance.

3.4 Modèle ZIP (Zero Inflatin Poisson)

d'après le graphe de la variable Nombre on peut constater qu'il existe un grand nombre de police qui n'ont fait aucun sinistre ce qui fait que la proportions de nombre=0 est dominante.

Soit Y une variable de comptage qui suit un modèle ZIP. La probabilité que Y prenne

une valeur spécifique peut être décrite par :

$$\begin{cases} \pi + (1-\pi)e^{-y}, & \text{si} \quad y = 0 \\ (1-\pi)\dfrac{\lambda^y e^{-\lambda}}{y!}, & \text{si} \quad y \neq 0 \end{cases}$$

```
proc genmod data=TARIF.DATA_TRAIN plots = all;
   Class SEXE combustion  ;
    Model nombre = SEXE COMBUSTION AGE_CONDUCTEUR
     ↪  anciennete_police*CRM PUISSANCE_FISCALE AGE_VEHICULE
     ↪  ANCIENNETE_PERMIS PUISSANCE_FISCALE  /
    dist = zip link = log offset=logexpo ;
   zeromodel / link = logit ;
   title "ZIP2 ";
   ods output modelfit = zip2;
run;
```

Analysis Of Maximum Likelihood Parameter Estimates

Parameter		DF	Estimate	Standard Error	Wald 95% Confidence Limits		Wald Chi-Square	Pr > ChiSq
Intercept		1	-0.9279	0.0459	-1.0178	-0.8380	409.15	<.0001
SEXE	F	1	0.1720	0.0256	0.1217	0.2222	45.02	<.0001
SEXE	M	0	0.0000	0.0000	0.0000	0.0000	.	.
Combustion	E	1	-0.0975	0.0218	-0.1402	-0.0548	20.02	<.0001
Combustion	G	0	0.0000	0.0000	0.0000	0.0000	.	.
AGE_CONDUCTEUR		1	-0.0124	0.0012	-0.0148	-0.0101	103.75	<.0001
puissance_fiscale		1	0.0151	0.0018	0.0116	0.0187	70.90	<.0001
AGE_VEHICULE		1	-0.0533	0.0013	-0.0560	-0.0507	1585.20	<.0001
ANCIENNETE_PERMIS		1	0.0109	0.0013	0.0084	0.0134	72.89	<.0001
ANCIENNETE_POLIC*CRM		1	0.0456	0.0029	0.0398	0.0513	240.86	<.0001
Scale		0	1.0000	0.0000	1.0000	1.0000		

Analysis Of Maximum Likelihood Zero Inflation Parameter Estimates

Parameter	DF	Estimate	Standard Error	Wald 95% Confidence Limits		Wald Chi-Square	Pr > ChiSq
Intercept	1	-22.3861	3783.518	-7437.94	7393.172	0.00	0.9953

On constate que dans ce modèle de ZIP la p-valeur de l'intercept est de 0.9953 ce qui est très loin d'être statistiquement significative.
Par conséquent, on élimine le modèle ZIP dans la suite.

3.5 Modèle binomiale négative

On va alors appliquer ce modèle en éxécutant le code suivant :

```
proc genmod data=Tarif.data_train ;
class Combustion SEXE;
model nombre = puissance_fiscale Combustion
 ↪  anciennete_police*CRM  age_conducteur age_vehicule  SEXE
 ↪  anciennete_permis
```

```
                / dist=negbin link=log;
        output out=TARIF.NB1 pred=pred_nombre;
    run;
```

On obtient alors :

Analysis Of Maximum Likelihood Parameter Estimates

Parameter		DF	Estimate	Standard Error	Wald 95% Confidence Limits		Wald Chi-Square	Pr > ChiSq
Intercept		1	-1.6054	0.0448	-1.6933	-1.5176	1283.88	<.0001
puissance_fiscale		1	0.0086	0.0016	0.0054	0.0118	27.79	<.0001
Combustion	E	1	-0.0908	0.0217	-0.1334	-0.0481	17.42	<.0001
Combustion	G	0	0.0000	0.0000	0.0000	0.0000	.	.
ANCIENNETE_POLIC*CRM		1	0.0831	0.0025	0.0782	0.0880	1113.52	<.0001
AGE_CONDUCTEUR		1	-0.0112	0.0012	-0.0135	-0.0088	86.60	<.0001
AGE_VEHICULE		1	-0.0577	0.0013	-0.0603	-0.0551	1909.88	<.0001
SEXE	F	1	0.2659	0.0256	0.2156	0.3161	107.69	<.0001
SEXE	M	0	0.0000	0.0000	0.0000	0.0000	.	.
ANCIENNETE_PERMIS		1	0.0170	0.0013	0.0145	0.0195	180.94	<.0001
Dispersion		0	0.0000	0.0000	0.0000	0.0000		

On constate que toutes les variables on une p-valeur inférieure à 0.001 ce qui signifie qu'ils sont tous significativement différentes de 0.

