Étude de méthodes de provisionnement

Sittah Traoré

1. Introduction

Contexte métier

Le calcul des provisions techniques est un enjeu majeur en assurance, permettant d'estimer les montants futurs à payer pour des sinistres déjà survenus.
Ces estimations sont obligatoires pour les rapports réglementaires et jouent un rôle clé dans la gestion des risques et la tarification.

Objectif

L'objectif de ce notebook est de **présenter plusieurs méthodes de projection** (Chain Ladder, Mack, Bornhuetter Ferguson, Bootstrap), **implémentées manuellement** et **via les packages R standards**, afin :

- de comprendre le fonctionnement interne des modèles ;
- de comparer leurs résultats ;
- de valider les hypothèses sous jacentes.

> **Note :** en pratique, les compagnies utilisent des logiciels spécialisés (ResQ, Aria, PM Expert…), mais l'implémentation manuelle est essentielle pour la validation, l'audit et la formation.

2. Chargement des packages et des données

Les données proviennent d'un jeu d'exemple utilisé en actuariat pour illustrer l'analyse des sinistres d'assurance automobile. Elles représentent un portefeuille anonymisé d'assurance automobile et sont fournies sous forme d'objets R dans l'ensemble freclaimset2motor. L'objet freclaimset2motor est une liste composée de deux éléments :

- **`freclaimset2motor$claimset`** : Ce tableau contient des informations **au niveau des sinistres individuels**. Ce fichier permet d'analyser le **développement des sinistres** dans le temps (suivi par année de survenance et de gestion).

- **`freclaimset2motor$claimsummary`** : Ce tableau contient une **synthèse annuelle par portefeuille**. Cette vue agrégée permet d'étudier la **fréquence** et le **ratio sinistres/primes** au niveau du portefeuille.

```r
# -------------------------------
# Chargement des packages
# -------------------------------
library(CASdatasets)
library(ChainLadder)
library(dplyr)
library(ggplot2)
library(knitr)
library(kableExtra)
library(latex2exp)
library(tinytex)

# -------------------------------
# Fonction pour le formatage des valeurs numériques
# -------------------------------

format_p = function(nombre) {
  format(nombre, big.mark = " ", decimal.mark = ",", nsmall = 2)}

# -------------------------------
# Chargement du jeu de données "freclaimset2motor
# -------------------------------
data(freclaimset2motor)
donnees = freclaimset2motor$claimset
data_freq=freclaimset2motor$aggdata

# -------------------------------
# Affichage du tableau claimset
# -------------------------------

claimset_vars = data.frame(
  Variable = c("ClaimID", "OccurYear", "ManagYear", "ClaimStatus",
               "PaidAmount", "RecourseAmount", "ExpectCharge", "ExpectRecourse"),
  Description = c("Identifiant du sinistre",
                  "Année de survenance du sinistre",
                  "Année de gestion (année d'observation)",
                  "Statut du sinistre (ouvert, clôturé, etc.)",
                  "Montant cumulé payé",
                  "Montant cumulé des recours récupérés",
                  "Provision attendue (montant attendu pour le sinistre)",
                  "Recours attendu"),
  `Unité / Type` = c("Caractère", "Année", "Année", "Caractère",
                     "Euro", "Euro", "Euro", "Euro")
)

kable(claimset_vars,format="latex", caption = "Variables de Claimset",
      align = "l") %>%
  kable_styling(latex_options = c("scale_down", "hold_position","striped"))
```

Table 1: Variables de Claimset

Variable	Description	Unité...Type
ClaimID	Identifiant du sinistre	Caractère
OccurYear	Année de survenance du sinistre	Année
ManagYear	Année de gestion (année d'observation)	Année
ClaimStatus	Statut du sinistre (ouvert, clôturé, etc.)	Caractère
PaidAmount	Montant cumulé payé	Euro
RecourseAmount	Montant cumulé des recours récupérés	Euro
ExpectCharge	Provision attendue (montant attendu pour le sinistre)	Euro
ExpectRecourse	Recours attendu	Euro

```
# -------------------------------
# Affichage du tableau claimsummary
# -------------------------------
claimsummary_vars = data.frame(
  Variable = c("Year", "Exposure", "GWP", "ClaimNb"),
  Description = c("Année de gestion",
                  "Exposition (somme des années-assurés du portefeuille)",
                  "Primes émises brutes (Gross Written Premium)",
                  "Nombre de sinistres enregistrés"),
  `Unité / Type` = c("Année", "Nombre d'années-assurés", "Euro", "Entier")
)

kable(claimsummary_vars, format="latex" ,caption = "Variables de Claimsummary",
      align = "l") %>%
  kable_styling(latex_options = c("scale_down", "hold_position","striped"))
```

Table 2: Variables de Claimsummary

Variable	Description	Unité...Type
Year	Année de gestion	Année
Exposure	Exposition (somme des années-assurés du portefeuille)	Nombre d'années-assurés
GWP	Primes émises brutes (Gross Written Premium)	Euro
ClaimNb	Nombre de sinistres enregistrés	Entier

```
# -------------------------------
# Statistique descriptive du jeu de données
# -------------------------------

summary(donnees)
```

```
##         ClaimID          OccurYear      ManagYear     ClaimStatus
##  1996-008979:     19   Min.   :1995   Min.   :1995   Length:1012839
##  1995-013304:     18   1st Qu.:2000   1st Qu.:2001   Class :character
##  1996-001656:     18   Median :2005   Median :2006   Mode  :character
##  1996-017705:     17   Mean   :2005   Mean   :2005
```

```
##  1998-012412:     17   3rd Qu.:2010   3rd Qu.:2010
##  1998-019564:     17   Max.   :2014   Max.   :2014
##  (Other)    :1012733
##    PaidAmount      RecourseAmount      ExpectCharge     ExpectRecourse
##  Min.   :      0   Min.   :     0.0   Min.   :      0   Min.   :     0.0
##  1st Qu.:      0   1st Qu.:     0.0   1st Qu.:    319   1st Qu.:     0.0
##  Median :    782   Median :     0.0   Median :   1260   Median :     0.0
##  Mean   :   1700   Mean   :   530.6   Mean   :   1797   Mean   :   592.9
##  3rd Qu.:   2069   3rd Qu.:   373.0   3rd Qu.:   1898   3rd Qu.:   568.0
##  Max.   :1013439   Max.   :170418.0   Max.   :1034175   Max.   :170418.0
##
```

3. Préparation des données

La préparation des données vise à nettoyer les doublons, calculer les paiements annuels à partir des montants cumulés et construire un triangle de paiements utilisable pour l'analyse.

- **Nettoyage des doublons et incohérences** Pour chaque sinistre (`ClaimID`) et chaque combinaison année de survenance / année de gestion (`OccurYear`, `ManagYear`), on conserve le **paiement maximal** afin de supprimer les doublons ou erreurs. Cela garantit que seuls les paiements réels sont conservés.
- **Calcul des paiements annuels** Les montants cumulés (`PaidAmount`) sont transformés en **incréments annuels**. Pour chaque sinistre, l'incrément est calculé comme la différence entre le paiement de l'année courante et celui de l'année précédente. Cela fournit le **paiement net pour chaque année** et permet de filtrer les valeurs négatives éventuelles.
- **Calcul de l'année de développement** L'année de développement (`Annee_Dev`) est définie comme la différence entre l'année de gestion et l'année de survenance du sinistre. Elle représente le nombre d'années écoulées depuis la survenance.
- **Aggrégation des paiements** Les données sont ensuite agrégées par année de survenance et année de développement pour obtenir le **triangle de paiements annuels**, où chaque cellule représente la somme des paiements nets pour un sinistre donné et une année de développement spécifique.

Cette préparation assure que le triangle final est cohérent, sans doublons, et reflète fidèlement les paiements annuels pour chaque sinistre sur toute la période d'analyse.

```r
# -------------------------------
# Exemple illustratif : création d'un petit jeu de données factice
# -------------------------------

donnees_demo = data.frame(
  ClaimID    = c("1995-001", "1995-001", "1996-002", "1996-002"),
  OccurYear  = c(1995, 1995, 1996, 1996),
  ManagYear  = c(1995, 1996, 1996, 1997),
  PaidAmount = c(10000, 15000, 8000, 10000)
) %>%
  group_by(ClaimID) %>%
```

```r
  mutate(
    Incremental = PaidAmount - lag(PaidAmount, default = 0),
    Annee_Dev  = ManagYear - OccurYear
  ) %>%
  ungroup()

kable(donnees_demo, caption = "Exemple de préparation des données
      (après nettoyage et ajout de l'anneé de développement)") %>%
  kable_styling(latex_options = c("hold_position", "striped","scale_down"),
                full_width = FALSE)
```

Table 3: Exemple de préparation des données (après nettoyage et ajout de l'anneé de développement)

ClaimID	OccurYear	ManagYear	PaidAmount	Incremental	Annee_Dev
1995-001	1995	1995	10000	10000	0
1995-001	1995	1996	15000	5000	1
1996-002	1996	1996	8000	8000	0
1996-002	1996	1997	10000	2000	1

```r
# -------------------------------
# Préparation et traitement des données de sinistres
# -------------------------------
#Agréger les paiements par sinistre et par année
donnees_corrected = donnees %>%
  group_by(ClaimID,OccurYear, ManagYear) %>%
  summarise(
    PaidAmount = max(PaidAmount, na.rm = TRUE),
    RecourseAmount = max(RecourseAmount, na.rm = TRUE),
    ExpectCharge = max(ExpectCharge, na.rm = TRUE),
    ExpectRecourse = max(ExpectRecourse, na.rm = TRUE),
    .groups = "drop")

#  Calculer les incrémentaux par sinistre
donnees_increm = donnees_corrected  %>%
  arrange(ClaimID, ManagYear) %>%
  group_by(ClaimID) %>%
  mutate(PaidLag = lag(PaidAmount, default = 0),         # valeur précédente
         Incremental = PaidAmount - PaidLag) %>%        # incrément pour cette année
  ungroup()

# Filtrer les incréments positifs et déterminer les années de développement Annee_Dev
donnees_filtrees = donnees_increm %>%
  mutate(Annee_Dev = ManagYear - OccurYear)

#  Agréger pour construire le triangle
triangle_donnees = donnees_filtrees %>%
```

```r
  group_by(OccurYear, Annee_Dev) %>%
  summarise(Paiements = sum(Incremental), .groups = "drop")
```

4. Construction du triangle

On construit la **matrice triangulaire** des paiements **incrémentaux**, avec les **années d'occurrence** en lignes et les **années de développement** en colonnes, puis on la transforme en objet `triangle`. Pour une approche prudente, les **incréments négatifs sont remplacés par zéro**, afin de corriger d'éventuelles surévaluations ou surréservations.

```r
# -------------------------------
# Construction du triangle de paiements annuels
# -------------------------------
# Dimensions du triangle
annee_min = min(triangle_donnees$OccurYear)
annee_max = max(triangle_donnees$OccurYear)
n_origine = annee_max - annee_min + 1
n_dev = max(triangle_donnees$Annee_Dev) + 1


# Initialisation de la matrice (paiements incrémentaux)
matrice_triangle = matrix(NA, nrow = n_origine, ncol = n_dev)
rownames(matrice_triangle) = annee_min:annee_max
colnames(matrice_triangle) = 0:(n_dev - 1)


# Remplissage de la matrice avec les paiements
for (i in 1:nrow(triangle_donnees)) {
  ligne = triangle_donnees[i, ]
  matrice_triangle[as.character(ligne$OccurYear),
                   as.character(ligne$Annee_Dev)] = ligne$Paiements
}

# -------------------------------
# Correction des surréservations
# -------------------------------
matrice_triangle[matrice_triangle < 0] = 0

# -------------------------------
# Transformation en objet "triangle"
# -------------------------------


triangle = as.triangle(matrice_triangle)

# -------------------------------
# Affichage du triangle
# -------------------------------
```

```r
# Créer un tableau vide pour les valeurs colorées
triangle_colore = triangle

for(i in 1:nrow(triangle_colore)) {
  for(j in 1:ncol(triangle_colore)) {
      # Données observées en noir
      triangle_colore[i, j] = cell_spec(format_p(triangle[i, j]),
                                        color = "black",format="latex")
    }
  }

kable(triangle_colore,format="latex", escape = FALSE, align = "r",
      caption = "Triangle des paiements") %>%
  kable_styling(latex_options = c("scale_down", "hold_position"))
```

