Run-off triangles: Deterministic vs Stochastic methodologies

Davide Zicca

A digression about the Non-Life industry

In the Non-Life industry, the typical contracts expire in 12 months. But the claims payment process can take years or even decades. Moreover, often not even the delivery date of their product is known to insurers. The biggest item on the liabilities side of an insurer's balance sheet is often the provision or reserves for future claims payments. Those reserves can be broken down into case reserves (or outstanding claims), which are losses already reported to the insurance company and losses that are incurred but not reported (IBNR) yet. We have basically two different ways to assess the reserving problems: via deterministic or stochastic models. Solvency II in Europe, have fostered further research and promoted the use of stochastic and statistical techniques. In particular, for many countries, extreme percentiles of reserve deterioration over a fixed time period have to be estimated for the purpose of capital setting. I'll show how the data are structured for a typical reserving case. Then there's the discussion of the classical deterministic chain-ladder reserving method with an introduction of the concept of a tail factor. The chain-ladder algorithm can be considered a weighted linear regression through the origin, and move on from there to introduce stochastic reserving models. The Mack model, which provides a stochastic framework for the chain-ladder methods and allows the estimation of the mean squared error of the payment predictions. To estimate the full distribution of the reserve, we consider a bootstrap approach. We will do this analysis for two triangles taken from: https://www.casact.org/research/index.cfm?fa=loss_reserves_data and in particular, the csv file "PP Auto Data Set". I've then selected info that corresponds to "Germania Insurance Group" and "Battle Creek Mutual Insurance Co.". For which regards "Battle Creek Mutual Insurance Co.", I'll comment the results obtained, due to the fact that I've followed the same methodology applied to "Germania Insurance Group" (all infos are available in the R files and I've also saved all the plots that can be provided, if needed). We initialize our data, in order to have the first column that holds the origin year, the second column the development year and the third column has the incremental payments / transactions, for "Germania Insurance Group" in this way:

```
#########################################################################################
#                    Development Triangles                           #
#########################################################################################
library(ChainLadder)
```

```
##
## Welcome to ChainLadder version 0.2.12
##
## To cite package 'ChainLadder' in publications use:
##
##   Markus Gesmann, Daniel Murphy, Yanwei (Wayne) Zhang, Alessandro
##   Carrato, Mario Wuthrich, Fabio Concina and Eric Dal Moro (2021).
##   ChainLadder: Statistical Methods and Models for Claims Reserving in
##   General Insurance. R package version 0.2.12.
##   https://CRAN.R-project.org/package=ChainLadder
##
## To suppress this message use:
## suppressPackageStartupMessages(library(ChainLadder))
```

```r
library(lattice)
library(AER)
```

```
## Loading required package: car

## Loading required package: carData

## Loading required package: lmtest

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: sandwich

## Loading required package: survival
```

```r
library(fitdistrplus)
```

```
## Loading required package: MASS
```

```r
n<-10
Claims <- data.frame(originf = factor(rep(1988:1997, n:1)),
                     dev=sequence(n:1),
                     inc.paid=
                       c(2510,2653,2771,2799,2828,2810,2806,2803,2800,2804,
                         2500,2813,2903,2846,2839,2840,2871,2872,2873,
                         3648,4057,4059,4061,3996,3989,3983,3984,
                         4623,4998,4927,4831,4812,4804,4804,
                         5834,5585,5371,5216,5259,5242,
                         7248,6837,6698,6641,6564,
                         8757,8789,8542,8884,
                         9855,9234,9311,
                         10435,9604,
                         13768))

(inc.triangle <- with(Claims, {
  M <- matrix(nrow=n, ncol=n,
              dimnames=list(origin=levels(originf), dev=1:n))
  M[cbind(originf, dev)] <- inc.paid
  M
}))
```

```
##        dev
## origin     1    2    3    4    5    6    7    8    9   10
##   1988  2510 2653 2771 2799 2828 2810 2806 2803 2800 2804
##   1989  2500 2813 2903 2846 2839 2840 2871 2872 2873   NA
##   1990  3648 4057 4059 4061 3996 3989 3983 3984   NA   NA
##   1991  4623 4998 4927 4831 4812 4804 4804   NA   NA   NA
##   1992  5834 5585 5371 5216 5259 5242   NA   NA   NA   NA
##   1993  7248 6837 6698 6641 6564   NA   NA   NA   NA   NA
##   1994  8757 8789 8542 8884   NA   NA   NA   NA   NA   NA
##   1995  9855 9234 9311   NA   NA   NA   NA   NA   NA   NA
##   1996 10435 9604   NA   NA   NA   NA   NA   NA   NA   NA
```

```
##   1997 13768    NA    NA    NA    NA    NA    NA    NA    NA    NA
```

We will forecast the future claims development in the bottom right corner of the triangle and potential further developments beyond development age 10. Often it is helpful to consider the cumulative development of claims as well, which is presented below:

```
(cum.triangle <- t(apply(inc.triangle, 1, cumsum)))
```

```
##       dev
## origin     1     2     3     4     5     6     7     8     9    10
##   1988  2510  5163  7934 10733 13561 16371 19177 21980 24780 27584
##   1989  2500  5313  8216 11062 13901 16741 19612 22484 25357    NA
##   1990  3648  7705 11764 15825 19821 23810 27793 31777    NA    NA
##   1991  4623  9621 14548 19379 24191 28995 33799    NA    NA    NA
##   1992  5834 11419 16790 22006 27265 32507    NA    NA    NA    NA
##   1993  7248 14085 20783 27424 33988    NA    NA    NA    NA    NA
##   1994  8757 17546 26088 34972    NA    NA    NA    NA    NA    NA
##   1995  9855 19089 28400    NA    NA    NA    NA    NA    NA    NA
##   1996 10435 20039    NA    NA    NA    NA    NA    NA    NA    NA
##   1997 13768    NA    NA    NA    NA    NA    NA    NA    NA    NA
```

The latest diagonal of the triangle presents the latest cumulative paid position of all origin years:

```
(latest.paid <- cum.triangle[row(cum.triangle) == n - col(cum.triangle) + 1])
```

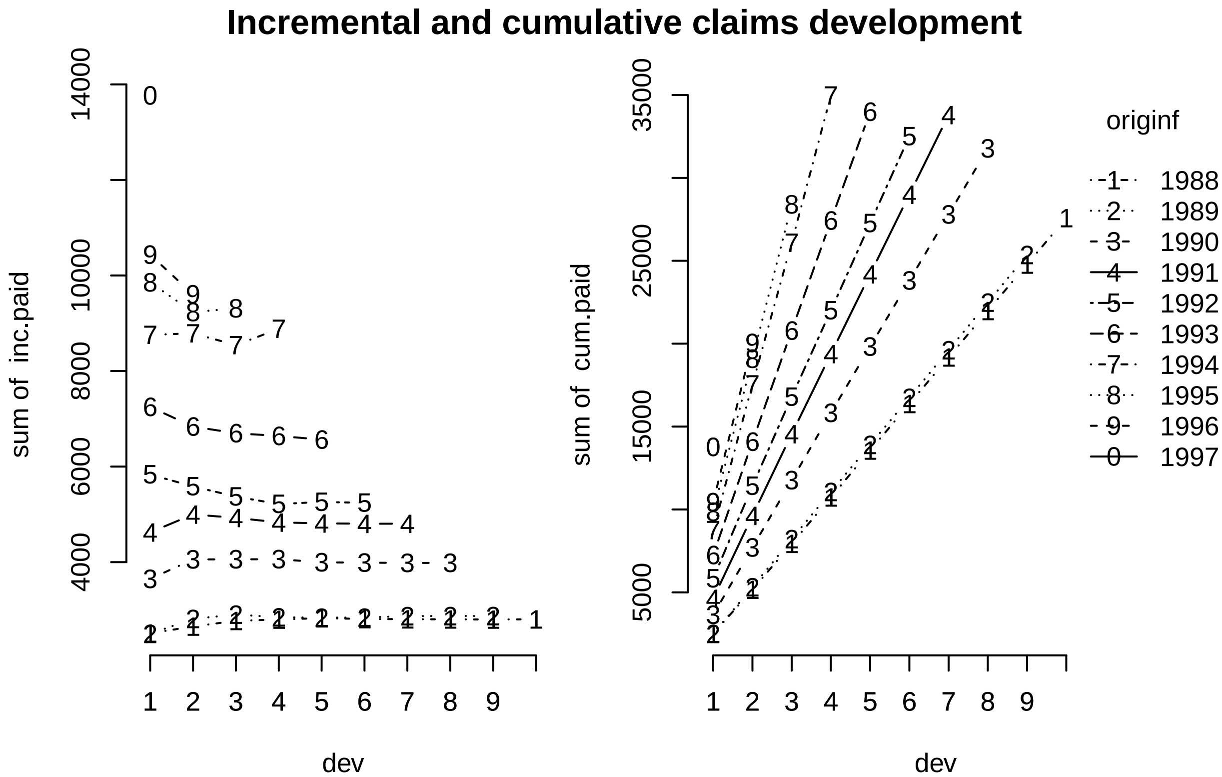
```
##  [1] 13768 20039 28400 34972 33988 32507 33799 31777 25357 27584
```