3.6 Modèle ZINB

```
   proc genmod data=TARIF.DATA_TRAIN plots = all;
Class SEXE combustion  ;
Model nombre = SEXE COMBUSTION AGE_CONDUCTEUR  AGE_VEHICULE
 ↪  ANCIENNETE_PERMIS PUISSANCE_FISCALE anciennete_police*CRM /
dist = ziNB link = log offset=logexpo ;
zeromodel / link = logit ;
title "ZINB ";
ods output modelfit = zinb;
run;
```

Analysis Of Maximum Likelihood Parameter Estimates

Parameter		DF	Estimate	Standard Error	Wald 95% Confidence Limits		Wald Chi-Square	Pr > ChiSq
Intercept		1	-0.9222	0.0464	-1.0131	-0.8313	395.44	<.0001
SEXE	F	1	0.1723	0.0257	0.1219	0.2226	45.02	<.0001
SEXE	M	0	0.0000	0.0000	0.0000	0.0000	.	.
Combustion	E	1	-0.0977	0.0218	-0.1405	-0.0550	20.07	<.0001
Combustion	G	0	0.0000	0.0000	0.0000	0.0000	.	.
AGE_CONDUCTEUR		1	-0.0125	0.0012	-0.0149	-0.0101	104.06	<.0001
AGE_VEHICULE		1	-0.0533	0.0013	-0.0559	-0.0506	1578.13	<.0001
ANCIENNETE_PERMIS		1	0.0110	0.0013	0.0085	0.0135	73.75	<.0001
puissance_fiscale		1	0.0144	0.0020	0.0105	0.0183	52.51	<.0001
ANCIENNETE_POLIC*CRM		1	0.0456	0.0030	0.0399	0.0514	239.06	<.0001
Dispersion		1	0.0019	0.0427	0.0000	4.098E16		

as estimated by maximum likelihood.

Analysis Of Maximum Likelihood Zero Inflation Parameter Estimates

Parameter	DF	Estimate	Standard Error	Wald 95% Confidence Limits		Wald Chi-Square	Pr > ChiSq
Intercept	1	-6.3620	1.2206	-8.7542	-3.9697	27.17	<.0001

	Binomiale Négative	ZINB
AIC	68321.0622	69555.7561
AICC	68321.0640	69555.7583
BIC	68406.6441	69650.8471
Déviance	46535.7147	69535.7561

Table 3.2 – Caption

3.7 Comparaison des modèles

On lisant les valeurs de AIC, AICC, BIC et la déviance qu'on regroupe dans le tableau suivant :

On constate d'après ces indicateurs que le meilleur modèle entre ces deux est le modèle binomial négative avec un AIC, AICC, BIC et Déviance plus petite que le modèle ZINB

3.8 Modèle finale

Pour le choix du modèle finale entre le modèle de Poisson2 ou le modèle binomial négative, on va procéder par une comparaison entre les résidus de ces deux modèles sur la base de donnée Test.

Pour applique le modèle de Poisson et le modèle Binomial négative, on calcule les valeurs prédite du nombre de sinistre selon les deux modèles avant de calculer les résidus est de tracer le graphe de ces derniers.

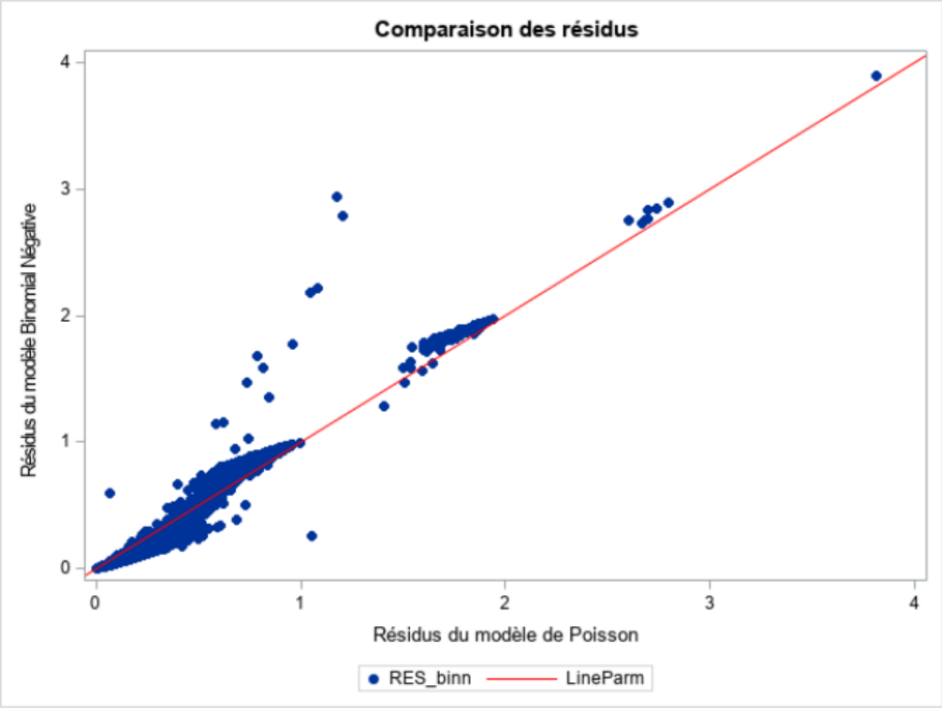
```
/*Création des variables ,calcul des résidus qu'on stock dans la table DATA_TEST_p2 */
  DATA TARIF.DATA_TEST_P2;
          SET TARIF.DATA_TEST;
         LN = -0.9279+0.1720*(SEXE ='F')-0.0975*(COMBUSTION='E')+0.0151*PUISSANCE_FISCALE-0.0533*AGE_VEHICUL
 +0.0456*ANCIENNETE_POLICE*CRM;
         EXP = EXP(LN);
         BIN = -1.6054+0.2659*(SEXE ='F')-0.0908*(COMBUSTION='E')+0.0086*

  PUISSANCE_FISCALE-0.0577*AGE_VEHICULE-0.0112*AGE_CONDUCTEUR+0.0170*ANCIENNETE_PE
         BINN = EXP(BIN);
         RES_pois = ABS(EXP-nombre);
         RES_binn = ABS(BINN-nombre);
RUN;
/*Pour tracer le graphe des résidus*/
proc sgplot data=tarif.data_test_p2;
    scatter x=res_pois y=res_binn / markerattrs=(symbol=circlefilled);
    lineparm x=0 y=0 slope=1 / lineattrs=(color=red);
```