Table 4: Triangle des paiements

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1995	34 791 250,00	8 642 918,00	0,00	3 657,00	0,00	1 409,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
1996	37 300 461,00	8 133 638,00	0,00	19 303,00	336,00	0,00	16 869,00	825,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	NA
1997	40 601 459,00	8 342 640,00	19 067,00	613,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	NA	NA
1998	44 615 586,00	9 220 080,00	28 634,00	0,00	4 811,00	294,00	15 663,00	97,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	NA	NA	NA
1999	47 964 158,00	11 714 559,00	51 509,00	13 910,00	1 889,00	812,00	0,00	316,00	1 614,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	NA	NA	NA	NA
2000	51 845 145,00	11 499 141,00	79 476,00	27 018,00	7 976,00	121,00	0,00	0,00	0,00	0,00	0,00	420,00	0,00	0,00	0,00	NA	NA	NA	NA	NA
2001	54 085 729,00	13 002 137,00	70 378,00	15 842,00	0,00	1 362,00	0,00	0,00	438,00	0,00	0,00	0,00	0,00	0,00	NA	NA	NA	NA	NA	NA
2002	54 351 659,00	10 211 823,00	84 433,00	2 787,00	629,00	3 800,00	44,00	0,00	0,00	0,00	0,00	0,00	0,00	NA	NA	NA	NA	NA	NA	NA
2003	50 996 325,00	9 553 672,00	85 755,00	5 400,00	974,00	17 500,00	0,00	132,00	0,00	0,00	0,00	0,00	NA	NA	NA	NA	NA	NA	NA	NA
2004	52 354 982,00	8 977 668,00	35 987,00	16 036,00	5 757,00	17 994,00	1 548,00	0,00	482,00	0,00	0,00	NA	NA	NA	NA	NA	NA	NA	NA	NA
2005	53 767 744,00	10 569 704,00	109 810,00	2 052,00	1 329,00	7 479,00	0,00	0,00	1 005,00	0,00	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2006	54 464 735,00	10 047 934,00	37 917,00	0,00	4 194,00	314,00	1 110,00	41,00	0,00	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2007	54 617 493,00	10 785 902,00	39 870,00	59 539,00	763,00	742,00	250,00	0,00	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2008	56 916 284,00	10 231 151,00	108 717,00	13 832,00	2 559,00	1 209,00	186,00	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2009	60 187 604,00	13 052 389,00	168 013,00	23 298,00	0,00	11 173,00	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2010	65 809 392,00	13 948 044,00	128 723,00	1 040 240,00	1 255,00	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2011	63 046 810,00	10 862 597,00	176 809,00	31 130,00	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2012	62 869 785,00	10 534 969,00	87 236,00	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2013	62 694 983,00	11 000 783,00	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2014	63 690 571,00	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

5. Implémentation manuelle du Chain Ladder

5.1. Triangle cumulé

On cumule les paiements le long de chaque ligne afin d'obtenir un triangle **cumulé**, base du Chain Ladder.

```r
# --------------------------------
# Triangle de paiements cumulés
# --------------------------------
# Cumul des paiements par ligne
matrice_triangle_cumulee = t(apply(matrice_triangle, 1, cumsum))
triangle_cumule = as.triangle(matrice_triangle_cumulee)

# --------------------------------
# Affichage du triangle
# --------------------------------
# Créer un tableau vide pour les valeurs colorées
```

```r
triangle_colore = triangle_cumule

for(i in 1:nrow(triangle_colore)) {
  for(j in 1:ncol(triangle_colore)) {
      # Données observées en noir
      triangle_colore[i, j] = cell_spec(format_p(triangle_cumule[i, j]),
                                        color = "black",format="latex")
    }
  }

kable(triangle_colore,format="latex", escape = FALSE, align = "r",
      caption = "Triangle des paiements cumulés") %>%
  kable_styling(latex_options = c("scale_down", "hold_position"))
```

Table 5: Triangle des paiements cumulés

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1995	34 791 250,00	43 434 168,00	43 434 168,00	43 437 825,00	43 437 825,00	43 439 234,00	43 439 234,00	43 439 234,00	43 439 234,00	43 439 234,00	43 439 234,00	43 439 234,00	43 439 234,00	43 439 234,00	43 439 234,00	43 439 234,00	43 439 234,00	43 439 234,00	43 439 234,00	43 439 234,00
1996	37 300 461,00	45 434 099,00	45 434 099,00	45 453 402,00	45 453 738,00	45 453 738,00	45 470 607,00	45 471 432,00	45 471 432,00	45 471 432,00	45 471 432,00	45 471 432,00	45 471 432,00	45 471 432,00	45 471 432,00	45 471 432,00	45 471 432,00	45 471 432,00	45 471 432,00	NA
1997	40 601 459,00	48 944 099,00	48 963 166,00	48 963 779,00	48 963 779,00	48 963 779,00	48 963 779,00	48 963 779,00	48 963 779,00	48 963 779,00	48 963 779,00	48 963 779,00	48 963 779,00	48 963 779,00	48 963 779,00	48 963 779,00	48 963 779,00	48 963 779,00	NA	NA
1998	44 615 586,00	53 835 666,00	53 864 300,00	53 864 300,00	53 869 111,00	53 869 405,00	53 885 068,00	53 885 165,00	53 885 165,00	53 885 165,00	53 885 165,00	53 885 165,00	53 885 165,00	53 885 165,00	53 885 165,00	53 885 165,00	53 885 165,00	NA	NA	NA
1999	47 964 158,00	59 678 717,00	59 730 226,00	59 744 136,00	59 746 025,00	59 746 837,00	59 746 837,00	59 747 153,00	59 748 767,00	59 748 767,00	59 748 767,00	59 748 767,00	59 748 767,00	59 748 767,00	59 748 767,00	59 748 767,00	NA	NA	NA	NA
2000	51 845 145,00	63 344 286,00	63 423 762,00	63 450 780,00	63 458 756,00	63 458 877,00	63 458 877,00	63 458 877,00	63 458 877,00	63 458 877,00	63 458 877,00	63 459 297,00	63 459 297,00	63 459 297,00	63 459 297,00	NA	NA	NA	NA	NA
2001	54 085 729,00	67 087 866,00	67 158 244,00	67 174 086,00	67 174 086,00	67 175 448,00	67 175 448,00	67 175 448,00	67 175 886,00	67 175 886,00	67 175 886,00	67 175 886,00	67 175 886,00	67 175 886,00	NA	NA	NA	NA	NA	NA
2002	54 351 659,00	64 563 482,00	64 647 915,00	64 650 702,00	64 651 331,00	64 655 131,00	64 655 175,00	64 655 175,00	64 655 175,00	64 655 175,00	64 655 175,00	64 655 175,00	64 655 175,00	NA	NA	NA	NA	NA	NA	NA
2003	50 996 325,00	60 549 997,00	60 635 752,00	60 641 152,00	60 642 126,00	60 659 626,00	60 659 626,00	60 659 758,00	60 659 758,00	60 659 758,00	60 659 758,00	60 659 758,00	NA	NA	NA	NA	NA	NA	NA	NA
2004	52 354 982,00	61 332 650,00	61 368 637,00	61 384 673,00	61 390 430,00	61 408 424,00	61 409 972,00	61 409 972,00	61 410 454,00	61 410 454,00	61 410 454,00	NA	NA	NA	NA	NA	NA	NA	NA	NA
2005	53 767 744,00	64 337 448,00	64 447 258,00	64 449 310,00	64 450 639,00	64 458 118,00	64 458 118,00	64 458 118,00	64 459 123,00	64 459 123,00	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2006	54 464 735,00	64 512 669,00	64 550 586,00	64 550 586,00	64 554 780,00	64 555 094,00	64 556 204,00	64 556 245,00	64 556 245,00	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2007	54 617 493,00	65 403 395,00	65 443 265,00	65 502 804,00	65 503 567,00	65 504 309,00	65 504 559,00	65 504 559,00	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2008	56 916 284,00	67 147 435,00	67 256 152,00	67 269 984,00	67 272 543,00	67 273 752,00	67 273 938,00	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2009	60 187 604,00	73 239 993,00	73 408 006,00	73 431 304,00	73 431 304,00	73 442 477,00	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2010	65 809 392,00	79 757 436,00	79 886 159,00	80 926 399,00	80 927 654,00	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2011	63 046 810,00	73 909 407,00	74 086 216,00	74 117 346,00	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2012	62 869 785,00	73 404 754,00	73 491 990,00	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2013	62 694 983,00	73 695 766,00	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2014	63 690 571,00	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

5.2. Facteurs de développement (lambda)

Chaque facteur de développement `j → j+1` est le **ratio moyen** des montants cumulés entre ces deux colonnes, calculé sur les lignes observées.

```r
# -------------------------------
# Détermination des facteurs de développement
# -------------------------------
n = nrow(triangle_cumule)
k = ncol(triangle_cumule)
facteurs_dev = numeric(k-1)

for (j in 1:(k - 1)) {
  numerateur = sum(triangle_cumule[1:(n - j), j + 1], na.rm = TRUE)
  denominateur = sum(triangle_cumule[1:(n - j), j], na.rm = TRUE)
  facteurs_dev[j] = numerateur / denominateur
}

# -------------------------------
# Affichage du tableau des facteurs de développement
# -------------------------------

titres = paste(0:(k-2), "->", 1:(k-1), sep = "")
tableau = as.data.frame(t(format_p(round(facteurs_dev,2))))
```

```r
colnames(tableau) = titres

kable(tableau, caption = "Facteurs de développement",format = "latex",
      ecape=FALSE,align="c")%>%
  kable_styling(latex_options = c("scale_down", "hold_position"))
```

Table 6: Facteurs de développement

0->1	1->2	2->3	3->4	4->5	5->6	6->7	7->8	8->9	9->10	10->11	11->12	12->13	13->14	14->15	15->16	16->17	17->18	18->19
1,20	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00

5.3. Complétion du triangle cumulé

Nous allons **projeter les valeurs manquantes** dans le triangle cumulé en utilisant les **facteurs de développement** calculés précédemment.
Chaque cellule manquante est estimée en multipliant la valeur de la période précédente par le facteur correspondant.

