Abstract

The ChainLadder package provides various statistical methods which are typically used for the estimation of outstanding claims reserves in general insurance, including those to estimate the claims development results as required under Solvency II.
# Contents

1 Introduction 4  
   1.1 Claims reserving in insurance 4  
2 The ChainLadder package 5  
   2.1 Motivation 5  
   2.2 Brief package overview 5  
   2.3 Installation 6  
3 Using the ChainLadder package 6  
   3.1 Working with triangles 6  
      3.1.1 Plotting triangles 8  
      3.1.2 Transforming triangles between cumulative and incremental representation 9  
      3.1.3 Importing triangles from external data sources 10  
      3.1.4 Creating triangles interactively 14  
4 Chain-ladder methods 14  
   4.1 Basic idea 15  
   4.2 Mack chain-ladder 19  
      4.2.1 Using a subset of the triangle 23  
   4.3 Munich chain-ladder 24  
   4.4 Bootstrap chain-ladder 26  
   4.5 Multivariate chain-ladder 29  
   4.6 The "triangles" class 30  
   4.7 Separate chain-ladder ignoring correlations 31  
   4.8 Multivariate chain-ladder using seemingly unrelated regressions 33  
   4.9 Other residual covariance estimation methods 34  
   4.10 Model with intercepts 37  
   4.11 Joint modelling of the paid and incurred losses 39  
5 Clark’s methods 40  
   5.1 Clark’s LDF method 41
1 Introduction

1.1 Claims reserving in insurance

The insurance industry, unlike other industries, does not sell products as such but promises. An insurance policy is a promise by the insurer to the policyholder to pay for future claims for an upfront received premium.

As a result insurers don’t know the upfront cost for their service, but rely on historical data analysis and judgement to predict a sustainable price for their offering. In General Insurance (or Non-Life Insurance, e.g. motor, property and casualty insurance) most policies run for a period of 12 months. However, the claims payment process can take years or even decades. Therefore often not even the delivery date of their product is known to insurers.

In particular losses arising from casualty insurance can take a long time to settle and even when the claims are acknowledged it may take time to establish the extent of the claims settlement cost. Claims can take years to materialize. A complex and costly example are the claims from asbestos liabilities, particularly those in connection with mesothelioma and lung damage arising from prolonged exposure to asbestos. A research report by a working party of the Institute and Faculty of Actuaries estimated that the un-discounted cost of UK mesothelioma-related claims to the UK Insurance Market for the period 2009 to 2050 could be around £10bn, see [GBB+09]. The cost for asbestos related claims in the US for the worldwide insurance industry was estimate to be around $120bn in 2002, see [Mic02].

Thus, it should come as no surprise that 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 in case reserves (or outstanding claims), which are losses already reported to the insurance company and losses that are incurred but not reported (IBNR) yet.

Historically, reserving was based on deterministic calculations with pen and paper, combined with expert judgement. Since the 1980’s, with the arrival of personal computer, spreadsheet software became very popular for reserving. Spreadsheets not only reduced the calculation time, but allowed actuaries to test different scenarios and the sensitivity of their forecasts.

As the computer became more powerful, ideas of more sophisticated models started to evolve. Changes in regulatory requirements, e.g. Solvency II\(^1\) 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.

Over the years several methods and models have been developed to estimate both the level and variability of reserves for insurance claims, see [Sch11] or [PR02] for an overview.

---

\(^1\)See https://ec.europa.eu/internal_market/insurance/solvency/index_en.htm
In practice the Mack chain-ladder and bootstrap chain-ladder models are used by many actuaries along with stress testing / scenario analysis and expert judgement to estimate ranges of reasonable outcomes, see the surveys of UK actuaries in 2002, \([LFK^{+}02]\), and across the Lloyd’s market in 2012, \([Orr12]\).

2 The ChainLadder package

2.1 Motivation

The ChainLadder \([GMZ^{+}20]\) package provides various statistical methods which are typically used for the estimation of outstanding claims reserves in general insurance. The package started out of presentations given by Markus Gesmann at the Stochastic Reserving Seminar at the Institute of Actuaries in 2007 and 2008, followed by talks at Casualty Actuarial Society (CAS) meetings joined by Dan Murphy in 2008 and Wayne Zhang in 2010.

Implementing reserving methods in R has several advantages. R provides:

- a rich language for statistical modelling and data manipulations allowing fast prototyping
- a very active user base, which publishes many extensions
- many interfaces to data bases and other applications, such as MS Excel
- an established framework for End User Computing, including documentation, testing and workflows with version control systems
- code written in plain text files, allowing effective knowledge transfer
- an effective way to collaborate over the internet
- built in functions to create reproducible research reports\(^2\)
- in combination with other tools such as \(\text{\LaTeX}\) and Sweave or Markdown easy to set up automated reporting facilities
- access to academic research, which is often first implemented in R

2.2 Brief package overview

This vignette will give the reader a brief overview of the functionality of the ChainLadder package. The functions are discussed and explained in more detail in the respective help files and examples, see also \([Ges14]\).