	Poisson 2	Binomial Négative
AIC	69549.2038	68321.0622
AICC	69549.2052	68321.0640
BIC	69625.2766	68406.6441
Déviance	47765.8563	46535.7147

Table 3.3 – Caption

```
    xaxis label="Résidus du modèle de Poisson";
    yaxis label="Résidus du modèle Binomial Négative";
    title "Comparaison des résidus";
run;
```



On remarque que la majorité des points se trouve sur ou près de la ligne rouge ce qui indique que pour plusieurs observations, les deux modèles produisent des résidus similaires. Il y a aussi plusieurs points au-dessus de la ligne rouge, en particulier pour les résidus élevés et par conséquent le modèle Binomial négative a tendance a surévaluer les valeurs résiduelles par rapport au modèle du Poisson pour ces cas.
Cependant, il y a faible cluster de points car un grand nombre de points sont très proche du point (0,0), et alors les deux modèles on des résidus faibles pour ces observations.
A l'addition de ce graphe et de son interprétation on compare les valeurs des 4 indicateurs usuels :

Ainsi le modèle finale pour la variable nombre est :

$$\begin{aligned} nombre = exp(&-1.6054 + 0.2659 * (SEXE =' F') - 0.0908 * (COMBUSTION =' E') \\ &+ 0.0086 * PUISSANCE_FISCALE - 0.0577 * AGE_VEHICULE \\ &- 0.0112 * AGE_CONDUCTEUR + 0.0170 * ANCIENNETE_PERMIS \\ &+ 0.0831 * ANCIENNETE_POLICE * CRM) \end{aligned}$$

4 Modélisation de la sévérité

4.1 Modèle de Gamma

Le modèle de Gamma

1.i Premier modèle

```
1
2  proc genmod data=TARIF.data_TRAIN;
3  Class sexe  CombuStion ;
4  Model severitE = sexe  CombUStion PUISSANCE_FISCALE AGE_VEHICULE
    ↪  anciennete_police ANCIENNETE_PERMIS  age_CONDUCTEUR CRM /
5  dist = gamma link = log  ;
6  ods output modelfit = Gam1;
7  run;
```

Analysis Of Maximum Likelihood Parameter Estimates

Parameter		DF	Estimate	Standard Error	Wald 95% Confidence Limits		Wald Chi-Square	Pr > ChiSq
Intercept		1	9.4808	0.0597	9.3637	9.5979	25197.9	<.0001
SEXE	F	1	-0.1540	0.0266	-0.2062	-0.1018	33.42	<.0001
SEXE	M	0	0.0000	0.0000	0.0000	0.0000	.	.
Combustion	E	1	-0.0099	0.0226	-0.0541	0.0344	0.19	0.6618
Combustion	G	0	0.0000	0.0000	0.0000	0.0000	.	.
puissance_fiscale		1	-0.0124	0.0043	-0.0208	-0.0039	8.29	0.0040
AGE_VEHICULE		1	0.0182	0.0013	0.0156	0.0208	186.60	<.0001
ANCIENNETE_POLICE		1	-0.0069	0.0027	-0.0123	-0.0016	6.47	0.0110
ANCIENNETE_PERMIS		1	-0.0057	0.0013	-0.0082	-0.0031	19.12	<.0001
AGE_CONDUCTEUR		1	0.0034	0.0012	0.0010	0.0058	7.57	0.0059
CRM		1	0.0488	0.0260	-0.0022	0.0999	3.52	0.0608
Scale		1	1.0069	0.0122	0.9833	1.0311		

1.ii Deuxième modèle

Pour remédier aux problèmes du premier modèle on va dans un premier lieu multiplier age_conducteur avec la variable dont elle est la plus corrélée qui est anciennete_permis et ensuite vu la grande valeur de puissance fiscale. on va tranformer en appliquante le logarithme népérien.

```
    data TARIF.DATA_TRAIN;
    set TARIF.DATA_TRAIN;
    transformed_PF= log(PUISSANCE_FISCALE);
run;

```

```
   proc genmod data=TARIF.DATA_TRAIN plots = all ;
class  sexe combustion;
model Cout_moyen=  combustion sexe AGE_VEHICULE  AGE_CONDUCTEUR
 anciennete_permis
 transformed_pf anciennete_police*CRM /
dist =gamma link=log  ;
title'Ajustement par loi de GAMMA';
ods output modelfit=Gam;
output out=tarif.G6 pred=predicted_v;
run;