Nous affichons ici le triangle cumulé complet en mettant en évidence :
* Les **données observées** (présentes dans le triangle initial)
* Les **données projetées** (calculées par multiplication des facteurs de développement)

Table 7: Triangle cumulé :Paiements Observés (noir) vs Projetés (bleu)

	0	1	2	3	4	5	6	7	8	9
1995	34 791 250,00	43 434 168,00	43 434 168,00	43 437 825,00	43 437 825,00	43 439 234,00	43 439 234,00	43 439 234,00	43 439 234,00	43 439 234,00
1996	37 300 461,00	45 434 099,00	45 434 099,00	45 453 402,00	45 453 738,00	45 453 738,00	45 470 607,00	45 471 432,00	45 471 432,00	45 471 432,00
1997	40 601 459,00	48 944 099,00	48 963 166,00	48 963 779,00	48 963 779,00	48 963 779,00	48 963 779,00	48 963 779,00	48 963 779,00	48 963 779,00
1998	44 615 586,00	53 835 666,00	53 864 300,00	53 864 300,00	53 869 111,00	53 869 405,00	53 885 068,00	53 885 165,00	53 885 165,00	53 885 165,00
1999	47 964 158,00	59 678 717,00	59 730 226,00	59 744 136,00	59 746 025,00	59 746 837,00	59 746 837,00	59 747 153,00	59 748 767,00	59 748 767,00
2000	51 845 145,00	63 344 286,00	63 423 762,00	63 450 780,00	63 458 756,00	63 458 877,00	63 458 877,00	63 458 877,00	63 458 877,00	63 458 877,00
2001	54 085 729,00	67 087 866,00	67 158 244,00	67 174 086,00	67 174 086,00	67 175 448,00	67 175 448,00	67 175 448,00	67 175 886,00	67 175 886,00
2002	54 351 659,00	64 563 482,00	64 647 915,00	64 650 702,00	64 651 331,00	64 655 131,00	64 655 175,00	64 655 175,00	64 655 175,00	64 655 175,00
2003	50 996 325,00	60 549 997,00	60 635 752,00	60 641 152,00	60 642 126,00	60 659 626,00	60 659 626,00	60 659 758,00	60 659 758,00	60 659 758,00
2004	52 354 982,00	61 332 650,00	61 368 637,00	61 384 673,00	61 390 430,00	61 408 424,00	61 409 972,00	61 409 972,00	61 410 454,00	61 410 454,00
2005	53 767 744,00	64 337 448,00	64 447 258,00	64 449 310,00	64 450 639,00	64 458 118,00	64 458 118,00	64 458 118,00	64 459 123,00	64 459 123,00
2006	54 464 735,00	64 512 669,00	64 550 586,00	64 550 586,00	64 554 780,00	64 555 094,00	64 556 204,00	64 556 245,00	64 556 245,00	64 556 245,00
2007	54 617 493,00	65 403 395,00	65 443 265,00	65 502 804,00	65 503 567,00	65 504 309,00	65 504 559,00	65 504 559,00	65 504 891,18	65 504 891,18
2008	56 916 284,00	67 147 435,00	67 256 152,00	67 269 984,00	67 272 543,00	67 273 752,00	67 273 938,00	67 274 062,35	67 274 403,50	67 274 403,50
2009	60 187 604,00	73 239 993,00	73 408 006,00	73 431 304,00	73 431 304,00	73 442 477,00	73 445 630,89	73 445 766,65	73 446 139,10	73 446 139,10
2010	65 809 392,00	79 757 436,00	79 886 159,00	80 926 399,00	80 927 654,00	80 933 402,10	80 936 877,68	80 937 027,28	80 937 437,72	80 937 437,72
2011	63 046 810,00	73 909 407,00	74 086 216,00	74 117 346,00	74 119 789,65	74 125 054,20	74 128 237,41	74 128 374,43	74 128 750,34	74 128 750,34
2012	62 869 785,00	73 404 754,00	73 491 990,00	73 580 553,60	73 582 979,55	73 588 205,98	73 591 366,13	73 591 502,15	73 591 875,34	73 591 875,34
2013	62 694 983,00	73 695 766,00	73 781 359,38	73 870 271,69	73 872 707,19	73 877 954,20	73 881 126,79	73 881 263,35	73 881 638,01	73 881 638,01
2014	63 690 571,00	76 408 080,91	76 496 824,48	76 589 009,14	76 591 534,28	76 596 974,40	76 600 263,76	76 600 405,34	76 600 793,79	76 600 793,79

	10	11	12	13	14	15	16	17	18	19
1995	43 439 234,00	43 439 234,00	43 439 234,00	43 439 234,00	43 439 234,00	43 439 234,00	43 439 234,00	43 439 234,00	43 439 234,00	43 439 234,00
1996	45 471 432,00	45 471 432,00	45 471 432,00	45 471 432,00	45 471 432,00	45 471 432,00	45 471 432,00	45 471 432,00	45 471 432,00	45 471 432,00
1997	48 963 779,00	48 963 779,00	48 963 779,00	48 963 779,00	48 963 779,00	48 963 779,00	48 963 779,00	48 963 779,00	48 963 779,00	48 963 779,00
1998	53 885 165,00	53 885 165,00	53 885 165,00	53 885 165,00	53 885 165,00	53 885 165,00	53 885 165,00	53 885 165,00	53 885 165,00	53 885 165,00
1999	59 748 767,00	59 748 767,00	59 748 767,00	59 748 767,00	59 748 767,00	59 748 767,00	59 748 767,00	59 748 767,00	59 748 767,00	59 748 767,00
2000	63 458 877,00	63 459 297,00	63 459 297,00	63 459 297,00	63 459 297,00	63 459 297,00	63 459 297,00	63 459 297,00	63 459 297,00	63 459 297,00
2001	67 175 886,00	67 175 886,00	67 175 886,00	67 175 886,00	67 175 886,00	67 175 886,00	67 175 886,00	67 175 886,00	67 175 886,00	67 175 886,00
2002	64 655 175,00	64 655 175,00	64 655 175,00	64 655 175,00	64 655 175,00	64 655 175,00	64 655 175,00	64 655 175,00	64 655 175,00	64 655 175,00
2003	60 659 758,00	60 659 758,00	60 659 758,00	60 659 758,00	60 659 758,00	60 659 758,00	60 659 758,00	60 659 758,00	60 659 758,00	60 659 758,00
2004	61 410 454,00	61 410 504,83	61 410 504,83	61 410 504,83	61 410 504,83	61 410 504,83	61 410 504,83	61 410 504,83	61 410 504,83	61 410 504,83
2005	64 459 123,00	64 459 176,35	64 459 176,35	64 459 176,35	64 459 176,35	64 459 176,35	64 459 176,35	64 459 176,35	64 459 176,35	64 459 176,35
2006	64 556 245,00	64 556 298,43	64 556 298,43	64 556 298,43	64 556 298,43	64 556 298,43	64 556 298,43	64 556 298,43	64 556 298,43	64 556 298,43
2007	65 504 891,18	65 504 945,39	65 504 945,39	65 504 945,39	65 504 945,39	65 504 945,39	65 504 945,39	65 504 945,39	65 504 945,39	65 504 945,39
2008	67 274 403,50	67 274 459,18	67 274 459,18	67 274 459,18	67 274 459,18	67 274 459,18	67 274 459,18	67 274 459,18	67 274 459,18	67 274 459,18
2009	73 446 139,10	73 446 199,88	73 446 199,88	73 446 199,88	73 446 199,88	73 446 199,88	73 446 199,88	73 446 199,88	73 446 199,88	73 446 199,88
2010	80 937 437,72	80 937 504,71	80 937 504,71	80 937 504,71	80 937 504,71	80 937 504,71	80 937 504,71	80 937 504,71	80 937 504,71	80 937 504,71
2011	74 128 750,34	74 128 811,69	74 128 811,69	74 128 811,69	74 128 811,69	74 128 811,69	74 128 811,69	74 128 811,69	74 128 811,69	74 128 811,69
2012	73 591 875,34	73 591 936,25	73 591 936,25	73 591 936,25	73 591 936,25	73 591 936,25	73 591 936,25	73 591 936,25	73 591 936,25	73 591 936,25
2013	73 881 638,01	73 881 699,16	73 881 699,16	73 881 699,16	73 881 699,16	73 881 699,16	73 881 699,16	73 881 699,16	73 881 699,16	73 881 699,16
2014	76 600 793,79	76 600 857,19	76 600 857,19	76 600 857,19	76 600 857,19	76 600 857,19	76 600 857,19	76 600 857,19	76 600 857,19	76 600 857,19

5.4. Vérification des hypothèses Chain Ladder

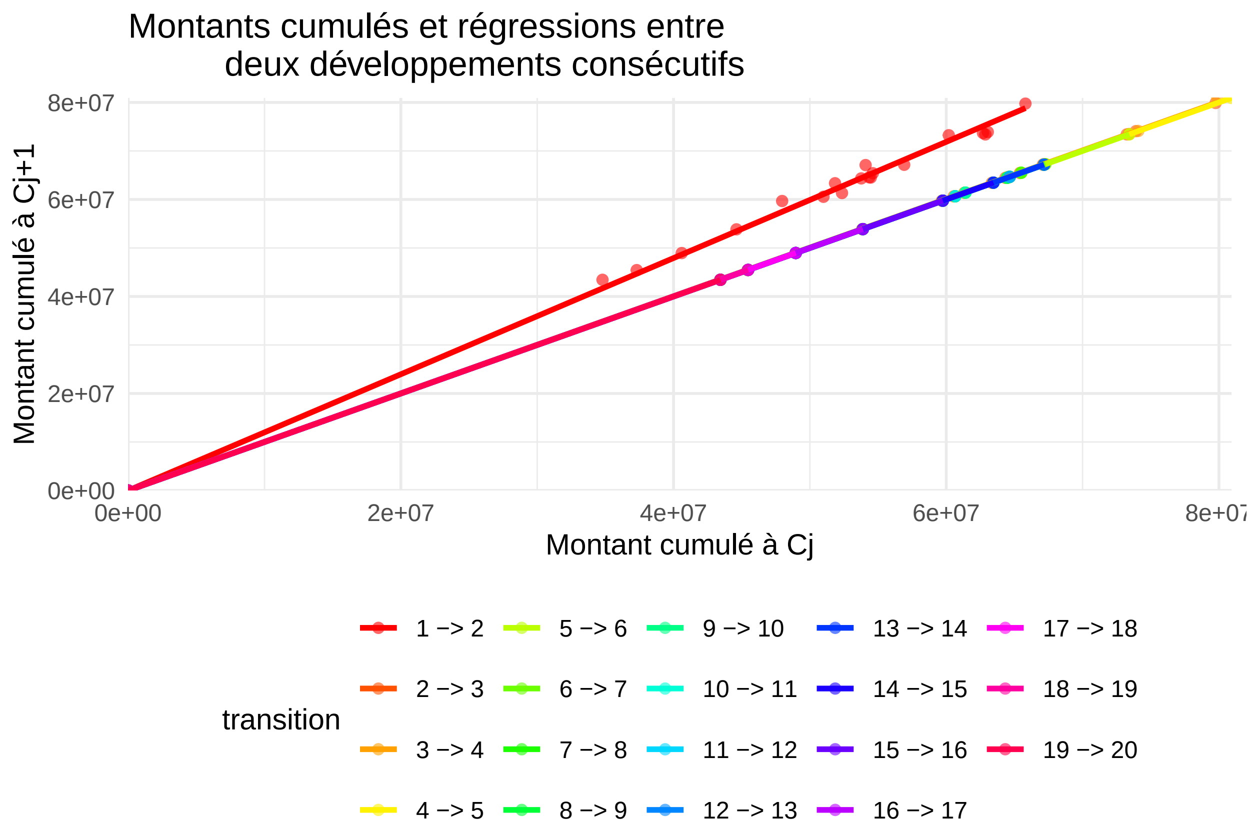
Avant d'interpréter les résultats, il est important de vérifier que les hypothèses de la méthode Chain Ladder sont respectées :

- Indépendance (H1) : les développements doivent être indépendants. On vérifie cela via l'analyse des résidus standardisés et les p-values.
- Existence d'un facteur de développement (H2) reliant la période j à la période j+1 : chaque développement doit être proportionnel au précédent. Cela se vérifie à l'aide d'une régression linéaire. Des graphes permettent de détecter d'éventuels écarts selon les exercices et les périodes.