A set of demos is shipped with the packages and the list of demos is available via:

\(^2\)For an example see the project: Formatted Actuarial Vignettes in R, \(https://github.com/cran/favir\)
2.3 Installation

You can install ChainLadder in the usual way from CRAN, e.g.:

R> install.packages("ChainLadder")

For more details about installing packages see [Tea12b]. The installation was successful if the command library(ChainLadder) gives you the following message:

R> library(ChainLadder)

Welcome to ChainLadder version 0.2.11

Type vignette('ChainLadder', package='ChainLadder') to access the overall package documentation.

See demo(package="ChainLadder") for a list of demos.

More information is available on the ChainLadder project web-site: https://github.com/mages/ChainLadder

To suppress this message use:
suppressPackageStartupMessages(library(ChainLadder))

3 Using the ChainLadder package

3.1 Working with triangles

Historical insurance data is often presented in form of a triangle structure, showing the development of claims over time for each exposure (origin) period. An origin period could be the year the policy was written or earned, or the loss occurrence period. Of course the origin period doesn't have to be yearly, e.g. quarterly or
monthly origin periods are also often used. The development period of an origin period is also called age or lag. Data on the diagonals present payments in the same calendar period. Note, data of individual policies is usually aggregated to homogeneous lines of business, division levels or perils.

Most reserving methods of the ChainLadder package expect triangles as input data sets with development periods along the columns and the origin period in rows. The package comes with several example triangles. The following R command will list them all:

```
R> require(ChainLadder)
R> data(package="ChainLadder")
```

Let’s look at one example triangle more closely. The following triangle shows data from the Reinsurance Association of America (RAA):

```
R> ## Sample triangle
R> RAA
```

This triangle shows the known values of loss from each origin year and of annual evaluations thereafter. For example, the known values of loss originating from the 1988 exposure period are 1351, 6947, and 13112 as of year ends 1988, 1989, and 1990, respectively. The latest diagonal – i.e., the vector 18834, 16704, ... 2063 from the upper right to the lower left – shows the most recent evaluation available. The column headings – 1, 2, ..., 10 – hold the ages (in years) of the observations in the column relative to the beginning of the exposure period. For example, for the 1988 origin year, the age of the 13112 value, evaluated as of 1990-12-31, is three years.

The objective of a reserving exercise is to forecast the future claims development in the bottom right corner of the triangle and potential further developments beyond development age 10. Eventually all claims for a given origin period will be settled, but it is not always obvious to judge how many years or even decades it will take.
We speak of long and short tail business depending on the time it takes to pay all claims.

### 3.1.1 Plotting triangles

The first thing you often want to do is to plot the data to get an overview. For a data set of class `triangle` the ChainLadder package provides default plotting methods to give a graphical overview of the data:

\[ R> \text{plot(RAA)} \]

![Figure 1: Claims development chart of the RAA triangle, with one line per origin period. Output of plot(RAA)](image)

Setting the argument `lattice=TRUE` will produce individual plots for each origin period\(^3\), see Figure 2.

\[ R> \text{plot(RAA, lattice=TRUE)} \]

\(^3\)ChainLadder uses the `lattice` package for plotting the development of the origin years in separate panels.
Figure 2: Claims development chart of the RAA triangle, with individual panels for each origin period. Output of plot(RAA, lattice=TRUE)

You will notice from the plots in Figures 1 and 2 that the triangle RAA presents claims developments for the origin years 1981 to 1990 in a cumulative form. For more information on the triangle plotting functions see the help pages of plot.triangle, e.g. via

R> ?plot.triangle

3.1.2 Transforming triangles between cumulative and incremental representation

The ChainLadder packages comes with two helper functions, cum2incr and incr2cum to transform cumulative triangles into incremental triangles and vice versa:

R> raa.inc <- cum2incr(RAA)
R> ## Show first origin period and its incremental development
R> raa.inc[1,]
R> raa.cum <- incr2cum(raa.inc)
R> ## Show first origin period and its cumulative development
R> raa.cum[1,]

1 2 3 4 5 6 7 8 9 10
5012 8269 10907 11805 13539 16181 18009 18608 18662 18834

3.1.3 Importing triangles from external data sources

In most cases you want to analyse your own data, usually stored in data bases or spreadsheets.

**Importing a triangle from a spreadsheet** There are many ways to import the data from a spreadsheet. A quick and dirty solution is using a CSV-file.

Open a new workbook and copy your triangle into cell A1, with the first column being the accident or origin period and the first row describing the development period or age.

Ensure the triangle has no formatting, such a commas to separate thousands, as those cells will be saved as characters.