```

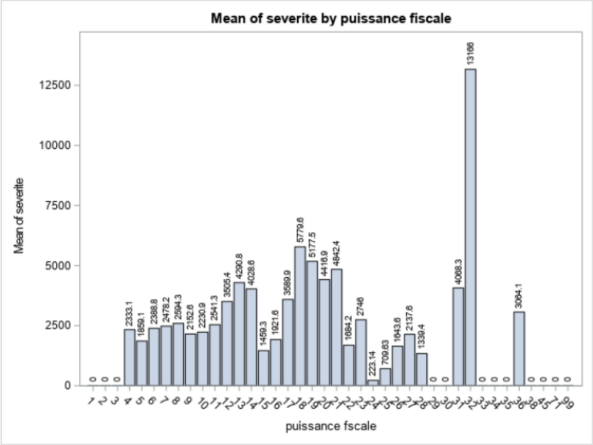
Analysis Of Maximum Likelihood Parameter Estimates

Parameter		DF	Estimate	Standard Error	Wald 95% Confidence Limits		Wald Chi-Square	Pr > ChiSq
Intercept		1	9.7295	0.0957	9.5419	9.9171	10332.6	<.0001
SEXE	F	1	-0.1587	0.0258	-0.2092	-0.1082	37.90	<.0001
SEXE	M	0	0.0000	0.0000	0.0000	0.0000	.	.
AGE_VEHICULE		1	0.0185	0.0013	0.0160	0.0211	199.28	<.0001
AGE_CONDUCTEUR		1	0.0033	0.0012	0.0009	0.0057	7.05	0.0079
ANCIENNETE_PERMIS		1	-0.0058	0.0013	-0.0084	-0.0033	20.06	<.0001
transformed_PF		1	-0.1512	0.0422	-0.2340	-0.0685	12.83	0.0003
ANCIENNETE_POLIC*CRM		1	-0.0077	0.0033	-0.0142	-0.0012	5.35	0.0207
Scale		1	1.0067	0.0122	0.9831	1.0308		

On constate alors une diminution de la p-valeur de toute les variables qui sont maintenant tous siginificativement différent de 0 à un seuil de 5%.

1.iii Troisième modèle

Si on essaye de visualizer la relation entre severite et puissance fiscale on trouve



Alors on propose la catégorisation de puissance fiscale de la façon suivante :

— Puissance fiscale inférieure à 8.
— Puissance fiscale entre 8 et 10.
— Puissance fiscale entre 10 et 14.
— Puissance fiscale supérieur à 14.

```
    data TARIF.DATA_TRAIN_svpf;
    set TARIF.DATA_TRAIN;

    /* Conditional logic to create age_group variable */
    if puissance_fiscale < 8    then PF_group = '<8';
        else if puissance_fiscale >= 8 and puissance_fiscale < 10 then
        ↪  pf_group = '[8-10[';
        else if puissance_fiscale >= 10 and puissance_fiscale < 14 then
        ↪  pf_group= '[10-14[' .
    else  PF_group = '>=14';
run;
```

Ainsi après la création de cete nouvelle variable on applique le modèle de GAMMA.

```
    proc genmod data=TARIF.data_TRAIN_svpf;
 Class sexe  CombuStion pf_group ;
 Model severitE = sexe  CombUStion pf_group AGE_VEHICULE
 ↪  anciennete_police ANCIENNETE_PERMIS*age_CONDUCTEUR CRM /
 dist = gamma link = log  ;
 ods output modelfit = Gam1;
 run;
```

On trouve alors :

	Gamma 1	Gamma2	Gamma 3
AIC	224509.5268	224508.6649	251427.5854
AICC	224509.5476	224508.6785	251427.6011
BIC	224582.2299	224566.8274	251493.7379
Déviance	12163.7543	12166.8713	17573.2491

Analysis Of Maximum Likelihood Parameter Estimates

Parameter		DF	Estimate	Standard Error	Wald 95% Confidence Limits		Wald Chi-Square	Pr > ChiSq
Intercept		1	9.8072	0.0335	9.7415	9.8729	85591.5	<.0001
SEXE	F	1	-0.1196	0.0292	-0.1768	-0.0625	16.83	<.0001
SEXE	M	0	0.0000	0.0000	0.0000	0.0000	.	.
Combustion	E	1	-0.1126	0.0248	-0.1612	-0.0640	20.65	<.0001
Combustion	G	0	0.0000	0.0000	0.0000	0.0000	.	.
PF_group	>=24	1	0.2299	0.1154	0.0036	0.4561	3.97	0.0464
PF_group	[4-18[	0	0.0000	0.0000	0.0000	0.0000	.	.
AGE_VEHICULE		1	0.0219	0.0014	0.0192	0.0247	243.27	<.0001
ANCIENNETE_POLICE		1	-0.0108	0.0029	-0.0165	-0.0052	14.11	0.0002
ANCIENNET*AGE_CONDUC		1	-0.0001	0.0000	-0.0001	-0.0000	24.76	<.0001
CRM		1	0.1268	0.0283	0.0713	0.1823	20.06	<.0001
Scale		1	0.7796	0.0088	0.7624	0.7971		

On constate qu'on a diminution de la p-valeur de la puissance fiscale pour avoir maintennat toutes les variables significativement différentes de 0 avec un sueil de 5%.