On ajuste, pour chaque colonne, une régression **linéaire simple** et on inspecte les **résidus** selon : * l'année d'origine, * la période de développement, * et le niveau des montants — pour juger l'indépendance

5.4.1. Vérification de l'hypothèse H2

```r
# -------------------------------
# Régressions
# -------------------------------
regressions = vector("list", length = k - 1)
df_all = data.frame()

for (j in 1:(k - 1)) {
  x = triangle_cumule[1:(n - j), j]
  y = triangle_cumule[1:(n - j), j + 1]
  df = data.frame(x = x, y = y, transition = paste(j, "->", j+1))
  df = df[complete.cases(df), ]

  regressions[[j]] = lm(y ~ x, data = df)

 # Ajouter le point (0,0) pour forcer le passage par l'origine
  df = rbind(data.frame(x = 0, y = 0, transition = paste(j, "->", j+1)), df)

  # Ajouter au data.frame global
  df_all = rbind(df_all, df)
}

df_all$transition=factor(df_all$transition, levels = paste(1:(k-1), "->", 2:k))

# -------------------------------
# Construction du graphe
# -------------------------------
ggplot(df_all, aes(x = x, y = y, color = transition)) +
  geom_point(alpha = 0.6) +
  geom_smooth(method = "lm", formula = y ~ 0 + x, se = FALSE, size = 1) +
  xlab("Montant cumulé à Cj") +
  ylab("Montant cumulé à Cj+1") +
  ggtitle("Montants cumulés et régressions entre
          deux développements consécutifs") +
  scale_x_continuous(expand = c(0, 0), limits = c(0, NA)) +
  scale_y_continuous(expand = c(0, 0), limits = c(0, NA)) +
  theme_minimal() +
  theme(legend.position = "bottom") +
  scale_color_manual(values = rainbow(k-1))
```

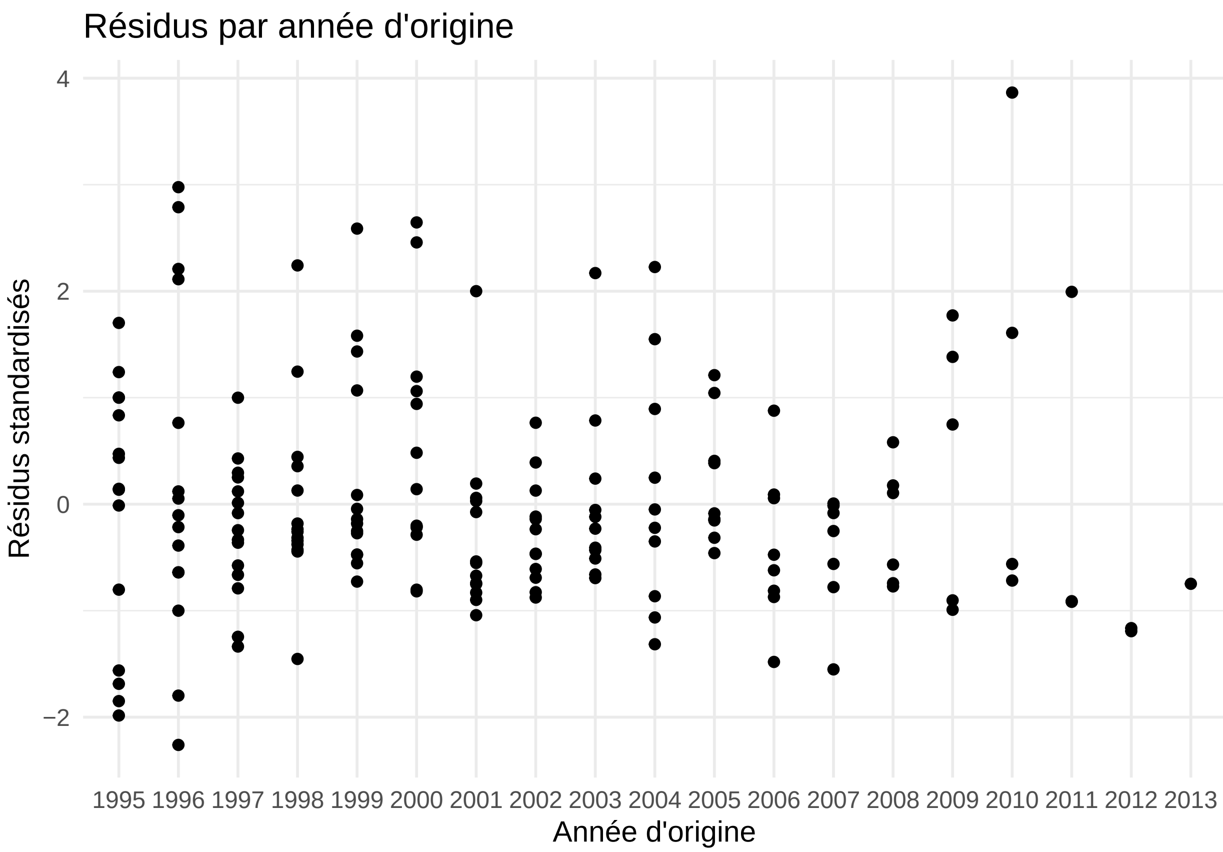


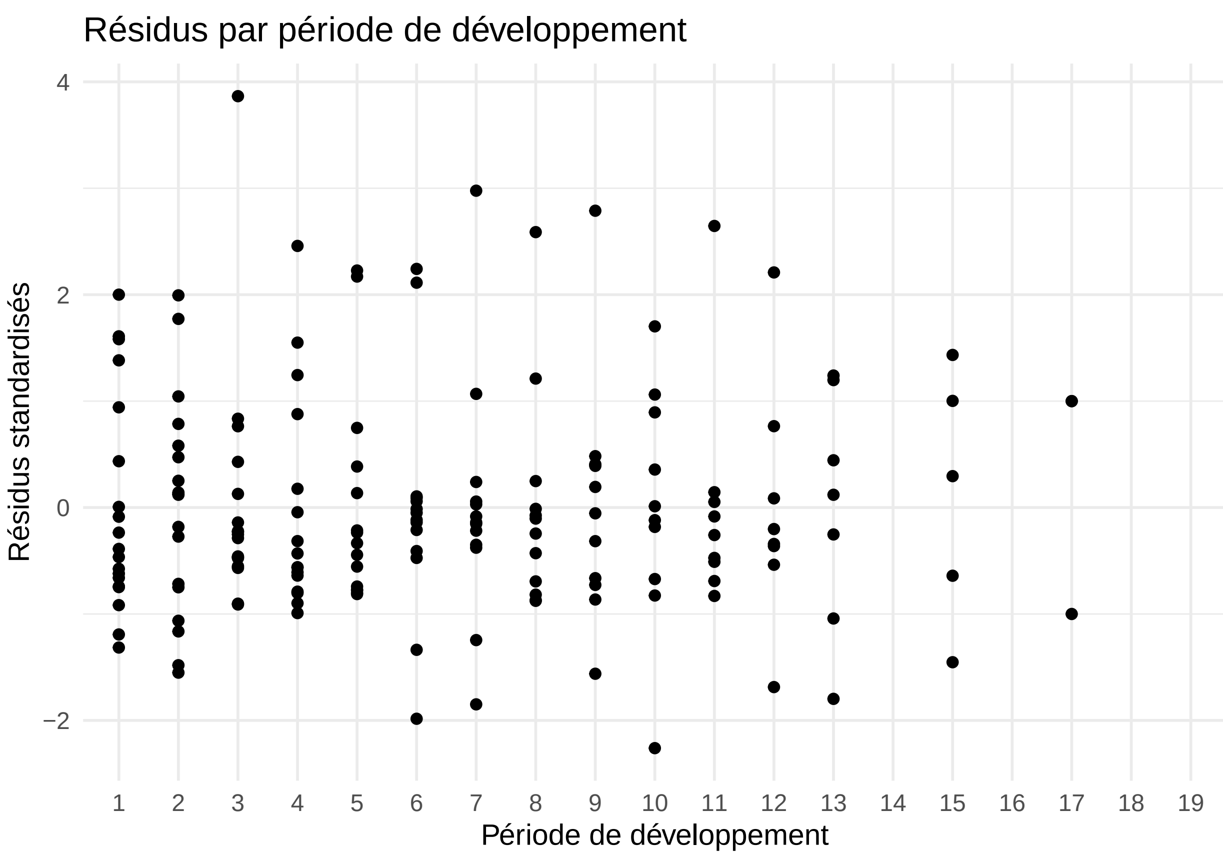
Les points observés suivent globalement une droite et les régressions linéaires offrent un bon ajustement, ce qui confirme une relation proportionnelle entre C_j et C_{j+1}. Ainsi, l'hypothèse **H2 (linéarité des développements successifs)** est validée : les facteurs de développement peuvent être estimés de manière fiable par les pentes des régressions.