![Figure 3: Screen shot of a triangle in a spreadsheet software.](image)

Now open R and go through the following commands:

R> myCSVfile <- "path/to/folder/with/triangle.csv"
R> ## Use the R command:
R> # myCSVfile <- file.choose()
R> ## to select the file interactively
R> tri <- read.csv(file=myCSVfile, header = FALSE)
R> ## Use read.csv2 if semicolons are used as a separator likely
R> ## to be the case if you are in continental Europe
R> library(ChainLadder)
R> ## Convert to triangle
R> tri <- as.triangle(as.matrix(tri))
R> # Job done.

**Copying and pasting from a spreadsheet**  Small data sets can be transferred to R backwards and forwards via the clipboard under MS Windows.

Select a data set in the spreadsheet and copy it into the clipboard, then go to R and type:

R> tri <- read.table(file="clipboard", sep="\t", na.strings="")

**Reading data from a data base**  R makes it easy to access data using SQL statements, e.g. via an ODBC connection, for more details see [Tea12a]. The ChainLadder packages includes a demo to showcase how data can be imported from a MS Access data base, see:

R> demo(DatabaseExamples)

In this section we use data stored in a CSV-file to demonstrate some typical operations you will want to carry out with data stored in data bases. CSV stands for comma separated values, stored in a text file. Note many European countries use a comma as decimal point and a semicolon as field separator, see also the help file to read.csv2. In most cases your triangles will be stored in tables and not in a classical triangle shape. The ChainLadder package contains a CSV-file with sample data in a long table format. We read the data into R's memory with the read.csv command and look at the first couple of rows and summarise it:

R> filename <- file.path(system.file("Database",
                   package="ChainLadder"),
                      "TestData.csv")
R> myData <- read.csv(filename)
R> head(myData)

<table>
<thead>
<tr>
<th>origin</th>
<th>dev</th>
<th>value</th>
<th>lob</th>
</tr>
</thead>
<tbody>
<tr>
<td>1977</td>
<td>1</td>
<td>153638</td>
<td>ABC</td>
</tr>
<tr>
<td>1978</td>
<td>1</td>
<td>178536</td>
<td>ABC</td>
</tr>
</tbody>
</table>

See the RODBC and DBI packages

Please ensure that your CSV-file is free from formatting, e.g. characters to separate units of thousands, as those columns will be read as characters or factors rather than numerical values.
R> summary(myData)

          origin      dev  value      lob
    Min. : 1  Min. : 1.00  Min. : -17657  AutoLiab :105
  1st Qu.: 3  1st Qu.: 2.00  1st Qu.: 10324  GeneralLiab :105
  Median : 6  Median : 4.00  Median :  72468  M3IR5 :105
  Mean : 642  Mean : 4.61  Mean :  176632  ABC :  66
  3rd Qu.:1979  3rd Qu.: 7.00  3rd Qu.: 197716  CommercialAutoPaid: 55
     Max. :1991  Max. :14.00  Max. :3258646  GenIns :  55
                 (Other) :210

Let's focus on one subset of the data. We select the RAA data again:

R> raa <- subset(myData, lob %in% "RAA")

R> head(raa)

       origin dev value  lob
   67  1981 1  5012 RAA
   68  1982 1  106  RAA
   69  1983 1 3410  RAA
   70  1984 1 5655  RAA
   71  1985 1 1092  RAA
   72  1986 1 1513  RAA

To transform the long table of the RAA data into a triangle we use the function as.triangle. The arguments we have to specify are the column names of the origin and development period and further the column which contains the values:

R> raa.tri <- as.triangle(raa, origin="origin", dev="dev", value="value")

R> raa.tri

     dev
    origin 1  2  3  4  5  6  7  8  9 10
1981   5012 3257 2638 898 1734 2642 1828 599  54 172
1982  106  4179 1111 5270 3116 1817  -103 673  535   NA
1983  3410 5582 4881 2268 2594 3479  649  603  NA   NA

12
We note that the data has been stored as an incremental data set. As mentioned above, we could now use the function `incr2cum` to transform the triangle into a cumulative format.

We can transform a triangle back into a data frame structure:

```r
R> raa.df <- as.data.frame(raa.tri, na.rm=TRUE)
R> head(raa.df)
```

```
origin dev value
1981-1 1981 1 5012
1982-1 1982 1 106
1983-1 1983 1 3410
1984-1 1984 1 5655
1985-1 1985 1 1092
1986-1 1986 1 1513
```

This is particularly helpful when you would like to store your results back into a database. Figure 4 gives you an idea of a potential data flow between R and databases.