1.iv comparaison des modèles

Pour comparer entre ces trois modèle de Gamma on va utiliser les 4 indicateurs usuels, AIC, AICC ,BIC et la Déviance.

D'après ces indicateurs on peut conclure que le modèle Gamma 2 est le meilleur .

4.2 Modèle log-normal

```
    proc genmod data=TARIF.DATA_TRAIN plots = all;
    class SEXE COMBUSTION;
    model COUT_MOYEN = sexe combustion AGE_VEHICULE AGE_CONDUCTEUR
     ↪  anciennete_police ANCIENNETE_PERMIS PUISSANCE_FISCALE CRM  /
     ↪  dist=normal link=log;
    output out=TARIF.LN1 p=predicted PRED=PREDICTED_V;
run;
```

Analysis Of Maximum Likelihood Parameter Estimates

Parameter		DF	Estimate	Standard Error	Wald 95% Confidence Limits		Wald Chi-Square	Pr > ChiSq
Intercept		1	8.4221	0.5932	7.2595	9.5846	201.60	<.0001
SEXE	F	1	0.0508	0.3065	-0.5499	0.6515	0.03	0.8683
SEXE	M	0	0.0000	0.0000	0.0000	0.0000	.	.
Combustion	E	1	-0.1766	0.2641	-0.6943	0.3411	0.45	0.5038
Combustion	G	0	0.0000	0.0000	0.0000	0.0000	.	.
AGE_VEHICULE		1	-0.0315	0.0161	-0.0632	0.0001	3.82	0.0505
AGE_CONDUCTEUR		1	-0.0076	0.0136	-0.0343	0.0192	0.31	0.5793
ANCIENNETE_POLICE		1	0.0425	0.0189	0.0055	0.0796	5.06	0.0245
ANCIENNETE_PERMIS		1	0.0052	0.0149	-0.0241	0.0345	0.12	0.7280
puissance_fiscale		1	0.0044	0.0154	-0.0258	0.0346	0.08	0.7763
CRM		1	-0.0723	0.3486	-0.7555	0.6109	0.04	0.8356
Scale		0	91513.30	0.0000	91513.30	91513.30		

Aucune variable n'est significativement différente de 0, et les 4 indicateurs sont dans l'ordre de 2 millions ce qui implique que ce modèle n'est pas du tout adéquat pour nos données.

4.3 Le modèle finale

Le modèle Finale de la sévérité est alors le modèle GAMMA4 ainsi on a le modèle prend la forme suivante :

5 Calcul de la prime

On sait que :

$$prime_1 = severite \times frequence$$

on calcul alors la prime sur notre base d'entraînement par le code suivant :

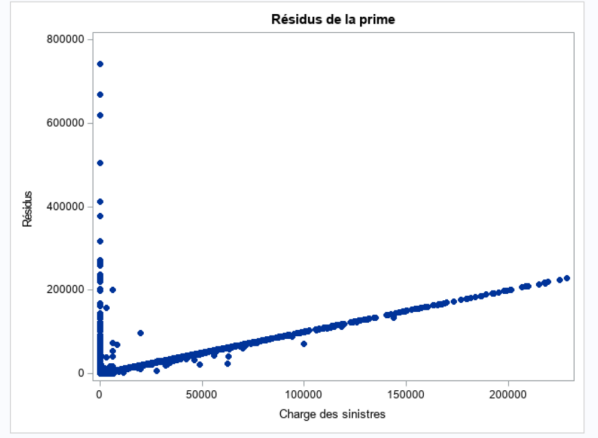
```
      DATA TARIF.DATA_Test;
         SET TARIF.DATA_test;

                BIN = -1.6054+0.2659*(SEXE
                 ↪  ='F')-0.0908*(COMBUSTION='E')+0.0086*PUISSANCE_FISCALE-0.0577*AGE

        frequence_predite = EXP(BIN)/ exposition;
        SEV =9.7295-0.1587*(SEXE =
         ↪  'F')+0.0185*age_vehicule-0.0077*anciennete_police*CRM-0.0058*anciennete_p
         ↪  transformed_pf;
        Severite_predite = EXP(sev);
        PRIME_predite = Severite_predite*frequence_predite;
        Res = abs(somme- Prime_predite);
        Res2 = abs(prime_acquise-prime_predite);
RUN;
```

On trace dans un premier lieu les résidus de la prime pour chaque charge :

```
    proc sgplot data=tarif.data_test;
    scatter x=somme y=Res/ markerattrs=(symbol=circlefilled);
    xaxis label="Charge des sinistres";
    yaxis label="Résidus";
    title "Résidus de la prime";
run;
```



On constate que les résidus augmente avec l'augmentation des sinsitres ce qui est logique vu qu'on a enlevé les grands sinistres dans notre modélisation.
On peut aussi voir le nombre de sinistres prédit etle comparer avec le nombre exacte dans la base de test avec le code suivant :

```
     proc sql;
CREATE TABLE tarif.resume(
 nombre_total INT ,
 nombre_model INT
 );
 INSERT INTO tarif.resume (nombre_total, nombre_model) VALUES
 (0, 0);
 UPDATE tarif.resume set
 nombre_total = (select sum(nombre) from tarif.data_test),
 nombre_model = (select sum(frequence_predite*exposition) from
  ↪  tarif.data_test);

 quit;
 proc print data = tarif.resume;
 run;
```