We add the cumulative paid data as a column to the data frame as well:

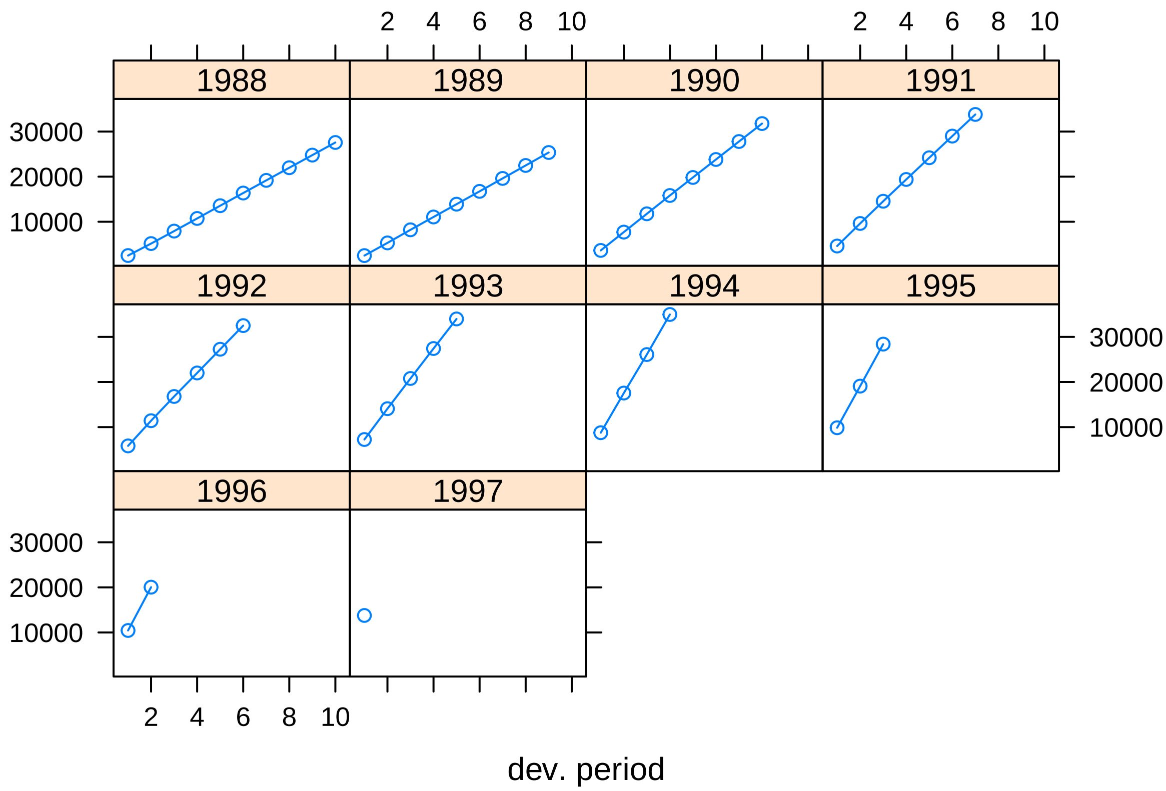
```
Claims$cum.paid <- cum.triangle[with(Claims, cbind(originf, dev))]
```

To start the reserving analysis, we plot the data:

```
op <- par(fig=c(0,0.5,0,1), cex=0.8, oma=c(0,0,0,0))

with(Claims, {
  interaction.plot(x.factor=dev, trace.factor=originf, response=inc.paid,
                   fun=sum, type="b", bty="n", legend=FALSE); axis(1, at=1:n)
  par(fig=c(0.45,1,0,1), new=TRUE, cex=0.8, oma=c(0,0,0,0))
  interaction.plot(x.factor=dev, trace.factor=originf, response=cum.paid,
                   fun=sum, type="b", bty="n"); axis(1,at=1:n)
})

mtext("Incremental and cumulative claims development",
      side=3, outer=TRUE, line=-3, cex = 1.1, font=2)
```




```r
par(op)

library(lattice)
plot(as.triangle(cum.triangle),lattice=TRUE)
```




```
#xyplot(cum.paid ~ dev | originf, data=Claims, t="b", layout=c(4,2),
#as.table=TRUE, main="Cumulative claims development")
```

We can see the incremental and cumulative claims development by origin year. The triangle appears to be fairly well behaved. The last year, 1997, appear to be considerably higher than years 1994 to 1996. The second graph is about the cumulative claims developments by origin year using the lattice package, with one panel per origin year. Using terminology in Wutherich & Merz (2008), cumulative payments are noted as

$$C_{i,j}$$

, for origin period i (or period of occurrence) seen after j periods of development and incremental payments

$$X_{i,j}$$

. The outstanding liabilities, or reserves, for accident year i at time j is given by:

$$R_{i,j} = \sum_{k>j} X_{i,j} = (\lim_{k\to\infty} C_{i,k}) - C_{i,j}$$

Because this quantity involves unobserved data (i.e. amounts that will be paid in the future),

$$R_{i,j}$$

will be the estimated claims reserves. Note that:

- For convenience, we will assume that we work on square matrices, with n rows and n columns.
- Throughout this chapter we note the first development period as 1. Other authors use 0 for the first development period. This assumption does not have any practical implications.

- Many of the methods and models presented here can be applied to paid and reported (often also called 'incurred') data. We either have to estimate the reserve or incurred but not reported (IBNR) claims. For the purpose of this chapter, we assume:

$$Ultimatelosscost = paid + reserve = paid + casereserve + IBNR = incurred + IBNR$$

.

- Some methods and models require data > 0, which for paid claims should be given (insurers rarely receive money back from the insured after a claim was paid, such as late salvage or subrogation payments), but case reserves can show negative adjustments over time; therefore incremental incurred triangles do show negatives occasionally. To ensure that data have only positive values, it can be temporarily shifted, or, in a given context, be ignored.

As previously stated, I'll now show the classical deterministic chain-ladder reserving method, also known as loss development factor (LDF) method. The idea is that, using this deterministic algorithm, we forecast claims based on historical data. It assumes that the proportional developments of claims from one development period to the next is the same for all origin periods. Most commonly as a first step, the age-to-age link ratios are calculated as the volume weighted average development ratios of a cumulative loss development triangle from one development period to the next

$$C_{i,k}, i, k = 1, ..., n$$

.

$$f_k = \frac{\sum_{i=1}^{n-k} C_{i,k+1}}{\sum_{i=1}^{n-k} C_{i,k}}$$

```
f <- sapply((n-1):1, function(i) {
  sum( cum.triangle[1:i, n-i+1] ) / sum( cum.triangle[1:i, n-i] )
})
f
```

```
## [1] 1.984840 1.495681 1.332426 1.247094 1.199364 1.168349 1.145069 1.127586
## [9] 1.113156
```

Initially we expect no further development after year 10. Hence, we set the last link ratio (often called the tail factor) to 1:

```
tail <- 1

(f <- c(f, tail))
```

```
##  [1] 1.984840 1.495681 1.332426 1.247094 1.199364 1.168349 1.145069 1.127586
##  [9] 1.113156 1.000000
```