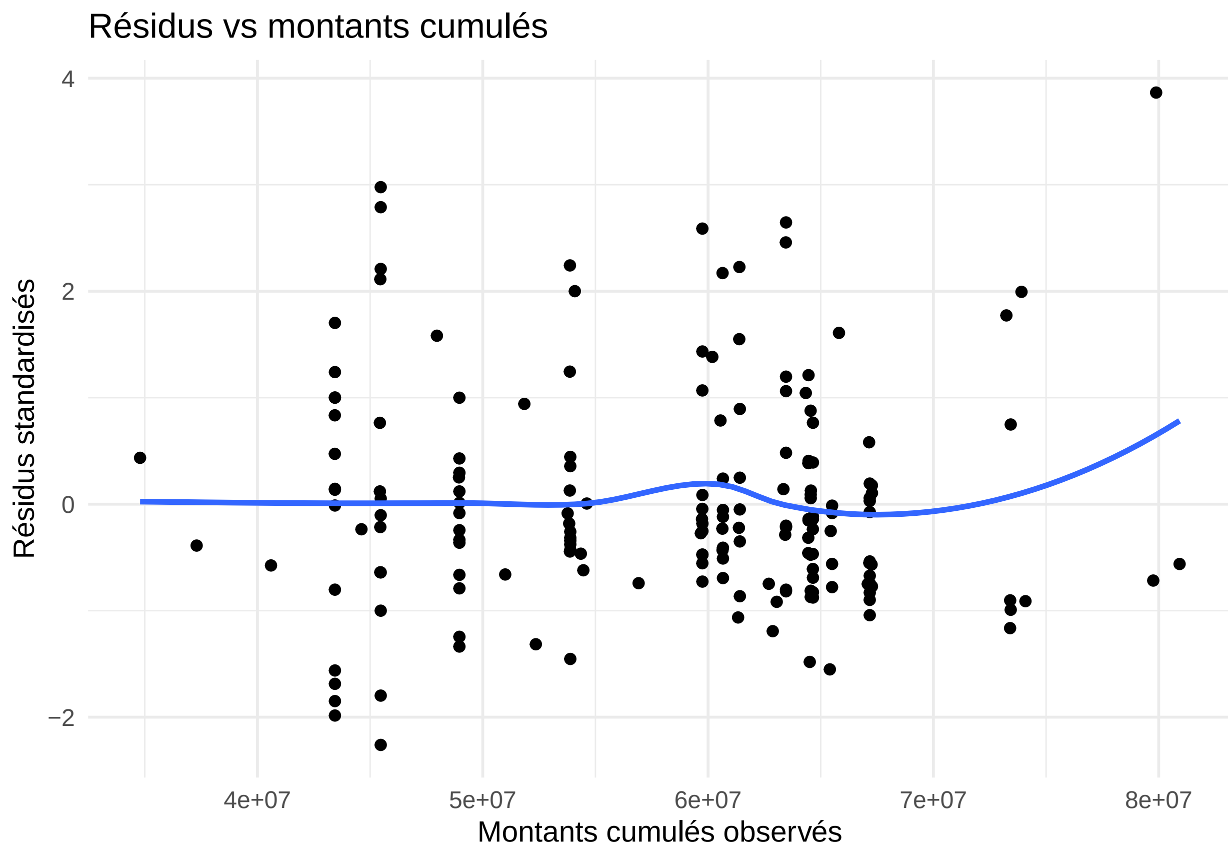
5.4.2. Indépendance (diagnostic via résidus)

```
# -------------------------------
# Diagnostics des résidus
# -------------------------------
# Résidus standardisés regroupés
residus_tous = data.frame()
for (j in 1:(k - 1)) {
  mod = regressions[[j]]
  if (is.null(mod)) next
  res = rstandard(mod)
  x = model.frame(mod)$x
  annees = rownames(triangle_cumule)[1:length(res)]
  df_res = data.frame(
    Residus = res,
    Developpement = j,
    Annee_Origine = annees,
    ValeurX = x
  )
```

```r
  residus_tous = rbind(residus_tous, df_res)
}

# -------------------------------
# Construction des graphes
# -------------------------------
# Graphiques de diagnostic
g1 = ggplot(residus_tous, aes(x = as.factor(Annee_Origine), y = Residus)) +
  geom_point() +
  ggtitle("Résidus par année d'origine") +
  xlab("Année d'origine") + ylab("Résidus standardisés") +
  theme_minimal()

g2 = ggplot(residus_tous, aes(x = as.factor(Developpement), y = Residus)) +
  geom_point()+
  ggtitle("Résidus par période de développement") +
  xlab("Période de développement") + ylab("Résidus standardisés") +
  theme_minimal()

g3 = ggplot(residus_tous, aes(x = ValeurX, y = Residus)) +
  geom_point() +
  geom_smooth(method = "loess", se = FALSE) +
  ggtitle("Résidus vs montants cumulés") +
  xlab("Montants cumulés observés") + ylab("Résidus standardisés") +
  theme_minimal()

print(g1); print(g2); print(g3)
```


Résidus par année d'origine
Résidus standardisés
4
2
0
−2
1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013
Année d'origine


Résidus par période de développement
Résidus standardisés
4
2
0
−2
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
Période de développement



Les trois graphiques montrent des structures non aléatoires dans les résidus :

- une dépendance avec les montants cumulés,
- une variabilité inégale selon les périodes de développement,
- et biais systématiques selon certaines années d'origine.

Ces constats indiquent que les hypothèses de base du Chain Ladder (indépendance, homoscédasticité, absence de tendance) ne sont pas vérifiées.

Conclusion : le modèle Chain Ladder n'est pas totalement adapté à ces données. Il peut être appliqué à titre illustratif ou pour fournir un premier estimateur des réserves, mais ses résultats doivent être interprétés avec prudence et idéalement complétés par des méthodes plus robustes (ex. Mack, bootstrap).

5.5. Réserves (Chain Ladder manuel)

Bien que les hypothèses du Chain Ladder ne soient pas strictement vérifiées, la méthode est néanmoins appliquée ici afin d'illustrer son fonctionnement et d'obtenir une estimation mécanique des ultimes et des réserves associées.

La **réserve** par ligne est la différence entre l'**ultime CL** projeté et le **cumul observé**. La somme donne la réserve totale.

```r
# -------------------------------
# Calcul de la réserve
# -------------------------------
# Charge ultime par ligne = dernière colonne du triangle complété
charge_ultime_cl = triangle_complet[, k]

# Dernier paiement cumulé observé par ligne
dernier_cumule_observe = apply(triangle_cumule, 1,
                               function(x) tail(na.omit(x), 1))


# Réserve par année d'origine + totale
reserve_par_annee_cl = charge_ultime_cl - dernier_cumule_observe
reserve_totale_cl = sum(reserve_par_annee_cl, na.rm = TRUE)


resultats_cl = data.frame(
  Annee_Origine = rownames(triangle_cumule),
  Charge_ultime=format_p(round(charge_ultime_cl,2)),
  Reserve_CL = format_p(round(reserve_par_annee_cl, 2))
)

# -------------------------------
# Affichage des résultats
# -------------------------------

print(resultats_cl)
```

```
##      Annee_Origine Charge_ultime    Reserve_CL
## 1995          1995 43 439 234,00          0,00
## 1996          1996 45 471 432,00          0,00
## 1997          1997 48 963 779,00          0,00
## 1998          1998 53 885 165,00          0,00
## 1999          1999 59 748 767,00          0,00
## 2000          2000 63 459 297,00          0,00
## 2001          2001 67 175 886,00          0,00
## 2002          2002 64 655 175,00          0,00
## 2003          2003 60 659 758,00          0,00
## 2004          2004 61 410 504,83         50,83
## 2005          2005 64 459 176,35         53,35
## 2006          2006 64 556 298,43         53,43
## 2007          2007 65 504 945,39        386,39
## 2008          2008 67 274 459,18        521,18
## 2009          2009 73 446 199,88      3 722,88
## 2010          2010 80 937 504,71      9 850,71
## 2011          2011 74 128 811,69     11 465,69
## 2012          2012 73 591 936,25     99 946,25
## 2013          2013 73 881 699,16    185 933,16
## 2014          2014 76 600 857,19 12 910 286,19
```

```r
cat("\nRéserve totale: ", format_p(reserve_totale_cl), "\n")
```

```
##
## Réserve totale:  13 222 270,06
```

6. Implémentation manuelle de Mack Chain Ladder

La méthode de **Mack** reprend la structure du **Chain Ladder**, tout en intégrant un **modèle de variance** et le calcul du **MSEP** pour quantifier l'incertitude.

6.1. Hypothèse de variance

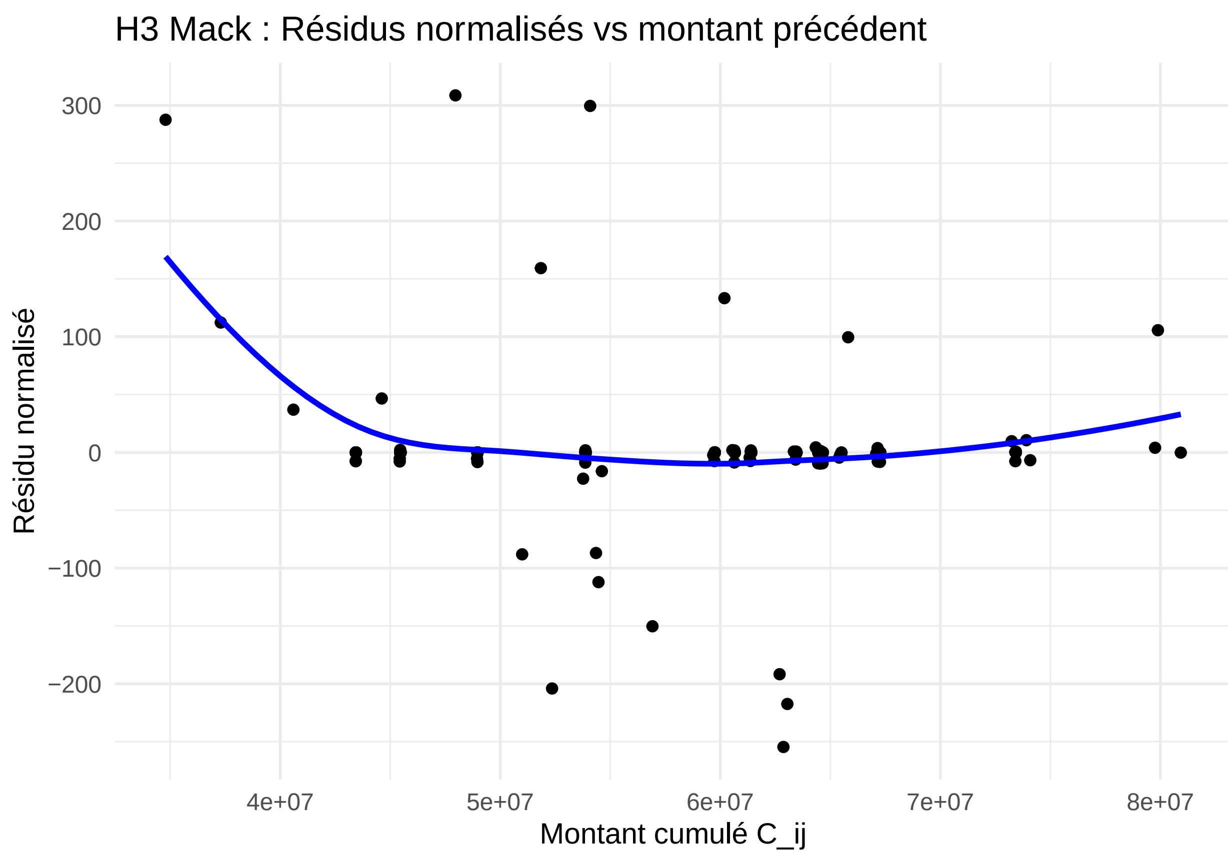
Ce qu'on attend si l'hypothèse est respectée : Les résidus normalisés

$$\varepsilon_{i,j} = \frac{C_{i,j+1} - \hat{\lambda}_j C_{i,j}}{\sqrt{C_{i,j}}}$$

devraient être dispersés aléatoirement autour de zéro, sans structure particulière, et leur variance ne devrait pas dépendre du montant cumulé $C_{i,j}$.

```r
# --------------------------------
# Vérification de la variance
# --------------------------------
variances_list = list()

for (j in 1:(k - 1)) {
  x = triangle_cumule[1:(n - j), j]
  y = triangle_cumule[1:(n - j), j + 1]
  df = data.frame(x = x, y = y)
  df = df[complete.cases(df), ]

  # Facteur de développement estimé (lambda_j)
  lambda_hat = facteurs_dev[j]

  # Résidu normalisé selon Mack
  df$resid_mack = (df$y - lambda_hat * df$x) / sqrt(df$x)
  df$Developpement = j
  variances_list[[j]] = df
}

variances_df = do.call(rbind, variances_list)

# --------------------------------
# Construction du graphe
# --------------------------------
g4 = ggplot(variances_df, aes(x = x, y = resid_mack)) +
  geom_point() +
  geom_smooth(method = "loess", se = FALSE, color = "blue") +
```

```r
  ggtitle("H3 Mack : Résidus normalisés vs montant précédent") +
  xlab("Montant cumulé C_ij") + ylab("Résidu normalisé") +
  theme_minimal()

print(g4)
```



Le graphique des résidus normalisés selon Mack montre une forte dépendance de la variance aux montants cumulés, en particulier pour les plus faibles valeurs où la dispersion est élevée. L'hypothèse **H3** n'est donc pas strictement vérifiée, ce qui suggère que la variance du modèle n'est pas correctement spécifiée.

Bien que les hypothèses de Mack ne soient pas strictement respectées, la méthode est appliquée ici à titre illustratif afin de montrer sa mise en œuvre et d'obtenir une estimation indicative des réserves et des mesures d'incertitude.