Obs	nombre_total	nombre_model
1	2814	2800.86

Maintenant qu'on s'assurer de l'adéquation du modèle, on calcul les primes en Maintenant on va recalculer les prime en prenant en considération les grands sinistres. tel que :

$$Prime = Prime_1 + \mathbb{P}(Grands_sinistres) \times \mathbb{E}(Montant)$$

```
    proc sql;
CREATE TABLE tarif.cot (cot float(4));
INSERT INTO cot (cot) VALUES (0);
UPDATE Tarif.cot set
cot = (select mean(somme)*count(*) from tarif.data where somme >66900
 ↪  )/(select count(*) from tarif.data);
quit;
```

On trouve alors que la cotisation est :

$$Cot = 14165.06$$

5.1 La prime finale

On applique maintenant notre formule totale avec la considération de la cotisation pour calculer les primes, on trouve :

nombre	somme	FREQUENCE	SEVERITE	LOGEXPO	EXN	frequence_predite	SEV	transformed_pf	Severite_predite	PRIME_predite
0	0.00	0.00000	0.00	-0.13785	-1.89273	0.17293	9.83322	1.94591	18642.96	17388.94
0	0.00	0.00000	0.00	0.00000	-1.49004	0.22536	9.68242	2.07944	16033.24	17778.37
0	0.00	0.00000	0.00	-0.48825	-0.96290	0.62210	9.55146	2.19722	14065.16	22915.05
1	46990.32	2.48299	46990.32	-0.90946	-1.32510	0.65992	9.65441	1.79176	15590.38	24453.46
0	0.00	0.00000	0.00	-0.54331	-1.73950	0.30234	9.76351	1.94591	17387.62	19422.11

Deuxième partie

Provisionnement

Pour calculer le provisionnement sous R, on va utiliser le package *ChainLadder*, qu'on install avec les libraey readxl.

```
    library(readxl)
library(ChainLadder)
```

1 Importation et exploration des données

1.1 Importation

On commence par importer les données :

```
    Prov <- read_excel("C:\\Users\\user\\OneDrive\\Bureau\\Assurance non
    ↪  vie\\provisions.xlsx")
    Prov <- as.data.frame(Prov)
```

Pour avoir la forme standard des tableaux de provisionnement on va faire des petite modification on rendant la première colonne comme indice des lignes

```
    rownames(Prov) <-  Prov[,1]
Prov = Prov[2:length(Prov[1,])]
head(Prov)
```

Description: df [6 × 10]

	0 <dbl>	1 <dbl>	2 <dbl>	3 <dbl>	4 <dbl>	5 <dbl>	6 <dbl>	7 <dbl>
2014	23763...	12111...	545883	313790	167151	80072	39235	16030
2015	25762...	15371...	662445	342694	188799	77047	35042	17199
2016	27632...	16402...	688959	364199	177108	78169	48371	26377
2017	27796...	16985...	661401	321434	162578	84581	54450	*NA*
2018	28432...	16736...	624401	299473	176842	106296	*NA*	*NA*
2019	29623...	16202...	591932	347434	238375	*NA*	*NA*	*NA*

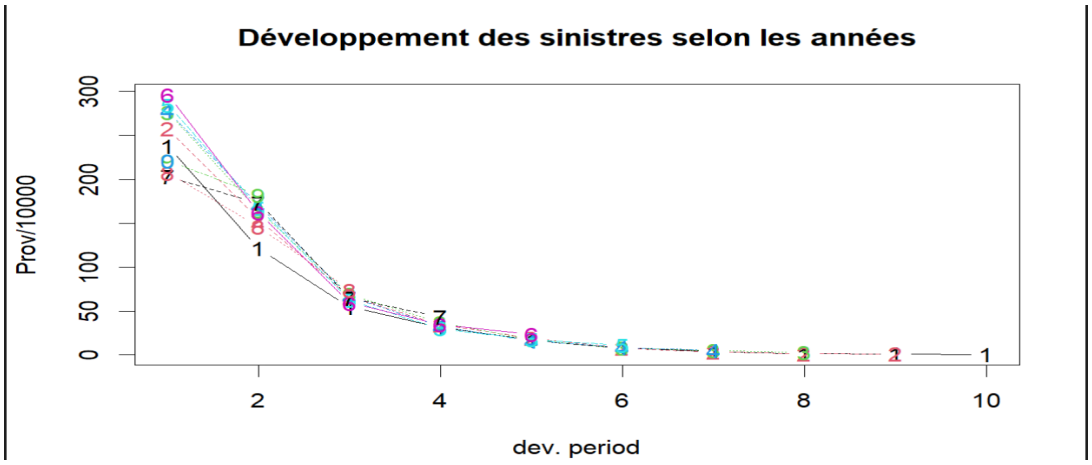
On transforme ensuite les données en forme triangulaire pour la compatibilité avec la package ChainLadder.