These factors fk are then applied to the latest cumulative payment in each row

$$C_{i,n-i+1}$$

to produce stepwise forecasts for future payment years

$$k \in n - i + 1, ..., n$$

:

$$\hat{C}_{i,k+1} = f_k \hat{C}_{i,k}$$

starting with

$$C_{i,n-i+1} = C_{i,n-i+1}$$

. The squaring of the claims triangle is calculated below:

```
full.triangle <- cum.triangle

for(k in 1:(n-1)){
  full.triangle[(n-k+1):n, k+1] <- full.triangle[(n-k+1):n,k]*f[k]
}

full.triangle
```

```
##       dev
## origin     1        2        3        4        5        6        7         8
##   1988  2510  5163.00  7934.00 10733.00 13561.00 16371.00 19177.00  21980.00
##   1989  2500  5313.00  8216.00 11062.00 13901.00 16741.00 19612.00  22484.00
##   1990  3648  7705.00 11764.00 15825.00 19821.00 23810.00 27793.00  31777.00
##   1991  4623  9621.00 14548.00 19379.00 24191.00 28995.00 33799.00  38702.20
##   1992  5834 11419.00 16790.00 22006.00 27265.00 32507.00 37979.51  43489.16
##   1993  7248 14085.00 20783.00 27424.00 33988.00 40763.98 47626.54  54535.68
##   1994  8757 17546.00 26088.00 34972.00 43613.38 52308.32 61114.35  69980.16
##   1995  9855 19089.00 28400.00 37840.89 47191.15 56599.37 66127.79  75720.90
##   1996 10435 20039.00 29971.94 39935.38 49803.19 59732.15 69787.97  79912.06
##   1997 13768 27327.28 40872.88 54460.07 67916.85 81457.02 95170.19 108976.46
##       dev
## origin         9        10
##   1988  24780.00  27584.00
##   1989  25357.00  28226.29
##   1990  35831.31  39885.83
##   1991  43640.07  48578.19
##   1992  49037.79  54586.70
##   1993  61493.69  68452.06
##   1994  78908.67  87837.65
##   1995  85381.85  95043.30
##   1996  90107.74 100303.95
##   1997 122880.36 136784.99
```

The last column contains the forecast ultimate loss cost:

```
(ultimate.paid <- full.triangle[,n])
```

```
##      1988      1989      1990      1991      1992      1993      1994      1995
##  27584.00  28226.29  39885.83  48578.19  54586.70  68452.06  87837.65  95043.30
##      1996      1997
## 100303.95 136784.99
```

The cumulative products of the age-to-age development ratios provide the loss development factors for the latest cumulative paid claims for each row to ultimate:

```
(ldf <- rev(cumprod(rev(f))))
```

```
##  [1] 9.934993 5.005437 3.346595 2.511656 2.014007 1.679229 1.437267 1.255179
##  [9] 1.113156 1.000000
```

The inverse of the loss development factor estimates the proportion of claims developed to date for each origin year, often also called the gross up factors or growth curve:

```r
(dev.pattern <- 1/ldf)
```

```
##  [1] 0.1006543 0.1997828 0.2988112 0.3981436 0.4965227 0.5955114 0.6957649
##  [8] 0.7966989 0.8983469 1.0000000
```

The total estimated outstanding loss reserve with this method is:

```r
(reserve <- sum (latest.paid * (ldf - 1)))
```

```
## [1] 405092
```

Note that: The basic chain-ladder algorithm has the implicit assumption that each origin period has its own unique level and that development factors are independent of the origin periods; or equivalently, there is a constant payment pattern. Therefore, if ai is the ultimate (cumulative) claim for origin period i and bj is the percentage of ultimate claims in development period j, with

$$\sum b_j = 1$$

, then the incremental payment

$$X_{i,j}$$

can be described as

$$X_{i,j} = a_i b_j$$

.

```r
#  otherwise
#sum(ultimate.paid - latest.paid)

a <- ultimate.paid

(b <- c(dev.pattern[1], diff(dev.pattern)))
```

```
##  [1] 0.10065432 0.09912843 0.09902841 0.09933248 0.09837903 0.09898874
##  [7] 0.10025346 0.10093408 0.10164792 0.10165313
```

```r
(X.hat <- a %*% t(b))
```

```
##            [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
##  [1,]  2776.449  2734.359  2731.600  2739.987  2713.687  2730.505  2765.392
##  [2,]  2841.098  2798.028  2795.205  2803.787  2776.875  2794.085  2829.783
##  [3,]  4014.681  3953.820  3949.831  3961.958  3923.929  3948.248  3998.693
##  [4,]  4889.605  4815.480  4810.621  4825.392  4779.075  4808.694  4870.132
##  [5,]  5494.387  5411.093  5405.634  5422.232  5370.186  5403.468  5472.505
##  [6,]  6889.996  6785.545  6778.699  6799.513  6734.247  6775.983  6862.556
##  [7,]  8841.239  8707.208  8698.423  8725.131  8641.382  8694.937  8806.028
##  [8,]  9566.519  9421.493  9411.988  9440.887  9350.268  9408.216  9528.420
##  [9,] 10096.026  9942.974  9932.942  9963.440  9867.805  9928.962 10055.819
## [10,] 13768.000 13559.281 13545.600 13587.191 13456.774 13540.173 13713.169
##            [,8]      [,9]     [,10]
##  [1,]  2784.166  2803.856  2804.000
##  [2,]  2848.995  2869.144  2869.291
##  [3,]  4025.840  4054.312  4054.520
##  [4,]  4903.195  4937.872  4938.125
##  [5,]  5509.658  5548.624  5548.909
##  [6,]  6909.146  6958.010  6958.366
##  [7,]  8865.812  8928.514  8928.972
##  [8,]  9593.108  9660.954  9661.449
##  [9,] 10124.087 10195.688 10196.211
## [10,] 13806.267 13903.909 13904.622
```

Bornhuetter & Ferguson methodology

As the chain-ladder method is a deterministic algorithm and does not regard the observations as realizations of random variables but absolute values, the forecast of the most recent origin periods can be quite unstable. To address this issue, Bornhuetter & Ferguson (BF, 1972) suggested a credibility approach, which combines the chain-ladder forecast with prior information on expected loss costs, for example, from pricing data. Under this approach the chain-ladder development to ultimate pattern is used as weighting factors between the pure chain-ladder and expected loss cost estimates. Suppose the expected loss cost for the 1997 origin year is 20,000; then the BF method would estimate the ultimate loss cost as:

```
#Suppose the expected loss cost for the 1997 origin year is 20,000
(BF1997 <- ultimate.paid[n] * dev.pattern[1] + 20000 * (1 - dev.pattern[1]))
```

```
##     1997
## 31754.91
```