![Figure 4: Flow chart of data between R and databases.](image-url)
3.1.4 Creating triangles interactively

For small data sets or while testing procedures, it may be useful to create triangles interactively from the command line. There are two main ways to proceed. With the first we create a matrix of data (including missing values in the lower right portion of the triangle) and then convert it into a triangle with \texttt{as.triangle}:

\begin{verbatim}
R> as.triangle(matrix(c(100, 150, 175, 180, 200,
    110, 168, 192, 205, NA,
    115, 169, 202, NA, NA,
    125, 185, NA, NA, NA,
    150, NA, NA, NA, NA),
    nrow = 5, byrow = TRUE))
\end{verbatim}

\begin{verbatim}
dev
origin 1 2 3 4 5
1 100 150 175 180 200
2 110 168 192 205 NA
3 115 169 202 NA NA
4 125 185 NA NA NA
5 150 NA NA NA NA
\end{verbatim}

We may also create the triangle directly with \texttt{triangle} by providing the rows (or columns) of known data as vectors, thereby omitting the missing values:

\begin{verbatim}
R> triangle(c(100, 150, 175, 180, 200),
            c(110, 168, 192, 205),
            c(115, 169, 202),
            c(125, 185),
            150)
\end{verbatim}

\begin{verbatim}
dev
origin 1 2 3 4 5
1 100 150 175 180 200
2 110 168 192 205 NA
3 115 169 202 NA NA
4 125 185 NA NA NA
5 150 NA NA NA NA
\end{verbatim}

4 Chain-ladder methods

The classical chain-ladder is a deterministic algorithm to forecast claims based on historical data. It assumes that the proportional developments of claims from one development period to the next are the same for all origin years.
4.1 Basic idea

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 $C_{ik}, i, k = 1, \ldots, n$.

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

```
R> n <- 10
R> f <- sapply(1:(n-1),
            function(i){
                sum(RAA[c(1:(n-i)),i+1])/sum(RAA[c(1:(n-i)),i])
            })
R> f
[1] 2.999 1.624 1.271 1.172 1.113 1.042 1.033 1.017 1.009
```

Often it is not suitable to assume that the oldest origin year is fully developed. A typical approach is to extrapolate the development ratios, e.g. assuming a log-linear model.

```
R> dev.period <- 1:(n-1)
R> plot(log(f-1) ~ dev.period, main="Log-linear extrapolation of age-to-age factors")
R> tail.model <- lm(log(f-1) ~ dev.period)
R> abline(tail.model)
R> co <- coef(tail.model)
R> ## extrapolate another 100 dev. period
R> tail <- exp(co[1] + c(n:(n + 100)) * co[2]) + 1
R> f.tail <- prod(tail)
R> f.tail
```

[1] 1.009
Log-linear extrapolation of age-to-age factors

The age-to-age factors allow us to plot the expected claims development patterns.

```r
R> plot(100*(rev(1/cumprod(rev(c(f, tail[tail>1.0001]))))), t="b",
         main="Expected claims development pattern",
         xlab="Dev. period", ylab="Development % of ultimate loss")
```

16
The link ratios are then applied to the latest known cumulative claims amount to forecast the next development period. The *squaring* of the RAA triangle is calculated below, where an *ultimate* column is appended to the right to accommodate the expected development beyond the oldest age (10) of the triangle due to the tail factor (1.009) being greater than unity.

```r
R> f <- c(f, f.tail)
R> fullRAA <- cbind(RAA, Ult = rep(0, 10))
R> for(k in 1:n){
R>   fullRAA[(n-k+1):n, k+1] <- fullRAA[(n-k+1):n,k]*f[k]
R> }
R> round(fullRAA)
```

<table>
<thead>
<tr>
<th>Dev. period</th>
<th>Dev. period</th>
<th>Dev. period</th>
<th>Dev. period</th>
<th>Dev. period</th>
<th>Dev. period</th>
<th>Dev. period</th>
<th>Dev. period</th>
<th>Dev. period</th>
<th>Dev. period</th>
<th>Dev. period</th>
<th>Dev. period</th>
</tr>
</thead>
<tbody>
<tr>
<td>1981</td>
<td>5012</td>
<td>8269</td>
<td>10907</td>
<td>11805</td>
<td>13539</td>
<td>16181</td>
<td>18009</td>
<td>18608</td>
<td>18662</td>
<td>18834</td>
<td>19012</td>
</tr>
<tr>
<td>1982</td>
<td>106</td>
<td>4285</td>
<td>5396</td>
<td>10666</td>
<td>13782</td>
<td>15599</td>
<td>15496</td>
<td>16169</td>
<td>16704</td>
<td>16858</td>
<td>17017</td>
</tr>
<tr>
<td>1983</td>
<td>3410</td>
<td>8992</td>
<td>13873</td>
<td>16141</td>
<td>18735</td>
<td>22214</td>
<td>22863</td>
<td>23466</td>
<td>23863</td>
<td>24083</td>
<td>24311</td>
</tr>
<tr>
<td>1984</td>
<td>5655</td>
<td>15555</td>
<td>15766</td>
<td>21266</td>
<td>23425</td>
<td>26083</td>
<td>27067</td>
<td>27967</td>
<td>28441</td>
<td>28703</td>
<td>28974</td>
</tr>
<tr>
<td>1985</td>
<td>1092</td>
<td>9565</td>
<td>15836</td>
<td>22169</td>
<td>25955</td>
<td>26180</td>
<td>27728</td>
<td>28185</td>
<td>28663</td>
<td>28927</td>
<td>29200</td>
</tr>
<tr>
<td>1986</td>
<td>1513</td>
<td>6445</td>
<td>11702</td>
<td>12935</td>
<td>15852</td>
<td>17649</td>
<td>18389</td>
<td>19001</td>
<td>19323</td>
<td>19501</td>
<td>19685</td>
</tr>
<tr>
<td>1987</td>
<td>557</td>
<td>4020</td>
<td>10946</td>
<td>12314</td>
<td>14428</td>
<td>16064</td>
<td>16738</td>
<td>17294</td>
<td>17587</td>
<td>17749</td>
<td>17917</td>
</tr>
<tr>
<td>1988</td>
<td>1351</td>
<td>6947</td>
<td>13112</td>
<td>16664</td>
<td>19525</td>
<td>21738</td>
<td>22650</td>
<td>23403</td>
<td>23800</td>
<td>24019</td>
<td>24246</td>
</tr>
</tbody>
</table>
The total estimated outstanding loss under this method is about 54100:

\[ \text{sum(fullRAA[ ,11] - getLatestCumulative(RAA))} \]

This approach is also called Loss Development Factor (LDF) method.

More generally, the factors used to square the triangle 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, since 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 interior of the triangle is usually not displayed in favor of that multiplicative calculation.

For example, suppose the actuary decides that the volume weighted factors from the RAA triangle are representative of expected future growth, but discards the 1.009 tail factor derived from the loglinear fit in favor of a five percent tail (1.05) based on loss data from a larger book of similar business. The LDF method might be displayed in R as follows.

\[ \text{linkratios <- c(attr(ata(RAA), "vwted"), tail = 1.05)} \]
\[ \text{round(linkratios, 3)} \]

\[ \text{LDF <- rev(cumprod(rev(linkratios)))} \]
\[ \text{names(LDF) <- colnames(RAA)} \]
\[ \text{round(LDF, 3)} \]

\[ \text{currentEval <- getLatestCumulative(RAA)} \]
\[ \text{# Reverse the LDFs so the first, least mature factor [1]} \]
\[ \text{# is applied to the last origin year (1990)} \]
\[ \text{EstdUlt <- currentEval * rev(LDF)} \]
\[ \text{# Start with the body of the exhibit} \]
\[ \text{Exhibit <- data.frame(currentEval, LDF = round(rev(LDF), 3), EstdUlt)} \]
Since the early 1990s several papers have been published to embed the simple chain-
ladder method into a statistical framework. Ben Zehnwirth and Glenn Barnett point
out in [ZB00] that the age-to-age link ratios can be regarded as the coefficients of
a weighted linear regression through the origin, see also [Mur94].

\[ R> \text{lmCL} <- \text{function}(i, \text{Triangle})\{
\text{lm}(y\sim x+0, \text{weights}=1/\text{Triangle}[,i],
\text{data} = \text{data.frame}(x = \text{Triangle}[,i], y = \text{Triangle}[,i+1]))
\} \]

4.2 Mack chain-ladder

Thomas Mack published in 1993 [Mac93] a method which estimates the standard
erors of the chain-ladder forecast without assuming a distribution under three
conditions.

Following the notation of Mack [Mac99] let \( C_{ik} \) denote the cumulative loss amounts
of origin period (e.g. accident year) \( i = 1, \ldots, m \), with losses known for development
period (e.g. development year) \( k \leq n+1-i \).

In order to forecast the amounts \( C_{ik} \) for \( k > n+1-i \) the Mack chain-ladder-model
assumes:

CL1: \( E[F_{ik}|C_{i1}, C_{i2}, \ldots, C_{ik}] = f_k \) with \( F_{ik} = \frac{C_{i,k+1}}{C_{ik}} \) \hspace{1cm} (2)

CL2: \( Var\left(\frac{C_{i,k+1}}{C_{ik}}|C_{i1}, C_{i2}, \ldots, C_{ik}\right) = \frac{\sigma^2_k}{w_{ik}C_{ik}^{\alpha}} \) \hspace{1cm} (3)

CL3: \( \{C_{i1}, \ldots, C_{in}\}, \{C_{j1}, \ldots, C_{jn}\} \), are independent for origin period \( i \neq j \) \hspace{1cm} (4)

with \( w_{ik} \in [0; 1], \alpha \in \{0, 1, 2\} \). If these assumptions hold, the Mack chain-ladder model gives an unbiased estimator for IBNR (Incurred But Not Reported) claims.

The Mack chain-ladder model can be regarded as a weighted linear regression through the origin for each development period: \( \text{lm}(y \sim x + 0, \text{weights}=w/x^{(2-\alpha)}) \), where \( y \) is the vector of claims at development period \( k+1 \) and \( x \) is the vector of claims at development period \( k \).

The Mack method is implemented in the ChainLadder package via the function MackChainLadder.