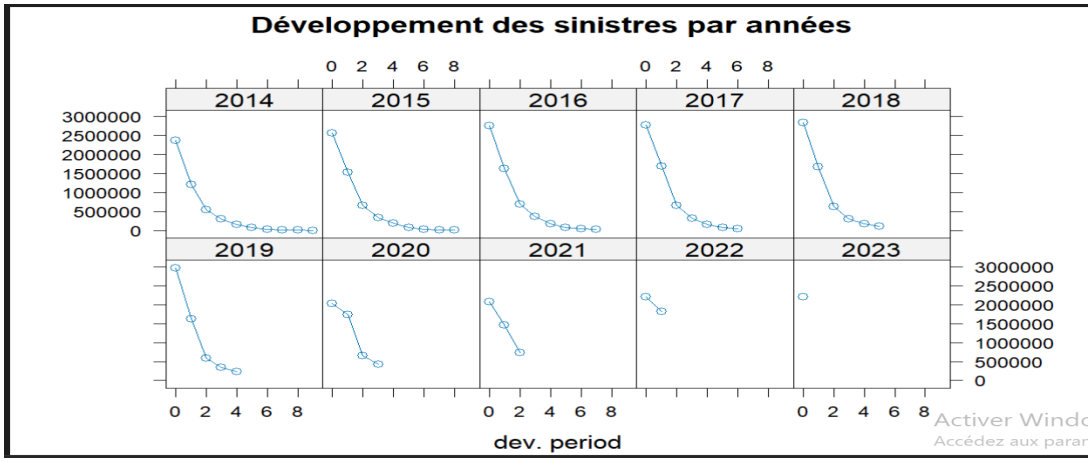
```
    Prov <- as.triangle(as.matrix(Prov))

```

1.2 Exploration

```
plot(Prov/10000, main = "Développement des sinistres selon les années")
plot(Prov, lattice= T, main = "Développement des sinistres par années")
```





Finalement on calcul les pertes cumulatives :

```
    Prov_cum <- incr2cum(Prov)   #Fonction de ChainLadder
```

2 Modèles

2.1 ChainLadder

On calcul tout d'abord les facteurs d'âge :

```
    n <- 6
f <- sapply(1:(n-1),
```

```
            function(i){
              sum(Prov_cum[c(1:(n-i)),i+1])/sum(Prov_cum[c(1:(n-i)),i])
            }
)
f
```

```
[1] 1.581810 1.154298 1.072897 1.037211 1.017353
```

Cependant, ce n'est pas une bonne idée d'assumer que la première année est totalement développer, pour remédier a ce problème on va faire une régression linéaire .

```
    dev.period <- 1:(n-1)
plot(log(f-1) ~ dev.period,
     main="Log-linear extrapolation of age-to-age factors")
tail.model <- lm(log(f-1) ~ dev.period)
abline(tail.model)
```

Maintenant on extrapole les coefficients surtout le dernier F

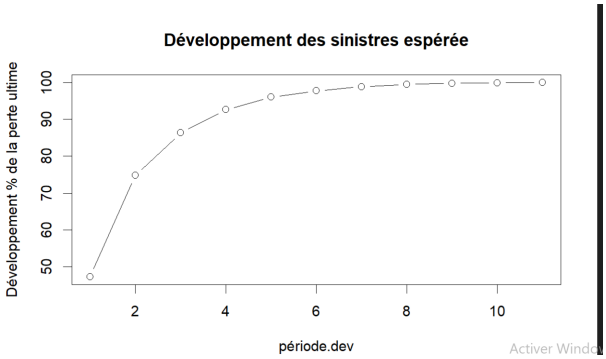
```
co <- coef(tail.model)
tail <- exp(co[1] + c(n:(n + 100)) * co[2]) + 1
f.tail <- prod(tail)
```

On a ainsi :

$$F.tail = 1.01175$$

On applique le modèle et on trouve :

```
    plot(100*(rev(1/cumprod(rev(c(f, tail[tail>1.0001]))))), t="b",
     main="Développement des sinistres espérée",
     xlab="période.dev", ylab="Développement \% de la perte ultime")
```




```
    f <- c(f, f.tail)
full_prov <- cbind(Prov_cum, Ult = rep(0, 6))
for(k in 1:n){
  full_prov[(n-k+1):n, k+1] <- full_prov[(n-k+1):n,k]*f[k]
}
round(full_prov)
```

1	2	3	4	5	6	7	8	9	ult	
2014	2376384	3587552	4133435	4447225	4614376	4694448	4749607	4749713	4760377	4764633	0
2015	2576278	4113428	4775873	5118567	5307366	5399463	5462906	5436654	5450067	NA	0
2016	2763277	4403508	5092467	5456666	5659712	5757924	5825579	5786691	NA	NA	0
2017	2779698	4478229	5139630	5514292	5719483	5818731	5887101	NA	NA	NA	0
2018	2843224	4516828	5213768	5593834	5801985	5902665	5972020	NA	NA	NA	0
2019	2962385	4685931	5408962	5803258	6019201	6123651	6195603	NA	NA	NA	0
2020	2033371	3763913	4412813	4845061	NA	NA	NA	NA	NA	NA	0
2021	2072061	3530602	4266700	NA	NA	NA	NA	NA	NA	NA	0
2022	2210754	4028255	NA	NA	NA	NA	NA	NA	NA	NA	0
2023	2206886	NA	NA	NA	NA	NA	NA	NA	NA	NA	0