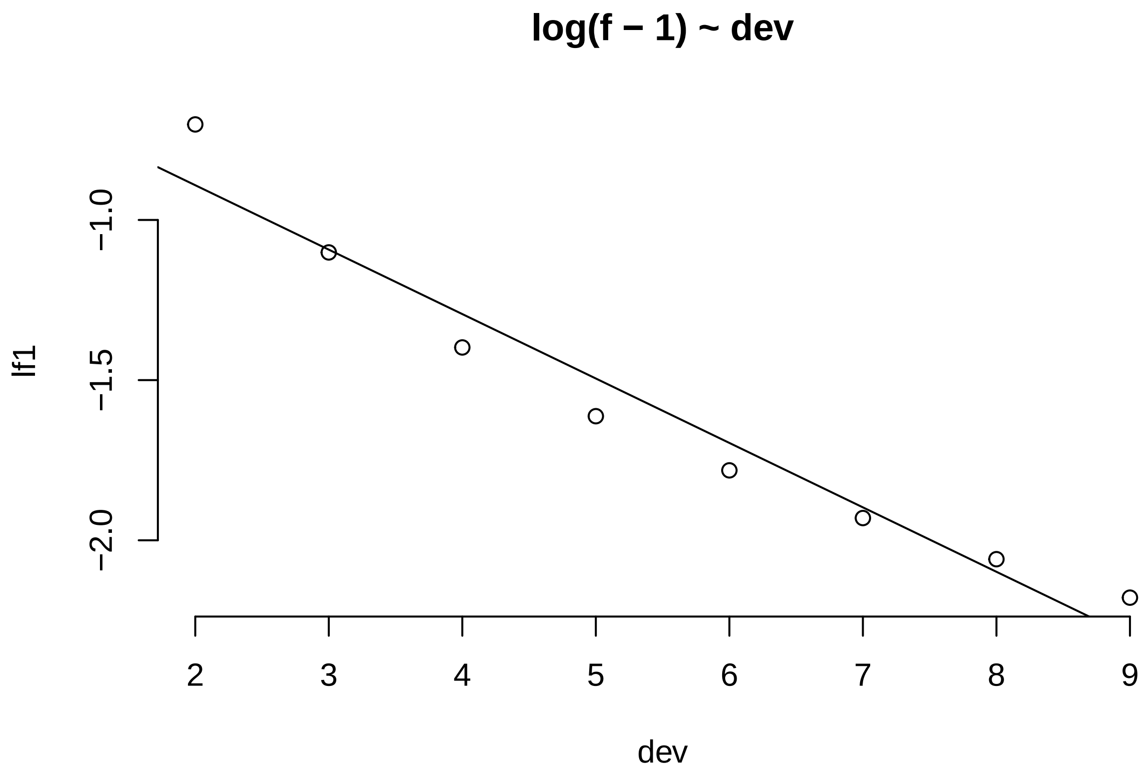
Tail Factors

In the previous section we implicitly assumed that there are no claims payment after 10 years, or in other words, that the oldest origin year is fully developed. However, often it is not suitable to assume that the oldest origin year is fully settled. A typical approach to overcome this shortcoming is to extrapolate the development ratios, for example, assuming a linear model of the log development ratios minus one, which reflects the incremental changes on the previous cumulative payments:

```
dat <- data.frame(lf1=log(f[-c(1,n)]-1), dev=2:(n-1))

(m <- lm(lf1 ~ dev , data=dat))
```

```
##
## Call:
## lm(formula = lf1 ~ dev, data = dat)
##
## Coefficients:
## (Intercept)          dev
##     -0.4893      -0.2011
```

```
plot(lf1 ~ dev, main="log(f - 1) ~ dev", data=dat, bty="n")

abline(m)
```




```r
sigma <- summary(m)$sigma

extrapolation <- predict(m, data.frame(dev=n:100))

(tail <- prod(exp(extrapolation + 0.5*sigma^2) + 1))
```

```
## [1] 1.558258
```

```r
#This is the plot of the loss development factors -1 on a log-scale against development period
```

More generally, the factors used to project the future payments need not always be drawn from the dollar weighted averages of the triangle. Other sources of factors from which the actuary may select link ratios include simple averages from the triangle, averages weighted toward more recent observations or adjusted for outliers, and benchmark patterns based on related, more credible loss experience. Also, because the ultimate value of claims is simply the product of the most current diagonal and the cumulative product of the link ratios, the completion of the interior of the triangle is usually not displayed; instead, the eventual value of the claims, or ultimate value is shown. For example, suppose the actuary decides that the volume weighted factors from the claims triangle are representative of expected future growth, but discards the tail factor derived from the linear fit in favour of a tail based on data from a larger book of similar business. The LDF method is as follows:

```r
library(ChainLadder)

ata(cum.triangle)
```

```
##       dev
## origin   1-2   2-3   3-4   4-5   5-6   6-7   7-8   8-9  9-10
##   1988 2.057 1.537 1.353 1.263 1.207 1.171 1.146 1.127 1.113
```

```
##   1989 2.125 1.546 1.346 1.257 1.204 1.171 1.146 1.128    NA
##   1990 2.112 1.527 1.345 1.253 1.201 1.167 1.143    NA    NA
##   1991 2.081 1.512 1.332 1.248 1.199 1.166    NA    NA    NA
##   1992 1.957 1.470 1.311 1.239 1.192    NA    NA    NA    NA
##   1993 1.943 1.476 1.320 1.239    NA    NA    NA    NA    NA
##   1994 2.004 1.487 1.341    NA    NA    NA    NA    NA    NA
##   1995 1.937 1.488    NA    NA    NA    NA    NA    NA    NA
##   1996 1.920    NA    NA    NA    NA    NA    NA    NA    NA
##   smpl 2.015 1.505 1.335 1.250 1.201 1.169 1.145 1.128 1.113
##   vwtd 1.985 1.496 1.332 1.247 1.199 1.168 1.145 1.128 1.113
```

Stochastic Reserving Models

As the provision for outstanding claims is often the biggest item on the liabilities side of an insurer's balance sheet, it is important not only to estimate the mean but also the uncertainty of the reserve. The key idea is to regard the observed data as one realization of a random variable, rather than absolutes. Statistical techniques also allow for more formal testing, make modelling assumptions more explicit and, in particular, help to monitor actual versus expected claims developments (A versus E). It is the regular A versus E exercise which can help to drive management actions. Hence, as a minimum, not only the mean reserve, or best estimate liabilities3, should be estimated but also the volatility of reserves. In this section we first show that the deterministic chain-ladder algorithm of the previous section can be considered a weighted linear regression through the origin. Indeed, the following Mack model provides a stochastic framework for the chain-ladder method and allows us to estimate the mean squared error of future payments, using many estimators from the linear regression output. An alternative to the Mack model is the Poisson model, a generalized linear model that replicates the chain-ladder forecasts as well. Yet, the Poisson model is often not directly applicable to insurance data, as the variance of the data is frequently greater than the mean and hence we consider a quasi-Poisson model to estimate uncertainty metrics. Following this, we present a bootstrap technique to estimate the full reserve distribution.

Chain-Ladder in the Context of Linear Regression Barnett & Zehnwirth (2000)

Let $C_{\cdot,k}$ denote the k-th column in the cumulative claims triangle. The chain-ladder algorithm can be seen as:

$$C_{\cdot,k+1} = f_k C_{\cdot,k} + \epsilon(k) \sim \mathcal{N}(0,\, \sigma_k^2 C_{\cdot,k}^{\delta})$$

The parameter fk describes the slope or the 'best' line through the origin and data points

$$(C_{\cdot,k}, C_{\cdot,k+1})$$

, with as a 'weighting' parameter. Barnett & Zehnwirth (2000) distinguish the cases:

- delta=0, ordinary regression with intercept 0
- delta=1, historical chain ladder age-to-age link ratios
- delta=2, straight averages of the individual link ratios

Indeed, we can demonstrate the different cases by applying different linear models to our data. First, we add columns to the original dataframe Claims, to have payments of the current and previous development period next to each other; additionally we add a column with the development period as a factor.

```r
##################################################################################
#                        Stochastic Reserving Models                    #
#                  Chain-Ladder in the Context of Linear Regression            #
##################################################################################

names(Claims)[3:4] <- c("inc.paid.k", "cum.paid.k")

ids <- with(Claims, cbind(originf, dev))

Claims <- within(Claims,{
  cum.paid.kp1 <- cbind(cum.triangle[,-1], NA)[ids]
  inc.paid.kp1 <- cbind(inc.triangle[,-1], NA)[ids]
  devf <- factor(dev)
}
)
```

In the next step we apply the linear regression function lm to each development period, vary the weighting parameter delta from 0 to 2 and extract the slope coefficients:

```r
delta <- 0:2

ATA <- sapply(delta, function(d)
  coef(lm(cum.paid.kp1 ~ 0 + cum.paid.k : devf,
          weights=1/cum.paid.k^d, data=Claims))
)

dimnames(ATA)[[2]] <- paste("Delta = ", delta)

ATA
```

```
##                  Delta =  0 Delta =  1 Delta =  2
## cum.paid.k:devf1   1.965004   1.984840   2.015225
## cum.paid.k:devf2   1.489876   1.495681   1.505314
## cum.paid.k:devf3   1.331220   1.332426   1.335315
## cum.paid.k:devf4   1.244770   1.247094   1.249881
## cum.paid.k:devf5   1.198127   1.199364   1.200722
## cum.paid.k:devf6   1.167763   1.168349   1.168966
## cum.paid.k:devf7   1.144806   1.145069   1.145317
## cum.paid.k:devf8   1.127589   1.127586   1.127584
## cum.paid.k:devf9   1.113156   1.113156   1.113156
```