As an example we apply the MackChainLadder function to our triangle RAA:

```r
R> mack <- MackChainLadder(RAA, est.sigma="Mack")
R> mack

MackChainLadder(Triangle = RAA, est.sigma = "Mack")

<table>
<thead>
<tr>
<th>Latest</th>
<th>Dev.To.Date</th>
<th>Ultimate</th>
<th>IBNR</th>
<th>Mack.S.E</th>
<th>CV(IBNR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1981</td>
<td>18,834</td>
<td>1.000</td>
<td>18,834</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1982</td>
<td>16,704</td>
<td>0.991</td>
<td>16,858</td>
<td>154</td>
<td>206</td>
</tr>
<tr>
<td>1983</td>
<td>23,466</td>
<td>0.974</td>
<td>24,083</td>
<td>617</td>
<td>623</td>
</tr>
<tr>
<td>1984</td>
<td>27,067</td>
<td>0.943</td>
<td>28,703</td>
<td>1,636</td>
<td>747</td>
</tr>
<tr>
<td>1985</td>
<td>26,180</td>
<td>0.905</td>
<td>28,927</td>
<td>2,747</td>
<td>1,469</td>
</tr>
<tr>
<td>1986</td>
<td>15,852</td>
<td>0.813</td>
<td>19,501</td>
<td>3,649</td>
<td>2,002</td>
</tr>
<tr>
<td>1987</td>
<td>12,314</td>
<td>0.694</td>
<td>17,749</td>
<td>5,435</td>
<td>2,209</td>
</tr>
<tr>
<td>1988</td>
<td>13,112</td>
<td>0.546</td>
<td>24,019</td>
<td>10,907</td>
<td>5,358</td>
</tr>
<tr>
<td>1989</td>
<td>5,395</td>
<td>0.336</td>
<td>16,045</td>
<td>10,650</td>
<td>6,333</td>
</tr>
<tr>
<td>1990</td>
<td>2,063</td>
<td>0.112</td>
<td>18,402</td>
<td>16,339</td>
<td>24,566</td>
</tr>
</tbody>
</table>

Totals

Latest: 160,987.00
Dev: 0.76
Ultimate: 213,122.23
IBNR: 52,135.23
Mack.S.E 26,909.01
CV(IBNR): 0.52

We can access the loss development factors and the full triangle via
To check that Mack’s assumption are valid review the residual plots, you should see no trends in either of them.

R> plot(mack)
We can plot the development, including the forecast and estimated standard errors by origin period by setting the argument `lattice=TRUE`.

```R
R> plot(mack, lattice=TRUE)
```
4.2.1 Using a subset of the triangle

The weights argument allows for the selection of a subset of the triangle for the projections.

For example, in order to use only the last 5 calendar years of the triangle, set the weights as follows:

\[
R> \text{calPeriods} <- (\text{row(RAA)} + \text{col(RAA)} - 1)
\]

\[
R> (\text{weights} <- \text{ifelse}(\text{calPeriods} <= 5, 0, \text{ifelse}(\text{calPeriods} > 10, NA, 1)))
\]

```
[1,]  0  0  0  0  0  1  1  1  1  1
[2,]  0  0  0  0  1  1  1  1  1  NA
[3,]  0  0  0  1  1  1  1  1  NA  NA
[4,]  0  0  1  1  1  1  1  NA  NA  NA
[5,]  0  1  1  1  1  1  NA  NA  NA  NA
[6,]  1  1  1  1  1  NA  NA  NA  NA  NA
[7,]  1  1  1  1  NA  NA  NA  NA  NA  NA
[8,]  1  1  1  NA  NA  NA  NA  NA  NA  NA
```
R> MackChainLadder(RAA, weights=weights, est.sigma = "Mack")

MackChainLadder(Triangle = RAA, weights = weights, est.sigma = "Mack")

<table>
<thead>
<tr>
<th>Year</th>
<th>Latest</th>
<th>Dev.To.Date</th>
<th>Ultimate</th>
<th>IBNR</th>
<th>Mack.S.E</th>
<th>CV(IBNR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1981</td>
<td>18,834</td>
<td>1.0000</td>
<td>18,834</td>
<td>0</td>
<td>0</td>
<td>NaN</td>
</tr>
<tr>
<td>1982</td>
<td>16,704</td>
<td>0.9909</td>
<td>16,858</td>
<td>154</td>
<td>206</td>
<td>1.339</td>
</tr>
<tr>
<td>1983</td>
<td>23,466</td>
<td>0.9744</td>
<td>24,083</td>
<td>617</td>
<td>623</td>
<td>1.010</td>
</tr>
<tr>
<td>1984</td>
<td>27,067</td>
<td>0.9430</td>
<td>28,703</td>
<td>1,636</td>
<td>747</td>
<td>0.457</td>
</tr>
<tr>
<td>1985</td>
<td>26,180</td>
<td>0.9050</td>
<td>28,927</td>
<td>2,747</td>
<td>1,469</td>
<td>0.535</td>
</tr>
<tr>
<td>1986</td>
<td>15,852</td>
<td>0.8229</td>
<td>19,264</td>
<td>3,412</td>
<td>2,039</td>
<td>0.598</td>
</tr>
<tr>
<td>1987</td>
<td>12,314</td>
<td>0.7106</td>
<td>17,329</td>
<td>5,015</td>
<td>2,144</td>
<td>0.428</td>
</tr>
<tr>
<td>1988</td>
<td>13,112</td>
<td>0.5613</td>
<td>23,361</td>
<td>10,249</td>
<td>4,043</td>
<td>0.395</td>
</tr>
<tr>
<td>1989</td>
<td>5,395</td>
<td>0.2935</td>
<td>18,384</td>
<td>12,989</td>
<td>5,931</td>
<td>0.457</td>
</tr>
<tr>
<td>1990</td>
<td>2,063</td>
<td>0.0843</td>
<td>24,463</td>
<td>22,400</td>
<td>16,779</td>
<td>0.749</td>
</tr>
</tbody>
</table>

Totals
Latest: 160,987.00
Dev: 0.73
Ultimate: 220,207.63
IBNR: 59,220.63
Mack.S.E 19,859.00
CV(IBNR): 0.34

### 4.3 Munich chain-ladder

Munich chain-ladder is 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. [QM04].

R> MCLpaid