6.2. Estimation des variances σ_j^2 (formules de Mack)

Sous les hypothèses du modèle de Mack, la variance conditionnelle s'écrit $\mathrm{Var}(C_{i,j+1} \mid C_{i,j}) = \sigma_j^2\, C_{i,j}$. Un estimateur sans biais de σ_j^2 est :

$$S_j^2 = \frac{1}{n-j-1} \sum_{i=1}^{n-j} C_{i,j} \left(\frac{C_{i,j+1}}{C_{i,j}} - \hat{\lambda}_j \right)^2$$

où $\hat{\lambda}_j$ est le facteur de développement estimé. Cet estimateur mesure la dispersion des développements observés autour de $\hat{\lambda}_j$, pondérée par les montants cumulés.

```r
# -------------------------------
# Calcul de la variance
# -------------------------------
sigma2 = numeric(k - 1)

for (j in 1:(k - 1)) {
  termes = c()
  for (i in 1:(n - j)) {
    if (!is.na(triangle_cumule[i, j]) && !is.na(triangle_cumule[i, j + 1])) {
      obs = triangle_cumule[i, j + 1]
      mu = facteurs_dev[j] * triangle_cumule[i, j]
      termes = c(termes, triangle_cumule[i, j] * ((obs / triangle_cumule[i, j])
                                                  - facteurs_dev[j])^2)
    }
  }
  # Degré de liberté (n - j - 1) > 0 requis
  denom = (n - j - 1)
  sigma2[j] = ifelse(denom > 0, sum(termes, na.rm = TRUE) / denom, NA_real_)
}

# Stabilisation simple pour la dernière colonne si besoin
if (is.na(sigma2[k - 1]) && k - 2 >= 2) {
  sigma2[k - 1] = min((sigma2[k - 2]^2) / sigma2[k - 3], sigma2[k - 2],
                      sigma2[k - 3], na.rm = TRUE)
}
titres = paste(1:(k-1), sep = "")
tableau = tibble(`Période` = titres, `Ecart type` =format_p(
                                             round(sqrt(sigma2),2)))

# -------------------------------
# Affichage des résultats
# -------------------------------
kable(t(tableau), caption = "Ecart type",format = "latex",ecape=FALSE,
      align="c")%>%
  kable_styling(latex_options = c("scale_down", "hold_position"))
```

Table 8: Ecart type

Période	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Ecart type	179,27	5,42	27,52	0,31	0,79	0,85	0,03	0,07	0,00	0,00	0,02	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00

6.3. MSEP (par ligne, puis total)

Sous Mack, la MSEP de l'ultime estimé se décompose en **erreur de modèle** (variabilité du processus stochastique) et en **erreur d'estimation** (incertitude sur les paramètres λ_j, S_j^2). Pour les réserves par année d'origine, une formule analytique combine ces deux sources d'erreur. La MSEP totale n'est pas égale à la somme des MSEP individuelles, car un terme supplémentaire apparaît pour tenir compte de la **corrélation entre années d'origine**, liée à l'utilisation des mêmes paramètres de développement.

```r
# -------------------------------
# Détermination du MSEP
# -------------------------------
# MSEP par année de survenance, formules type Mack
MSEP_individuel = rep(NA_real_, n)

for (i in 2:n) {
  somme_j = 0
  for (j in (n - i + 1):(n - 1)) {
    if (j < 1) next
    terme1 = 1 / triangle_complet[i, j]
    terme2 = 1 / sum(triangle_complet[1:(n - j), j], na.rm = TRUE)
    somme_j = somme_j + (sigma2[j] / (facteurs_dev[j]^2)) * (terme1 + terme2)
  }
  MSEP_individuel[i] = (triangle_complet[i, k]^2) * somme_j
}

# Terme de covariance entre les années de survenance (approximation Mack)
cov_termes = rep(0, n)
for (i in 2:n) {
  somme_k = ifelse(i + 1 <= n, sum(triangle_complet[(i + 1):n, k], na.rm = TRUE)
                   , 0)
  somme_j = 0
  for (j in (n - i + 1):(n - 1)) {
    if (j < 1) next
    denom = sum(triangle_complet[1:(n - j), j], na.rm = TRUE)
    somme_j = somme_j + (2 * sigma2[j]) / (facteurs_dev[j]^2 * denom)
  }
  cov_termes[i] = triangle_complet[i, k] * somme_k * somme_j
}

#MSEP global de la réserve

MSEP_par_ligne = MSEP_individuel + cov_termes
MSEP_total = sum(MSEP_par_ligne, na.rm = TRUE)
```

6.4.Quantiles

On déduit le **quantile à 75 %** et le **quantile à 95 %** des réserves par ligne et du total à partir des **MSEP** calculés, sous l'hypothèse que la réserve suit une loi normale.

```r
# -------------------------------
# Calcul des quantiles
# -------------------------------
z1 = qnorm(0.75)
z2= qnorm(0.95)

q75 = reserve_par_annee_cl - z1 * sqrt(MSEP_individuel)
q95 = reserve_par_annee_cl + z2 * sqrt(MSEP_individuel)
```

```r
resultats_mack = data.frame(
  Annee_Origine = rownames(triangle_cumule),
  Reserve_CL = format_p(round(reserve_par_annee_cl, 2)),
  MSEP = format_p(round(MSEP_individuel, 2)),
  Quantile_75 = format_p(round(q75, 2)),
  Quantile_95 = format_p(round(q95, 2))
)
# -------------------------------
# Affichage des résultats
# -------------------------------
# Totaux
reserve_totale = sum(reserve_par_annee_cl, na.rm = TRUE)
q75_tot = reserve_totale - z1* sqrt(MSEP_total)
q95_tot = reserve_totale + z2 * sqrt(MSEP_total)

print("Réserves, MSEP , Quantiles :")
```

```
## [1] "Réserves, MSEP , Quantiles :"
```

```r
print(resultats_mack)
```

```
##      Annee_Origine    Reserve_CL         MSEP   Quantile_75   Quantile_95
## 1995          1995          0,00           NA            NA            NA
## 1996          1996          0,00 0,000000e+00          0,00          0,00
## 1997          1997          0,00 0,000000e+00          0,00          0,00
## 1998          1998          0,00 0,000000e+00          0,00          0,00
## 1999          1999          0,00 0,000000e+00          0,00          0,00
## 2000          2000          0,00 0,000000e+00          0,00          0,00
## 2001          2001          0,00 0,000000e+00          0,00          0,00
## 2002          2002          0,00 0,000000e+00          0,00          0,00
## 2003          2003          0,00 0,000000e+00          0,00          0,00
## 2004          2004         50,83 2,092918e+04        -46,75        288,79
## 2005          2005         53,35 2,208593e+04        -46,89        297,80
## 2006          2006         53,43 2,212296e+04        -46,89        298,08
## 2007          2007        386,39 3,349079e+05         -3,94      1 338,29
## 2008          2008        521,18 4,333439e+05         77,17      1 603,97
## 2009          2009      3 722,88 5,792399e+07     -1 410,51     16 241,50
## 2010          2010      9 850,71 1,189889e+08      2 493,24     27 793,10
## 2011          2011     11 465,69 1,156920e+08      4 210,86     29 157,77
## 2012          2012     99 946,25 5,966548e+10    -64 808,12    501 726,73
## 2013          2013    185 933,16 6,222765e+10     17 678,51    596 249,64
## 2014          2014 12 910 286,19 2,252589e+12 11 897 969,60 15 378 985,83
```

```r
cat("\n--- Résultats globaux ---\n")
```

```
##
## --- Résultats globaux ---
```

```r
cat("Réserve totale (CL) :", format_p(reserve_totale), "\n")
```

```
## Réserve totale (CL) : 13 222 270,06
```

```r
cat("MSEP total :", format_p(MSEP_total), "\n")
```

```
## MSEP total : 2,399212e+12
```

```r
cat("Quantile à 75% : " ,format_p(q75_tot), "\n")
```

```
## Quantile à 75% :  12 177 526,51
```

```r
cat("Quantile à 95%  : ", format_p(q95_tot), "\n")
```

```
## Quantile à 95%  :  15 770 048,15
```

Les valeurs obtenues sont cruciales pour la gestion du risque en assurance :

- Évaluation du risque : Le MSEP quantifie le risque de sous-estimation ou de surestimation des réserves. Un MSEP élevé indique une plus grande incertitude et un risque plus important.
- Capitalisation : Les régulateurs exigent que les compagnies détiennent suffisamment de capital pour couvrir les incertitudes. Les quantiles et le MSEP sont utilisés pour déterminer le montant de ce capital requis.
- Communication du risque : Présenter les réserves avec leurs quantiles (par exemple, 75 % et 95 %) permet de communiquer de manière transparente l'incertitude dans la détermination des réserves.

7. Implémentation manuelle de Bornhuetter Ferguson (BF)

La méthode Bornhuetter-Ferguson (BF) est une approche de provisionnement qui combine deux techniques classiques :

- **Chain-Ladder (CL)**, basé sur l'hypothèse que les tendances passées vont se reproduire dans le futur.
- **Loss Ratio (LR)**, qui repose sur un ratio sinistres/primes prédéfini, souvent issu d'analyses internes ou de données marché.

L'idée centrale de la méthode BF est de **pondérer simultanément** ces deux approches afin de limiter les faiblesses de chacune. La méthode Bornhuetter-Ferguson est une approche déterministe qui **équilibre prudence et réalisme** :

- elle protège contre les biais du Chain-Ladder lorsque l'historique est peu développé,
- tout en intégrant progressivement l'information empirique lorsque l'historique est suffisant.

7.1. Facteurs de développement cumulés et schéma de paiement de paiement

Calcul du CDF et de la proportion payée

On calcule les **facteurs cumulés** depuis chaque colonne jusqu'à l'ultime en multipliant les facteurs de développement successifs. Le **CDF** et la **proportion payée** sont simplement l'inverse de ces facteurs cumulés, indiquant la fraction du montant total déjà réglée à chaque étape.

```r
# ------------------------------
# Facteurs cumulés et schéma de dévelopement
# ------------------------------

# Facteurs cumulés jusqu'à l'ultime
facteurs_cumules = numeric(k)
facteurs_cumules[k] = 1
for (j in (k - 1):1) {
  facteurs_cumules[j] = facteurs_cumules[j + 1] * facteurs_dev[j]
}

# Pattern cumulé (CDF) et proportion payée à date
cdf = 1 / facteurs_cumules
proportion_payee = cdf  # même objet sémantiquement

list(facteurs_cumules = facteurs_cumules, proportion_payee = proportion_payee)
```

```
## $facteurs_cumules
##  [1] 1.202703 1.002523 1.001360 1.000155 1.000122 1.000051 1.000008 1.000006
##  [9] 1.000001 1.000001 1.000001 1.000000 1.000000 1.000000 1.000000 1.000000
## [17] 1.000000 1.000000 1.000000 1.000000
##
## $proportion_payee
##  [1] 0.8314603 0.9974834 0.9986419 0.9998453 0.9998783 0.9999493 0.9999923
##  [8] 0.9999941 0.9999992 0.9999992 0.9999992 1.0000000 1.0000000 1.0000000
## [15] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
```