Indeed, the development ratios for

$$\delta = 1$$

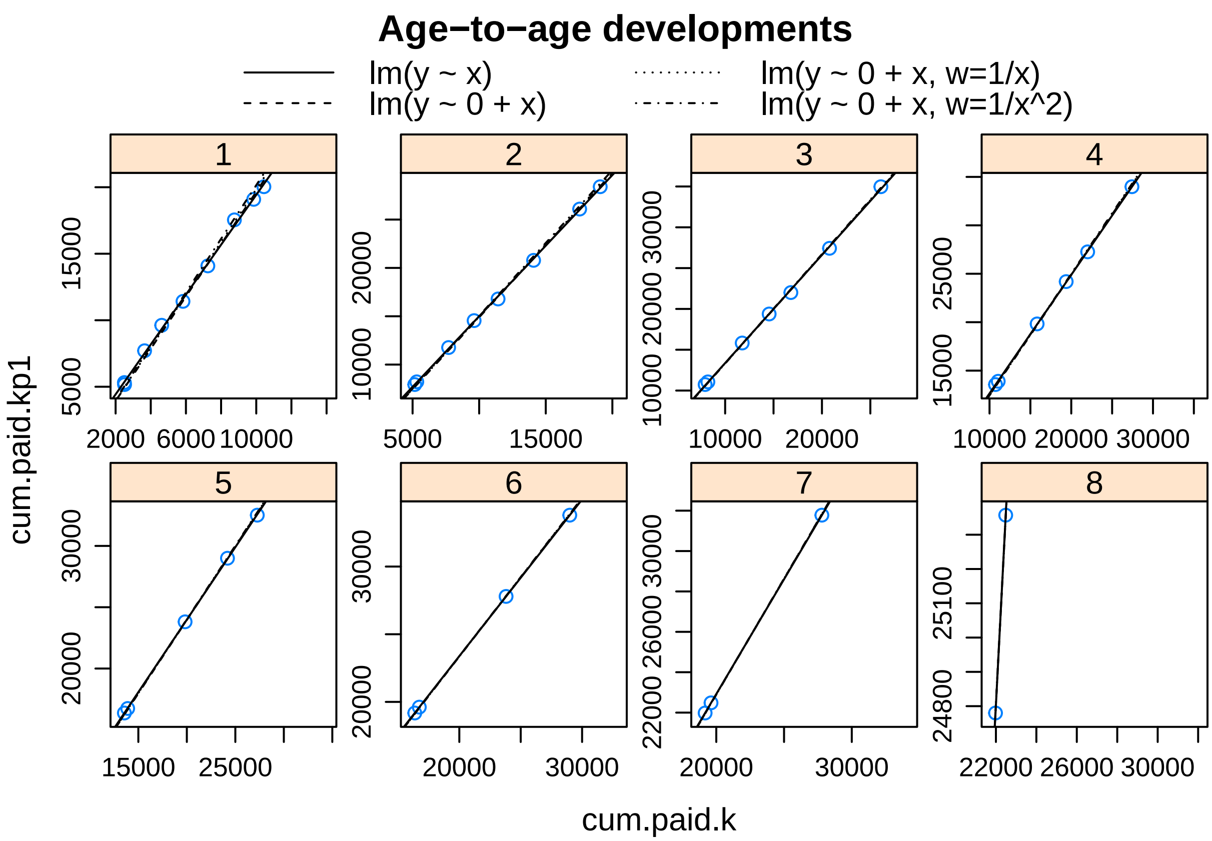
and

$$\delta = 2$$

correspond with those of the previous section. The following graph is the plot of the cumulative development positions from one development year to the next for each development year, including regression lines of different linear models:

```r
xyplot(cum.paid.kp1 ~ cum.paid.k | devf,
       data=subset(Claims, dev < (n-1)),
       main="Age-to-age developments", as.table=TRUE,
       scales=list(relation="free"),
       key=list(columns=2, lines=list(lty=1:4, type="l"),
                text=list(lab=c("lm(y ~ x)",
```

```
                                "lm(y ~ 0 + x)",
                                "lm(y ~ 0 + x, w=1/x)",
                                "lm(y ~ 0 + x, w=1/x^2)"))),
       panel=function(x,y,...){
         panel.xyplot(x,y,...)
         if(length(x)>1){
           panel.abline(lm(y ~ x), lty=1)
           panel.abline(lm(y ~ 0 + x), lty=2)
           panel.abline(lm(y ~ 0 + x, weights=1/x), lty=3)
           panel.abline(lm(y ~ 0 + x, weights=1/x^2), lty=4)
         }
       }
)
```



Note that for all development periods we observe no difference in the slope of the linear regression with and without an intercept.

Mack Model

Mack (1993, 1999) suggested a model to estimate the first two moments (mean and standard errors) of the chain-ladder forecast, without assuming a distribution under three conditions. Note that Mack uses the following notation for the weighting parameter alpha = 2 − delta , with delta defined as in the previous section. In other words, the Mack model assumes that the link ratios for each development period are consistent across all origin periods (CL1), the volatility decreases as losses are paid (CL2) and all origin periods are independent; for example, there is no structural change or market cycle (CL3). If these assumptions hold, the Mack model gives an unbiased estimator for future claims.

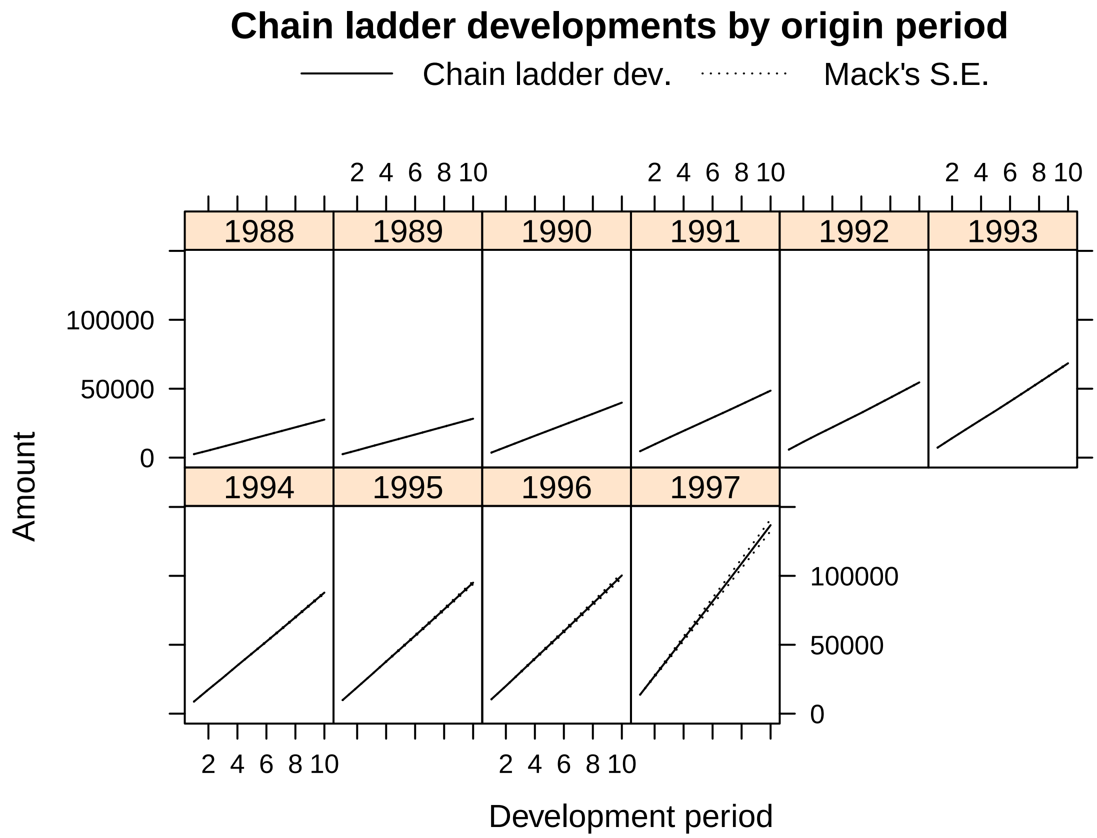
In R we have the following output:

```r
#############################################################################################
#                            Mack Model                                    #
#############################################################################################

library(ChainLadder)

(mack <- MackChainLadder(cum.triangle, weights=1, alpha=1,est.sigma="Mack"))
```

```
## MackChainLadder(Triangle = cum.triangle, weights = 1, alpha = 1,
##     est.sigma = "Mack")
##
##      Latest Dev.To.Date Ultimate    IBNR Mack.S.E CV(IBNR)
## 1988 27,584       1.000   27,584       0     0.00      NaN
## 1989 25,357       0.898   28,226   2,869     1.44 0.000503
## 1990 31,777       0.797   39,886   8,109    10.88 0.001342
## 1991 33,799       0.696   48,578  14,779    76.66 0.005187
## 1992 32,507       0.596   54,587  22,080   154.56 0.007000
## 1993 33,988       0.497   68,452  34,464   342.22 0.009930
## 1994 34,972       0.398   87,838  52,866   666.46 0.012607
## 1995 28,400       0.299   95,043  66,643 1,116.37 0.016751
## 1996 20,039       0.200  100,304  80,265 1,793.19 0.022341
## 1997 13,768       0.101  136,785 123,017 4,265.46 0.034674
##
##               Totals
## Latest:   282,191.00
## Dev:            0.41
## Ultimate: 687,282.96
## IBNR:     405,091.96
## Mack.S.E    5,305.39
## CV(IBNR):       0.01
```