```r
dev
origin 1 2 3 4 5 6 7
1 576 1804 1970 2024 2074 2102 2131
2 866 1948 2162 2232 2284 2348 NA
3 1412 3758 4252 4416 4494 NA NA
4 2286 5292 5724 5850 NA NA NA
5 1868 3778 4648 NA NA NA NA
6 1442 4010 NA NA NA NA NA
7 2044 NA NA NA NA NA NA
```
```r
R> MCLincurred

dev
origin 1 2 3 4 5 6 7
 1 978 2104 2134 2144 2174 2182 2174
 2 1844 2552 2466 2508 2454 NA NA
 3 2904 4354 4698 4600 4644 NA NA
 4 3502 5958 6070 6142 NA NA NA
 5 2812 4882 4852 NA NA NA NA
 6 2642 4406 NA NA NA NA NA
 7 5022 NA NA NA NA NA NA

R> op <- par(mfrow=c(1,2))
R> plot(MCLpaid)
R> plot(MCLincurred)
R> par(op)
R> # Following the example in Quarg's (2004) paper:
R> MCL <- MunichChainLadder(MCLpaid, MCLincurred, est.sigmaP=0.1, est.sigmaI=0.1)
R> MCL

MunichChainLadder(Paid = MCLpaid, Incurred = MCLincurred, est.sigmaP = 0.1,
est.sigmaI = 0.1)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2,131</td>
<td>2,174</td>
<td>0.980</td>
<td>2,131</td>
<td>2,174</td>
</tr>
<tr>
<td>2 2,348</td>
<td>2,454</td>
<td>0.957</td>
<td>2,383</td>
<td>2,444</td>
</tr>
<tr>
<td>3 4,494</td>
<td>4,644</td>
<td>0.968</td>
<td>4,597</td>
<td>4,629</td>
</tr>
<tr>
<td>4 5,850</td>
<td>6,142</td>
<td>0.952</td>
<td>6,119</td>
<td>6,176</td>
</tr>
<tr>
<td>5 4,648</td>
<td>4,852</td>
<td>0.958</td>
<td>4,937</td>
<td>4,950</td>
</tr>
<tr>
<td>6 4,010</td>
<td>4,406</td>
<td>0.910</td>
<td>4,656</td>
<td>4,665</td>
</tr>
<tr>
<td>7 2,044</td>
<td>5,022</td>
<td>0.407</td>
<td>7,549</td>
<td>7,650</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ult. P/I Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1   0.980</td>
</tr>
<tr>
<td>2   0.975</td>
</tr>
<tr>
<td>3   0.993</td>
</tr>
<tr>
<td>4   0.991</td>
</tr>
<tr>
<td>5   0.997</td>
</tr>
<tr>
<td>6   0.998</td>
</tr>
<tr>
<td>7   0.987</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Totals</th>
<th>Paid</th>
<th>Incurred</th>
<th>P/I Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latest:</td>
<td>25,525</td>
<td>29,694</td>
<td>0.86</td>
</tr>
<tr>
<td>Ultimate:</td>
<td>32,371</td>
<td>32,688</td>
<td>0.99</td>
</tr>
</tbody>
</table>