2.2 Mack

```
    mack <- MackChainLadder(Prov_cum, est.sigma="Mack")
mack
```

On obtient alors :

Description: df [10 × 6]

	Latest <S3: AsIs>	Dev.To.D... <S3: AsIs>	Ultimate <S3: AsIs>	IBNR <S3: AsIs>	Mack.S.E <S3: AsIs>	CV(IBNR) <S3: AsIs>
2014	4,764,633	1.000	4,764,633	0	0	NaN
2015	5,450,067	0.999	5,454,940	4,873	239	0.0490
2016	5,786,691	0.997	5,805,555	18,864	1,095	0.0580
2017	5,762,673	0.993	5,803,113	40,440	5,140	0.1271
2018	5,723,840	0.985	5,811,489	87,649	9,514	0.1086
2019	5,760,424	0.969	5,941,648	181,224	16,450	0.0908
2020	4,845,061	0.936	5,174,026	328,965	31,322	0.0952
2021	4,266,700	0.874	4,882,119	615,419	71,432	0.1161
2022	4,028,255	0.756	5,330,516	1,302,261	138,183	0.1061
2023	2,206,886	0.462	4,778,035	2,571,149	392,504	0.1527

On obtient alors ainsi la table finale

Description: df [6 × 1]

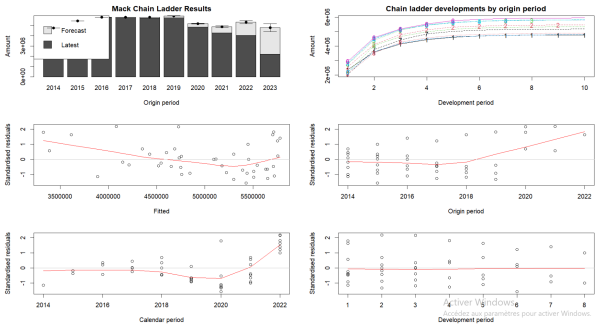
	Totals <chr>
Latest:	48,595,230.00
Dev:	0.90
Ultimate:	53,746,072.86
IBNR:	5,150,842.86
Mack.S.E	432,923.14
CV(IBNR):	0.08

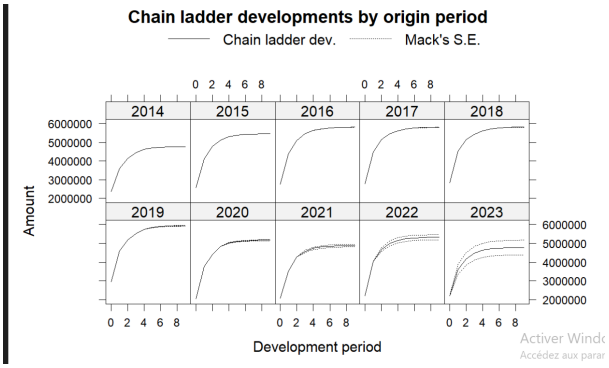
Pour visualizer le tableau complet :

```
mack\$FullTriangle
```

	dev									
origin	0	1	2	3	4	5	6	7	8	9
2014	2376384	3587552	4133435	4447225	4614376	4694448	4733683	4749713	4760377	4764633
2015	2576278	4113428	4775873	5118567	5307366	5384413	5419455	5436654	5450067	5454940
2016	2763277	4403508	5092467	5456666	5633774	5711943	5760314	5786691	5800369	5805555
2017	2779698	4478229	5139630	5461064	5623642	5708223	5762673	5784258	5797930	5803113
2018	2843224	4516828	5141229	5440702	5617544	5723840	5770990	5792606	5806298	5811489
2019	2962385	4582683	5174615	5522049	5760424	5852036	5900242	5922342	5936340	5941648
2020	2033371	3763913	4412813	4845061	5016215	5095991	5137969	5157214	5169404	5174026
2021	2072061	3530602	4266700	4571714	4733212	4808487	4848097	4866256	4877758	4882119
2022	2210754	4028255	4658574	4991602	5167932	5250121	5293369	5313196	5325755	5330516
2023	2206886	3610746	4175736	4474247	4632302	4705972	4744738	4762510	4773767	4778035

```
plot(mack)
plot(mack, lattice = T)
```





2.3 GLM

3.i Gamma

```
    ligne <- rep(1:n,n)
colonne <- rep(1:n, each = n)
X <- as.vector(Prov_cum)
lig <- as.factor(ligne)
col <- as.factor(colonne)
prov_glm <- as.data.frame(cbind(X, lig, col))
fit1 <- glm(X~lig+col , data = prov_glm, family = Gamma(link = "log"))
summary(fit1)
```

On trouve ainsi

```
Call:
glm(formula = X ~ lig + col, family = Gamma(link = "log"), data = prov_glm

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 15.061041   0.097780 154.030  < 2e-16 ***
lig          0.008425   0.019563   0.431 0.668493
col          0.069679   0.019779   3.523 0.000899 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for Gamma family taken to be 0.06216729)

    Null deviance: 4.3364  on 54  degrees of freedom
Residual deviance: 3.5663  on 52  degrees of freedom
  (45 observations effacées parce que manquantes)
AIC: 1695.5

Number of Fisher Scoring iterations: 4
```