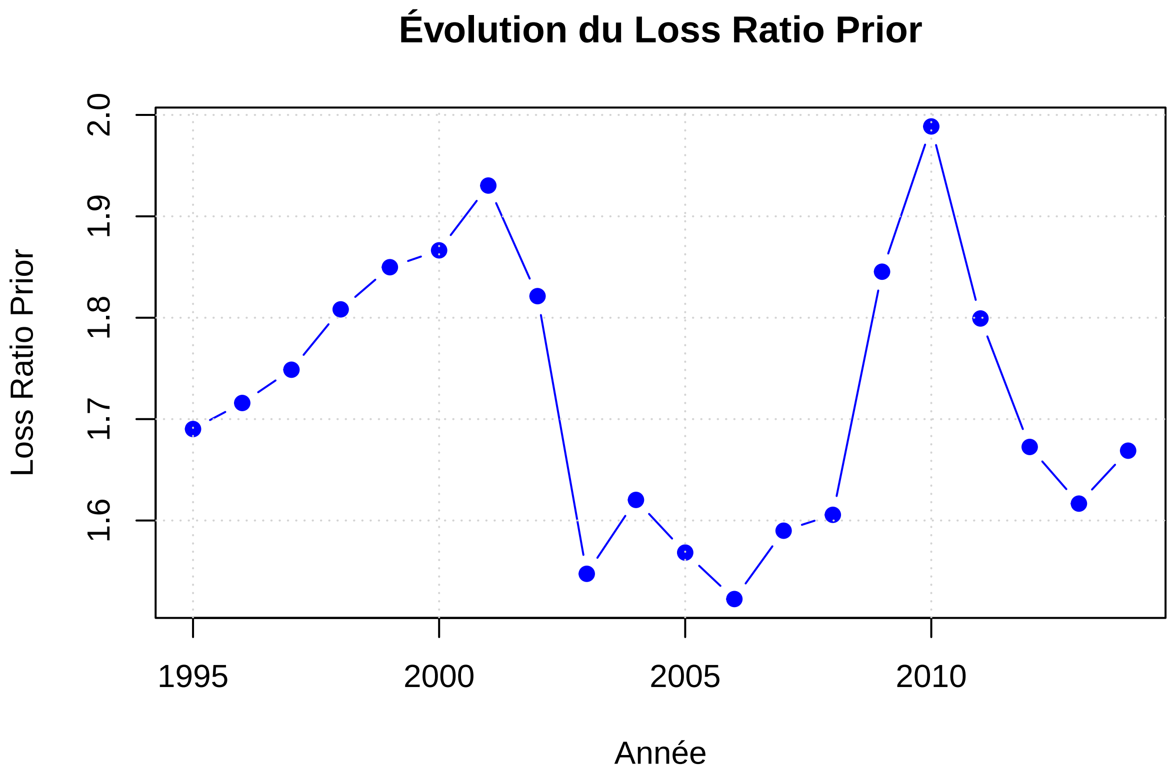
7.2. Le ratio de sinistralité (Loss ratio)

Dans cette partie, on cherche à estimer le **taux de sinistralité a priori** (Loss Ratio).

- Concrètement, on compare les **paiements observés** dans le triangle avec les **primes émises** chaque année.
- Cela donne un premier indicateur de la charge des sinistres par rapport aux primes encaissées.

Le Loss Ratio présente une forte variabilité, avec une progression jusqu'en 2001, une baisse en 2002–2003, un pic en 2010, puis une stabilisation autour de 1,6–1,7. Par souci de prudence et de cohérence avec la tendance récente, nous avons retenu la valeur de **1.70**. Cette valeur servira de référence dans les méthodes de projection ultérieures, en particulier pour la **Bornhuetter-Ferguson**.

```r
# -------------------------------
# Détermination du loss ratio
# -------------------------------
# Calcul du taux de sinistralité global a priori à partir de données agrégées

# Somme totale des paiements cumulés à la dernière période observée (par année)
paiements_totaux_par_annee = sapply(1:n, function(i) {
  last_dev = max(which(!is.na(triangle_cumule[i, ])))
  triangle_cumule[i, last_dev]
})

# Somme totale des primes (GWP) par année
primes_emises_par_annee = data_freq$GWP[match(rownames(triangle_cumule),
                                              data_freq$Year)]

# Calcul du loss ratio a priori global
loss_ratio_prior = charge_ultime_cl/primes_emises_par_annee

# -------------------------------
# Graphe de l'évolution du loss ratio
# -------------------------------
# Plot basique
plot(
  as.numeric(names(loss_ratio_prior)),loss_ratio_prior,type = "b",col = "blue",
  pch = 19,
  xlab = "Année",
  ylab = "Loss Ratio Prior",
  main = "Évolution du Loss Ratio Prior"
)
grid()
```




```r
loss_ratio=1.7

cat("Taux de sinistralité (loss ratio) a priori estimé :", loss_ratio, "\n")
```

```
## Taux de sinistralité (loss ratio) a priori estimé : 1.7
```

7.3. Ultimes BF et réserves

L'objectif de cette étape est d'appliquer la **méthode Bornhuetter-Ferguson (BF)** pour estimer la **charge ultime** et en déduire les **réserves nécessaires**.

Étapes principales

1. **Détermination de la charge a priori par année**

   - On calcule d'abord une charge ultime théorique basée uniquement sur le **Loss Ratio retenu (2.30)** et les **primes émises** :

$$\text{Charge a priori}_i = \text{Primes émises}_i \times \text{Loss Ratio}$$

2. **Combinaison observation / a priori selon Bornhuetter-Ferguson**

   - Pour chaque année i, l'ultime BF est obtenu en ajoutant :

- les **paiements déjà observés** (dernier cumulé disponible dans le triangle),
- plus la partie **non encore payée**, estimée via la charge a priori et pondérée par la proportion non encore développée.

Formellement :

$$\text{Ultime BF}_i = \text{Observé}_i + (1 - \text{Proportion payée}) \times \text{Charge a priori}_i$$

La **proportion déjà payée** est dérivée du facteur de développement (Chain-Ladder).

3. **Estimation des réserves**

   - Une fois l'ultime BF déterminé, la réserve par année correspond simplement à :

$$\text{Réserve}_i = \text{Ultime BF}_i - \text{Observé}_i$$

   - La somme sur toutes les années donne la **réserve totale Bornhuetter-Ferguson**.

Ici, on applique la méthode **Bornhuetter-Ferguson** pour estimer les ultimes et les réserves.

- L'idée est simple : on part de ce qui est **déjà observé** dans le triangle,
- et on ajoute une partie **non encore payée**, estimée grâce au **Loss Ratio a priori** et à la cadence de développement.

Concrètement, pour chaque année :

- si l'année est ancienne et presque intégralement payée, on fait surtout confiance aux observations,
- si l'année est récente, on s'appuie davantage sur l'estimation a priori.

Cela permet d'obtenir une estimation **plus robuste** que le Chain-Ladder seul, en évitant de sous-estimer les charges des années récentes.

```
# -------------------------------
# Calcul des réserves
# -------------------------------
# Dernier cumulé observé par ligne (déjà calculé) : 'dernier_cumule_observe'
# Colonne observée la plus à droite par ligne
indice_dernier_dev = apply(triangle_cumule, 1,
                           function(x) max(which(!is.na(x))))

charge_ultime_apriori = primes_emises_par_annee * loss_ratio


# Pour chaque ligne i : BF = Obs + (1 - proportion_payee[dernier_dev]) *a priori
ultime_bf = rep(NA_real_, n)
for (i in 1:n) {
  j = indice_dernier_dev[i]
```

```r
  proportion_deja_payee = proportion_payee[j]
  ultime_bf[i] = dernier_cumule_observe[i] +
    (1 - proportion_deja_payee) * charge_ultime_apriori[i]
}


reserve_bf_par_annee = ultime_bf - dernier_cumule_observe
reserve_bf_totale = sum(reserve_bf_par_annee, na.rm = TRUE)

resultats_bf=data.frame(
  Annee_Origine = rownames(triangle_cumule),
  Charge_BF = format_p(round(ultime_bf, 2)),
  Reserve_BF = format_p(round(reserve_bf_par_annee, 2))
)
# ------------------------------
# Affichage des résultats
# ------------------------------

print("Réserves Bornhuetter-Ferguson :")
```

```
## [1] "Réserves Bornhuetter-Ferguson :"
```

```r
print(resultats_bf)
```

```
##      Annee_Origine     Charge_BF    Reserve_BF
## 1995          1995 43 439 234,00          0,00
## 1996          1996 45 471 432,00          0,00
## 1997          1997 48 963 779,00          0,00
## 1998          1998 53 885 165,00          0,00
## 1999          1999 59 748 767,00          0,00
## 2000          2000 63 459 297,00          0,00
## 2001          2001 67 175 886,00          0,00
## 2002          2002 64 655 175,00          0,00
## 2003          2003 60 659 758,00          0,00
## 2004          2004 61 410 507,33         53,33
## 2005          2005 64 459 180,83         57,83
## 2006          2006 64 556 304,66         59,66
## 2007          2007 65 504 972,14        413,14
## 2008          2008 67 274 489,82        551,82
## 2009          2009 73 445 906,59      3 429,59
## 2010          2010 80 936 074,95      8 420,95
## 2011          2011 74 128 179,26     10 833,26
## 2012          2012 73 593 576,94    101 586,94
## 2013          2013 73 891 283,26    195 517,26
## 2014          2014 76 841 724,50 13 151 153,50
```

```
cat("Réserve totale (BF) :", format_p(reserve_bf_totale), "\n")
```

```
## Réserve totale (BF) : 13 472 077,27
```

8. Implémentation manuelle de Bootstrap CL

La méthode **Bootstrap Chain-Ladder** est une extension stochastique de la méthode Chain-Ladder classique. Elle permet de quantifier l'incertitude des provisions techniques en simulant non seulement une estimation moyenne, mais aussi la variabilité des résultats.

Le principe est le suivant : on calcule d'abord les valeurs attendues et les résidus à partir du modèle Chain-Ladder. Ces résidus sont ensuite rééchantillonnés afin de créer de nombreux **pseudo-triangles**. Pour chacun, on réestime les facteurs de développement et on projette les sinistres futurs en introduisant également de l'aléa de processus. La répétition de cette opération un grand nombre de fois fournit une **distribution empirique des réserves**, à partir de laquelle on peut calculer la moyenne, l'écart-type et des intervalles de confiance.

Cette méthode fournit donc un cadre cohérent pour évaluer à la fois le montant attendu des provisions et l'incertitude associée, ce qui est essentiel pour la gestion des risques.

8.1. Résidus standardisés (Pearson like)

Cette section vise à préparer les résidus nécessaires au bootstrap. On commence par compléter le triangle cumulé attendu grâce aux facteurs Chain-Ladder, puis on en déduit les incrémentaux attendus.

Les **résidus standardisés** sont calculés comme l'écart entre les observations et les valeurs attendues, pondéré par la racine carrée de l'espérance. Ils sont ensuite corrigés afin de refléter correctement la variance, et un paramètre de dispersion ϕ est estimé.

Ces résidus ajustés et le paramètre ϕ seront utilisés dans la suite du bootstrap pour réintroduire de l'aléa et simuler les sinistres futurs.

```
set.seed(314)
R= 10000 # nombre de simulations

# --------------------------------
# Détermination des résidus ajustés
# --------------------------------

# Estimation des paiements avec Chain Ladder
triangle_cum_esp = triangle_cumule
for (j in 2:k) {
  for (i in 1:n) {
      triangle_cum_esp[i,j] = triangle_cum_esp[i,j-1] * facteurs_dev[j-1]
  }
}

# Triangle des valeurs prévues
mu_hat = triangle_cum_esp
```

```r
#  Calcul des résidus et ajustement

inc_triangle =triangle  # incrémental

exp_inc = t(apply(mu_hat, 1, function(x) c(x[1], diff(x, lag = 1))))

residuals =(inc_triangle - exp_inc)/sqrt(abs(exp_inc))
residuals[is.nan(residuals)]= 0
residuals[is.infinite(residuals)]=0
nobs = 0.5 * k * (k + 1)
scale.factor =(nobs - 2*k + 1)
residuals_adj = residuals * sqrt(nobs / scale.factor)

# -------------------------------
# Calcul du paramètre de dispertion
# -------------------------------


phi = sum(residuals^2, na.rm=TRUE)/scale.factor
```

8.2. Boucle de Bootstrap et Estimation des Réserves

Cette section met en œuvre le cœur de la méthode Bootstrap Chain-Ladder : la simulation de nombreux triangles de sinistres afin d'obtenir une distribution empirique des provisions futures.