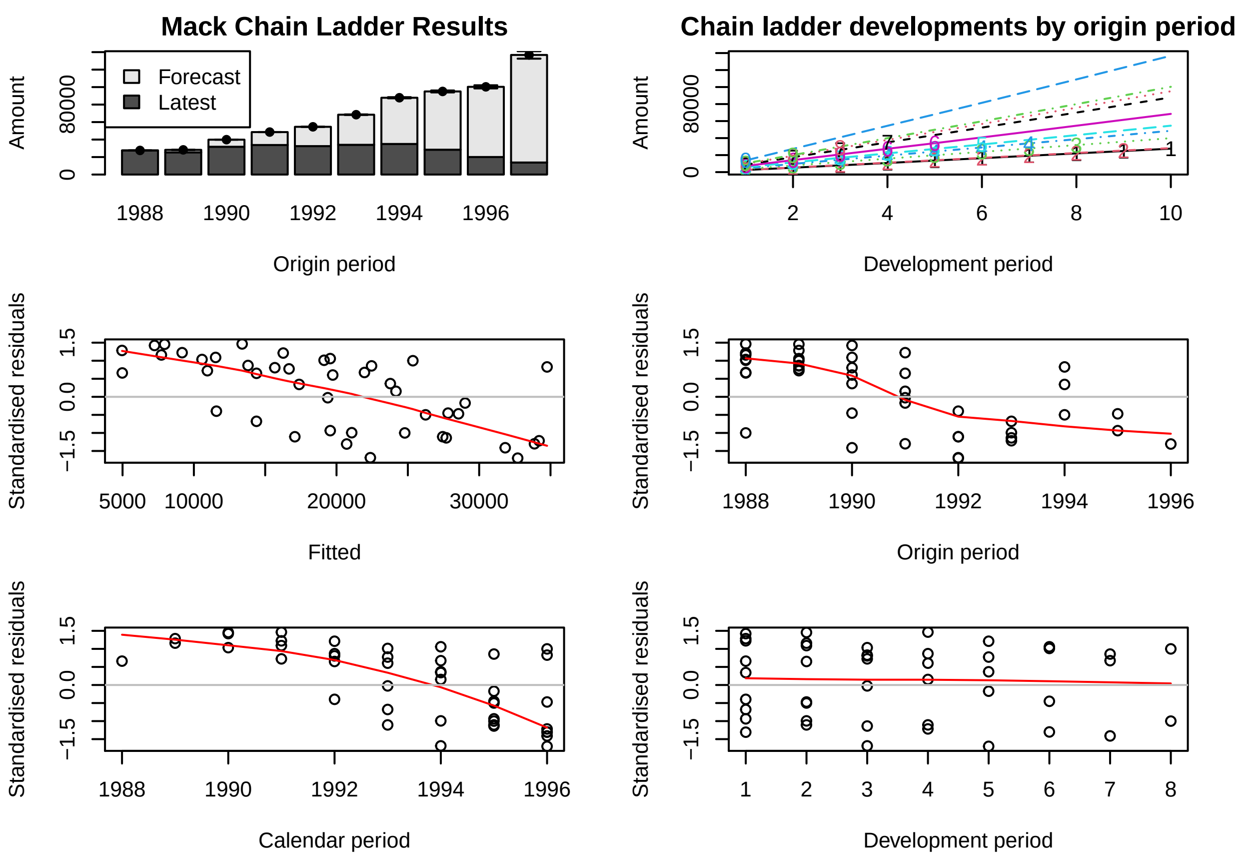
Plot of the actual and expected cumulative claims development and estimated standard error of the Mack model forecast:

```r
plot(mack, lattice=TRUE, layout=c(6,2))
```



Chain ladder developments by origin period
Chain ladder dev.
Mack's S.E.
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
Amount
Development period
100000
50000
0
2 4 6 8 10


```
plot(mack)
```



The top-left panel shows the latest actual position with the forecasts stacked on top and whiskers indicating the estimated standard error. The top-right panel presents the claims developments to ultimate for each origin year. The four residual plots show the standardized residuals against fitted values, origin, calendar and development period. The residual plots should not show any obvious patterns and about 95% of the standardized residuals should be contained in the range of -2 to 2 for the Mack model to be strictly applicable (and this is exactly what we got).

Bootstrap Chain-Ladder

An alternative to asymptotic econometric relationships can be to use the bootstrap methodology. We use a two-stage simulation approach, following P.D. England & R.J. Verrall (2002). In the first stage, a quasi-Poisson model is applied to the claims triangle to forecast future payments. From this we calculate the scaled-Pearson residuals, assuming that they are approximately independent and identical distributed. These residuals are re-sampled with replacement many times to generate bootstrapped (pseudo) triangles and to forecast future claims payments to estimate the parameter error. Recall that the predictions of the quasi-Poisson model are the same as those from the chain-ladder method, hence we use the latter faster algorithm. In the second stage, we simulate the process error with the bootstrap value as the mean and an assumed process distribution, here a quasi-Poisson. The set of reserves obtained in this way forms the predictive distribution, from which summary statistics such as mean, prediction error or quantiles can be derived.

This two-stage bootstrapping/simulation approach is implemented in the BootChainLadder function as part of the ChainLadder package. As input parameters we provide the cumulative triangle, the number of bootstraps and the process distribution to be assumed:

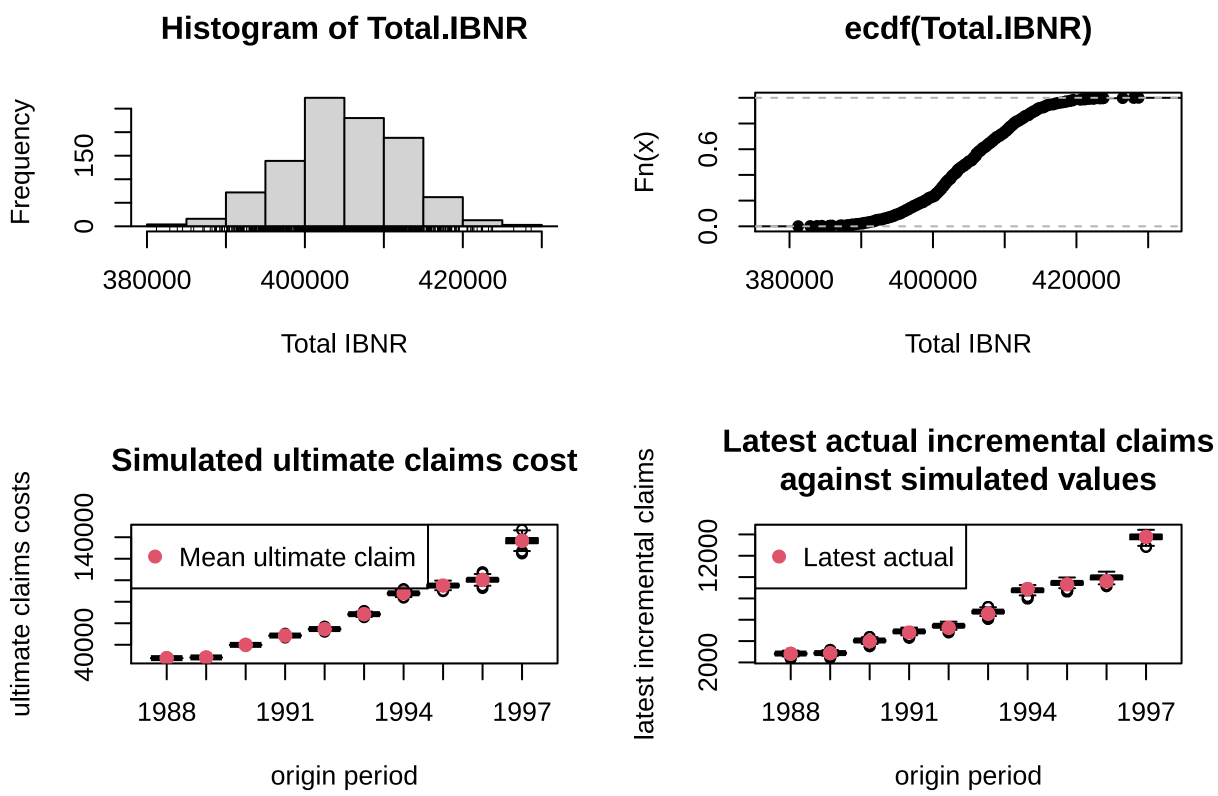
```
#################################################################################
#                         Bootstrap Chain-Ladder                           #
#################################################################################
```

```r
set.seed(1)

(B <- BootChainLadder(cum.triangle, R=1000, process.distr="od.pois"))
```

```
## BootChainLadder(Triangle = cum.triangle, R = 1000, process.distr = "od.pois")
##
##      Latest Mean Ultimate Mean IBNR IBNR.S.E IBNR 75% IBNR 95%
## 1988 27,584        27,584         0        0        0        0
## 1989 25,357        28,227     2,870      217    3,011    3,216
## 1990 31,777        39,896     8,119      389    8,393    8,730
## 1991 33,799        48,569    14,770      551   15,170   15,648
## 1992 32,507        54,583    22,076      716   22,559   23,210
## 1993 33,988        68,466    34,478      956   35,082   36,033
## 1994 34,972        87,821    52,849    1,328   53,715   54,963
## 1995 28,400        95,061    66,661    1,685   67,860   69,398
## 1996 20,039       100,262    80,223    2,108   81,636   83,516
## 1997 13,768       136,686   122,918    3,710  125,483  129,038
##
##                  Totals
## Latest:         282,191
## Mean Ultimate:  687,155
## Mean IBNR:      404,964
## IBNR.S.E          7,366
## Total IBNR 75%: 410,307
## Total IBNR 95%: 416,818
```

IBNR 75% is the quantile which is useful in some line of business due to capital requirement purpouses. In the 75% of the cases you'll have that IBNR and so you must set aside that amount of value. The same meaning is for IBNR 95%.

```r
plot(B)
```




```r
quantile(B, c(0.75,0.95,0.99, 0.995))
```

```
## $ByOrigin
##        IBNR 75%  IBNR 95%  IBNR 99% IBNR 99.5%
## 1988       0.00      0.00      0.00       0.00
## 1989    3011.00   3216.05   3358.11    3418.26
## 1990    8393.00   8730.15   8947.05    9035.02
## 1991   15170.25  15647.65  16037.24   16130.21
## 1992   22558.50  23210.15  23654.06   23871.65
## 1993   35082.25  36033.00  36803.07   37106.50
## 1994   53715.00  54962.50  56216.16   56396.76
## 1995   67859.75  69397.80  70657.85   70908.18
## 1996   81635.50  83516.05  85015.75   85355.81
## 1997  125482.50 129037.55 130787.56  131834.02
##
## $Totals
##                 Totals
## IBNR 75%:     410307.0
## IBNR 95%:     416817.8
## IBNR 99%:     421419.0
## IBNR 99.5%: 423251.8
```

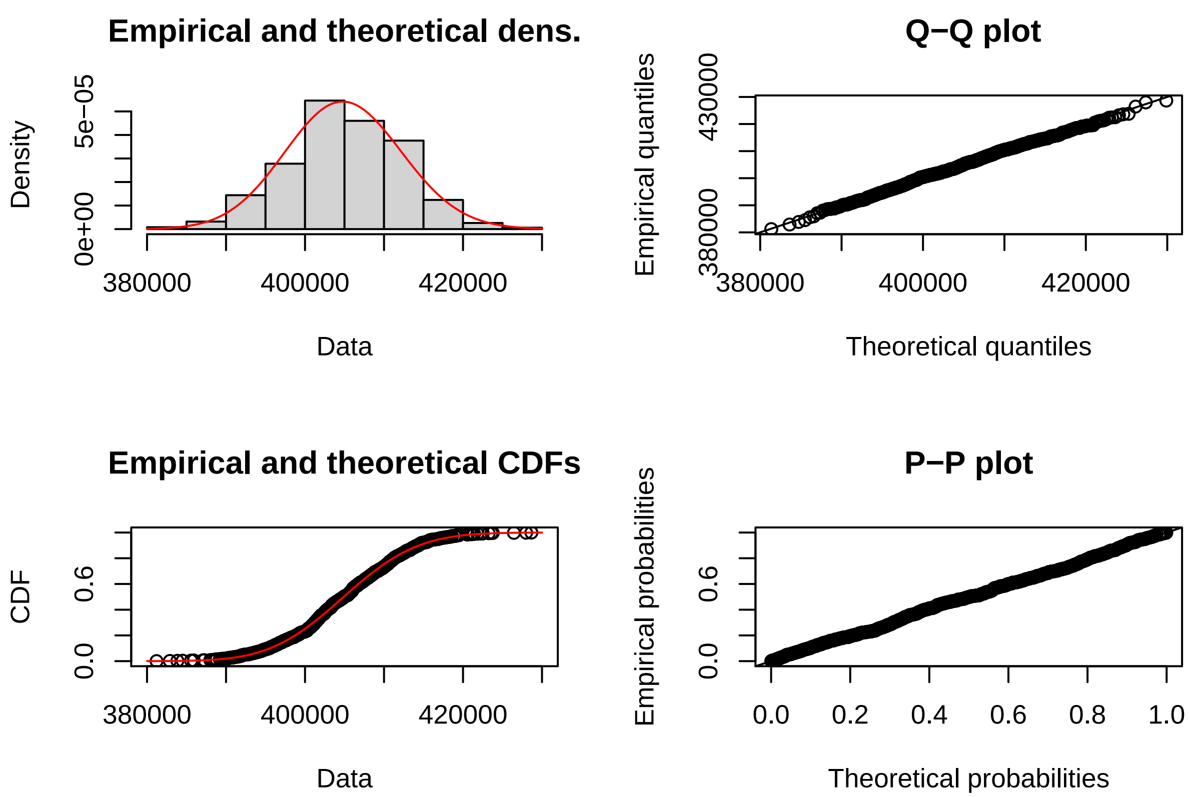
Losses may follow a lognormal distribution. We can test this idea for our data by fitting a log-normal distribution to the predicted future payments. The fitdistrplus package by Delignette-Muller et al. (2013) makes it a one line in R:

```r
library(fitdistrplus)

(fit <- fitdist(B$IBNR.Totals[B$IBNR.Totals>0], "lnorm"))
```

```
## Fitting of the distribution ' lnorm ' by maximum likelihood
## Parameters:
##            estimate   Std. Error
## meanlog 12.91138705 0.0005753809
## sdlog    0.01819514 0.0004013392
```

Plots of simulated data from BootChainLadder and a fitted log-normal distribution:

```r
plot(fit)
```




```r
qlnorm(0.995, fit$estimate["meanlog"], fit$estimate["sdlog"])
```

```
## [1] 424324.9
```

```r
ny <- (col(inc.triangle) == (nrow(inc.triangle) - row(inc.triangle) + 2))
paid.ny <- apply(B$IBNR.Triangles, 3,
                 function(x){
                   next.year.paid <- x[col(x) == (nrow(x) - row(x) + 2)]
                   sum(next.year.paid)
                 })
paid.ny.995 <- B$IBNR.Triangles[,,order(paid.ny)[round(B$R*0.995)]]
inc.triangle.ny <- inc.triangle
(inc.triangle.ny[ny] <- paid.ny.995[ny])
```

```
## [1] 14776 10003  9485  9218  7156  5351  5126  4242  2873
```

The fit looks very reasonable.