1. **Définition des zones observées et futures** Le triangle est séparé en deux parties : les cellules déjà observées et celles qui restent à estimer. Cette distinction permet de cibler le rééchantillonnage des résidus et la projection des sinistres futurs.

2. **Boucle de simulation** Pour chaque itération bootstrap (parmi R) :

   - Les **résidus ajustés** sont rééchantillonnés aléatoirement.
   - Un **pseudo-triangle incrémental** est reconstruit en réinjectant ces résidus aux valeurs attendues.
   - Ce triangle est converti en cumulatif puis complété par la méthode **Chain-Ladder** pour estimer les cellules manquantes.
   - Les sinistres futurs sont simulés avec un bruit de **processus (Poisson)** afin de refléter l'aléa de réalisation.
   - Les **réserves simulées** sont calculées, à la fois par année d'origine et au total.

3. **Extraction des statistiques** Après toutes les itérations, on obtient la distribution bootstrap des réserves. À partir de celle-ci, on calcule :

   - la réserve moyenne,
   - l'écart-type (incertitude),
   - la mesure d'erreur prédictive,
   - des quantiles (75 % et 95 %).

Ainsi, cette boucle bootstrap permet de passer d'une estimation déterministe (Chain-Ladder classique) à une vision probabiliste, où l'on dispose non seulement d'une valeur centrale, mais aussi d'une mesure de la variabilité et du risque associé aux provisions.

```
# -------------------------------
#  Indices des cellules observées et futures
# -------------------------------
left_upper = row(triangle_cumule) + col(triangle_cumule) <= n + 1
right_lower = !left_upper
mask=left_upper
mask[,1]=FALSE

# -------------------------------
# Boucle bootstrap
# -------------------------------
sim_reserves = numeric(R)
sim_reserves_indiv = matrix(NA, nrow = n, ncol = R)

for (b in 1:R) {

  #  Resample des résidus
  resampled_residuals = residuals_adj
  resampled_residuals[left_upper] = sample(residuals_adj[left_upper],
              size=length(residuals_adj[left_upper]), replace=TRUE)


  #  Reconstruction du triangle incrémental simulé
  pseudo_triangle_inc= inc_triangle

  pseudo_triangle_inc[!is.na(triangle)] = exp_inc[!is.na(triangle)] +
    resampled_residuals[!is.na(triangle)] * sqrt(abs(exp_inc[!is.na(triangle)]))

  #  Transformation en triangle cumulatif
  pseudo_triangle_cum = t(apply(pseudo_triangle_inc, 1, cumsum))

  #  Complétion des futures cellules via Chain-Ladder
  pseudo_triangle_full = pseudo_triangle_cum

  for (j in 2:k) {
    for (i in 1:n) {
      if (is.na(pseudo_triangle_full[i,j])) {
      pseudo_triangle_full[i,j]=pseudo_triangle_full[i,j-1] * facteurs_dev[j-1]
      }
    }
  }

  # Réserve par année et totale
```

```r
pseudo_triangle_char = t(apply(pseudo_triangle_full, 1,
                               function(x) c(x[1], diff(x, lag = 1))))
  future_cells = matrix(NA, n, n)
  future_cells[right_lower] = pseudo_triangle_char [right_lower]
  sim_counts = rpois(1, lambda = sum(future_cells, na.rm=TRUE)/phi) * phi
  sim_reserves[b] = sim_counts

  for (i in 1:n) {
    sim_reserves_indiv[i, b] = sum(future_cells[i, ], na.rm = TRUE)
  }
}

# -------------------------------
#  Statistiques bootstrap
# -------------------------------
reserve_moyenne = mean(sim_reserves)
reserve_origine_moyenne= rowMeans(sim_reserves_indiv)
se_bs =sd(sim_reserves)
se_origine=apply(sim_reserves_indiv,1,sd)
PE_bs = sqrt(phi * reserve_moyenne + se_bs^2)

reserve_q = quantile(sim_reserves, probs = c(0.75, 0.95), na.rm = TRUE)

resultats_boot = data.frame(
  Annee_Origine = rownames(triangle_cumule),
  Reserve_Moyenne = format_p(round(reserve_origine_moyenne, 2)),
  Ecart_type = format_p(round(se_origine, 2))
)


print("Réserves Bootsrap :")
```

```
## [1] "Réserves Bootsrap :"
```

```r
print(resultats_boot)
```

```
##    Annee_Origine Reserve_Moyenne Ecart_type
## 1           1995            0,00       0,00
## 2           1996            0,00       0,00
## 3           1997            0,00       0,00
## 4           1998            0,00       0,00
## 5           1999            0,00       0,00
## 6           2000            0,00       0,00
## 7           2001            0,00       0,00
## 8           2002            0,00       0,00
## 9           2003            0,00       0,00
## 10          2004           52,09       1,90
## 11          2005           53,47       1,69
```

```
## 12          2006           54,20       1,95
## 13          2007          387,44      13,92
## 14          2008          530,07      18,22
## 15          2009        3 667,28     116,25
## 16          2010        9 628,93     298,05
## 17          2011       11 726,30     373,28
## 18          2012      102 691,07   3 464,46
## 19          2013      189 805,63   6 618,23
## 20          2014   12 903 108,02 458 897,72
```

```r
cat("\nRéserve totale moyenne (bootstrap) :",
    format_p(round(reserve_moyenne, 2)), "\n")
```

```
##
## Réserve totale moyenne (bootstrap) : 13 217 032,00
```

```r
cat("Quantile 75% total :", format_p(round(reserve_q[1], 2)), "\n")
```

```
## Quantile 75% total : 13 895 967,86
```

```r
cat("Quantile 95% total :", format_p(round(reserve_q[2], 2)), "\n")
```

```
## Quantile 95% total : 15 033 649,44
```

9. Méthodes standards R (références)

Dans cette section, nous allons utiliser les fonctions déjà implémentées dans le package **ChainLadder** de R afin de comparer nos résultats manuels avec ceux obtenus par des méthodes standards reconnues.

L'objectif est double :

- **Vérification** : s'assurer que notre implémentation manuelle est cohérente avec les résultats du package de référence.
- **Comparaison** : observer les éventuelles différences (par exemple sur la variance ou les intervalles de confiance) et analyser leur origine.

Cette étape constitue donc un point de contrôle important, permettant de valider la robustesse de notre approche et de mettre en perspective les résultats obtenus par programmation manuelle avec ceux issus d'outils actuariels utilisés en pratique.

9.1. Chain Ladder (package)

```r
modele_cl = chainladder(triangle_cumule)
```

9.2. Mack Chain Ladder (package)

```
modele_mack = MackChainLadder(triangle_cumule, est.sigma = "Mack")
```

9.3. Bornhuetter Ferguson (package)

```
#La méthode de Bornhuetter-Ferguson n’est actuellement pas implémentée
#directement dans R via un package standard.
```

9.4. Bootstrap (package)

```
modele_boot = BootChainLadder(triangle_cumule, R = 10000,
                              process.distr = "od.pois")
```

Explication : Bootstrap **package** pour obtenir distributions des IBNR/ultimes selon un schéma paramétrique (ici, gamma).

10. Comparaison synthétique des résultats

```
# ------------------------------
# Résultats
# ------------------------------

# Calcul des réserves finales pour chaque méthode
reserves_cl_manuel = reserve_totale_cl
reserves_mack_manuel = reserve_totale_cl
reserves_bf_manuel = reserve_bf_totale
reserves_bootstrap_manuel = reserve_moyenne

# Extraction des réserves des modèles R standards
# Correction de la ligne pour le modèle chainladder
reserves_cl_package = sum( apply(predict(modele_cl), 1, function(x)
  tail(na.omit(x), 1))-apply(modele_cl$Triangle, 1, function(x)
    tail(na.omit(x), 1)))
reserves_mack_package = sum(summary(modele_mack)$ByOrigin[["IBNR"]])
# Le package 'ChainLadder' n'a pas de fonction intégrée pour BF.
# Ici, nous utilisons l'implémentation manuelle ou une alternative.
reserves_bf_package = NA
reserves_bootstrap_package = sum(summary(modele_boot)$ByOrigin[["Mean IBNR"]])


# ------------------------------
# Construction du tableau de comparaison
```

```r
# ------------------------------
# Création du tableau de comparaison
comparaison = data.frame(
  Methode = c("Chain Ladder (Manuel)", "Chain Ladder (Package)",
              "Mack (Manuel)", "Mack (Package)",
              "Bornhuetter Ferguson (Manuel)", "Bornhuetter Ferguson (Package)",
              "Bootstrap (Manuel)", "Bootstrap (Package)"),
  Reserves = c(reserves_cl_manuel, reserves_cl_package,
               reserves_mack_manuel, reserves_mack_package,
               reserves_bf_manuel,reserves_bf_package,
               reserves_bootstrap_manuel, reserves_bootstrap_package),
  Uncertainty = c("N/A", "N/A",
                  format_p(sqrt(MSEP_total)),
                  format_p(summary(modele_mack)$Totals[5,1]),
                  "N/A", "N/A",
                  format_p(sd(sim_reserves)),
                  format_p(summary(modele_boot)$Totals[4,1]))
)

# Ajout de la mise en forme
comparaison$Reserves = format_p(comparaison$Reserves)

# ------------------------------
# Affichage des résultats
# ------------------------------
kable(comparaison,
      caption = "Comparaison des réserves finales par méthode",
      col.names = c("Méthode", "Réserve Totale", "Mesure d'Incertitude")) %>%
  kable_styling(latex_options = c("scale_down", "hold_position"))
```

Table 9: Comparaison des réserves finales par méthode

Méthode	Réserve Totale	Mesure d'Incertitude
Chain Ladder (Manuel)	13 222 270,06	N/A
Chain Ladder (Package)	13 222 270,06	N/A
Mack (Manuel)	13 222 270,06	1 548 939,10
Mack (Package)	13 222 270,06	1 548 939,10
Bornhuetter Ferguson (Manuel)	13 472 077,27	N/A
Bornhuetter Ferguson (Package)	NA	N/A
Bootstrap (Manuel)	13 217 032,00	1 133 837,69
Bootstrap (Package)	13 228 326,33	1 193 739,55

11. Conclusion

- Les résultats manuels sont cohérents avec ceux des fonctions `ChainLadder`, confirmant ainsi le **bon paramétrage** et la **compréhension** des hypothèses.

- La méthode de **Mack** apporte un **cadre de variance** et permet le calcul des **quantiles** utiles pour l'analyse du risque de réserve.
- **Bornhuetter Ferguson (BF)** permet d'introduire une **connaissance a priori** (primes/LR ou ultimes) afin de modérer les extrapolations Chain Ladder en queue.
- Le **Bootstrap** fournit une **distribution** des réserves, pratique pour des analyses de **quantiles** et de **capital**.