Munich chain-ladder

It's a reserving method that reduces the gap between IBNR projections based on paid losses and IBNR projections based on incurred losses. The Munich chain-ladder method uses correlations between paid and incurred losses of the historical data into the projection for the future. I'll report the codes that I've run on R:

```
Incurred= cum.triangle
n<-10
Claimspaid <- data.frame(originf = factor(rep(1988:1997, n:1)),
                         dev=sequence(n:1),
                         cum.paid=
                           c(1586,2307,2604,2702,2745,2809,2806,2803,2800,2797,
                             1190,2044,2641,2756,2796,2818,2871,2872,2873,
                             2007,3281,3909,3955,3963,3982,3983,3984,
                             2300,4150,4631,4762,4782,4804,4804,
                             2685,4473,4982,5136,5203,5232,
                             3212,5409,6102,6439,6526,
                             3881,6833,7957,8357,
                             4804,7373,8486,
                             4664,7369,
                             5915))

(cumpaid.triangle <- with(Claimspaid, {
  M <- matrix(nrow=n, ncol=n,
              dimnames=list(origin=levels(originf), dev=1:n))
  M[cbind(originf, dev)] <- cum.paid
  M
}))
```

```
##       dev
## origin    1    2    3    4    5    6    7    8    9   10
##   1988 1586 2307 2604 2702 2745 2809 2806 2803 2800 2797
##   1989 1190 2044 2641 2756 2796 2818 2871 2872 2873   NA
##   1990 2007 3281 3909 3955 3963 3982 3983 3984   NA   NA
##   1991 2300 4150 4631 4762 4782 4804 4804   NA   NA   NA
##   1992 2685 4473 4982 5136 5203 5232   NA   NA   NA   NA
##   1993 3212 5409 6102 6439 6526   NA   NA   NA   NA   NA
##   1994 3881 6833 7957 8357   NA   NA   NA   NA   NA   NA
##   1995 4804 7373 8486   NA   NA   NA   NA   NA   NA   NA
##   1996 4664 7369   NA   NA   NA   NA   NA   NA   NA   NA
##   1997 5915   NA   NA   NA   NA   NA   NA   NA   NA   NA
```

```
(cum.paid.triangle <- t(apply(cumpaid.triangle, 1, cumsum)))
```

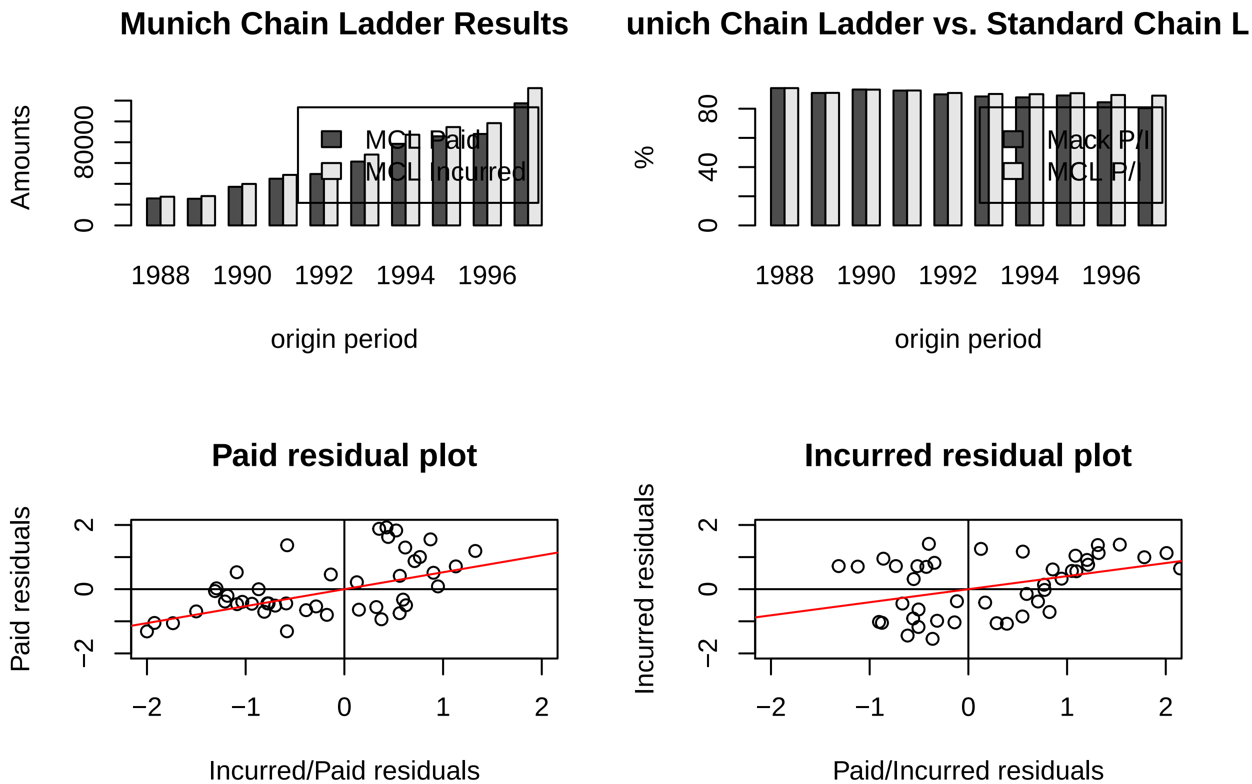
```
##       dev
## origin    1     2     3     4     5     6     7     8     9    10
##   1988 1586  3893  6497  9199 11944 14753 17559 20362 23162 25959
##   1989 1190  3234  5875  8631 11427 14245 17116 19988 22861    NA
##   1990 2007  5288  9197 13152 17115 21097 25080 29064    NA    NA
##   1991 2300  6450 11081 15843 20625 25429 30233    NA    NA    NA
##   1992 2685  7158 12140 17276 22479 27711    NA    NA    NA    NA
```

```
##   1993 3212  8621 14723 21162 27688    NA    NA    NA    NA    NA
##   1994 3881 10714 18671 27028    NA    NA    NA    NA    NA    NA
##   1995 4804 12177 20663    NA    NA    NA    NA    NA    NA    NA
##   1996 4664 12033    NA    NA    NA    NA    NA    NA    NA    NA
##   1997 5915    NA    NA    NA    NA    NA    NA    NA    NA    NA
```

```r
Paid=cum.paid.triangle

MCL=MunichChainLadder(Paid=Paid ,Incurred= Incurred,
                      est.sigmaP = "log-linear", est.sigmaI = "log-linear",
                      tailP=FALSE, tailI=FALSE)
MCL
```

```
## MunichChainLadder(Paid = Paid, Incurred = Incurred, est.sigmaP = "log-linear",
##     est.sigmaI = "log-linear", tailP = FALSE, tailI = FALSE)
##
##      Latest Paid Latest Incurred Latest P/I Ratio Ult. Paid Ult. Incurred
## 1988      25,959          27,584            0.941    25,959        27,584
## 1989      22,861          25,357            0.902    25,640        28,224
## 1990      29,064          31,777            0.915    37,127        39,888
## 1991      30,233          33,799            0.894    44,918        48,574
## 1992      27,711          32,507            0.852    49,454        54,476
## 1993      27,688          33,988            0.815    61,380        68,131
## 1994      27,028          34,972            0.773    78,450        87,219
## 1995      20,663          28,400            0.728    85,713        94,567
## 1996      12,033          20,039            0.600    87,935        98,376
## 1997       5,915          13,768            0.430   117,421       132,024
##      Ult. P/I Ratio
## 1988          0.941
## 1989          0.908
## 1990          0.931
## 1991          0.925
## 1992          0.908
## 1993          0.901
## 1994          0.899
## 1995          0.906
## 1996          0.894
## 1997          0.889
##
## Totals
##              Paid Incurred P/I Ratio
## Latest:   229,155  282,191      0.81
## Ultimate: 613,997  679,064      0.90
```

```r
plot(MCL)
```



Recalling that Ultimate loss cost = paid + reserve = paid + case reserve + IBNR = incurred + IBNR:

- Mack ChainLadder -> Ultimate= 687,282.96
- Munich ChainLadder -> Ultimate= 679,064
- Bootstrap ChainLadder -> (Mean) Ultimate= 687,267

As a final comment between ChainLadder-based models, we can say that the Munich ChainLadder underestimated the value of the Ultimate loss cost w.r.to the other 2 models that I've taken into account. In terms of Reserving:

- Basic ChainLadder-> Reserve= 405092
- Mack ChainLadder -> Reserve= 405,091.96
- Bootstrap ChainLadder -> Reserve= 405,076
- Munich ChainLadder -> Reserve= 396,873