R> plot(MCL)
```
4.4 Bootstrap chain-ladder

The BootChainLadder function uses a two-stage bootstrapping/simulation approach following the paper by England and Verrall [PR02]. In the first stage an ordinary chain-ladder methods is applied to the cumulative claims triangle. From this we calculate the scaled Pearson residuals which we bootstrap R times to forecast future incremental claims payments via the standard chain-ladder method. In the second stage we simulate the process error with the bootstrap value as the mean and using the process distribution assumed. 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.

R> ## See also the example in section 8 of England & Verrall (2002)
R> ## on page 55.
R> B <- BootChainLadder(RAA, R=999, process.distr="gamma")
R> B

BootChainLadder(Triangle = RAA, R = 999, process.distr = "gamma")

<table>
<thead>
<tr>
<th>Year</th>
<th>Latest</th>
<th>Mean</th>
<th>Ultimate</th>
<th>Mean</th>
<th>IBNR</th>
<th>IBNR.S.E</th>
<th>IBNR 75%</th>
<th>IBNR 95%</th>
</tr>
</thead>
<tbody>
<tr>
<td>1981</td>
<td>18,834</td>
<td>18,834</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1982</td>
<td>16,704</td>
<td>16,848</td>
<td>144</td>
<td>640</td>
<td>192</td>
<td>1,200</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1983</td>
<td>23,466</td>
<td>24,128</td>
<td>662</td>
<td>1,247</td>
<td>1,159</td>
<td>2,955</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1984</td>
<td>27,067</td>
<td>28,863</td>
<td>1,796</td>
<td>1,913</td>
<td>2,828</td>
<td>5,262</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1985</td>
<td>26,180</td>
<td>29,035</td>
<td>2,855</td>
<td>2,372</td>
<td>4,157</td>
<td>7,229</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1986 15,852 19,568 3,716 2,464 5,067 8,124
1987 12,314 17,980 5,666 3,177 7,498 11,611
1988 13,112 24,400 11,288 4,961 14,047 20,135
1989 5,395 16,458 11,063 6,202 14,639 22,355
1990 2,063 18,994 16,931 13,817 25,228 42,063

Totals
Latest: 160,987
Mean Ultimate: 215,108
Mean IBNR: 54,121
IBNR.S.E 18,936
Total IBNR 75%: 65,739
Total IBNR 95%: 89,010

R> plot(B)

Quantiles of the bootstrap IBNR can be calculated via the quantile function:

R> quantile(B, c(0.75, 0.95, 0.99, 0.995))

$ByOrigin
   IBNR 75% IBNR 95% IBNR 99% IBNR 99.5%
1981  0.0   0   0   0
1982  192.5 1200 2393 3293
1983 1158.7 2955 4645 5557
1984 2828.0 5262 7439 8518
1985 4157.2 7229 9999 11509
1986 5066.9 8124 11229 12160
1987 7498.4 11611 15021 15973
1988 14046.9 20135 25508 28374
1989 14638.9 22355 29091 32227
1990 25228.5 42063 59507 61531

$Totals
Totals
IBNR 75%:  65739
IBNR 95%:  89010
IBNR 99%:  104189
IBNR 99.5%: 115657

The distribution of the IBNR appears to follow a log-normal distribution, so let’s fit it:

```R
## fit a distribution to the IBNR
library(MASS)
plot(ecdf(B$IBNR.Totals))
## fit a log-normal distribution
fit <- fitdistr(B$IBNR.Totals[B$IBNR.Totals>0], "lognormal") fit

  meanlog     sdlog
10.835355 0.366464
(0.011594) (0.008198)

curve(plnorm(x,fit$estimate["meanlog"], fit$estimate["sdlog"]),
col="red", add=TRUE)
```
4.5 Multivariate chain-ladder

The Mack chain-ladder technique can be generalized to the multivariate setting where multiple reserving triangles are modelled and developed simultaneously. The advantage of the multivariate modelling is that correlations among different triangles can be modelled, which will lead to more accurate uncertainty assessments. Reserving methods that explicitly model the between-triangle contemporaneous correlations can be found in [PS05, MW08b]. Another benefit of multivariate loss reserving is that structural relationships between triangles can also be reflected, where the development of one triangle depends on past losses from other triangles. For example, there is generally need for the joint development of the paid and incurred losses [QM04]. Most of the chain-ladder-based multivariate reserving models can be summarised as sequential seemingly unrelated regressions [Zha10]. We note another strand of multivariate loss reserving builds a hierarchical structure into the model to allow estimation of one triangle to “borrow strength” from other triangles, reflecting the core insight of actuarial credibility [ZDG12].

Denote $Y_{i,k} = (Y_{i,k}^{(1)}, \cdots, Y_{i,k}^{(N)})$ as an $N \times 1$ vector of cumulative losses at accident year $i$ and development year $k$ where $(n)$ refers to the $n$-th triangle. [Zha10] specifies
the model in development period $k$ as:

$$Y_{i,k+1} = A_k + B_k \cdot Y_{i,k} + \epsilon_{i,k}, \quad (5)$$

where $A_k$ is a column of intercepts and $B_k$ is the development matrix for development period $k$. Assumptions for this model are:

$$E(\epsilon_{i,k}|Y_{i,1}, \ldots, Y_{i,t+1-k}) = 0. \quad (6)$$

$$cov(\epsilon_{i,k}|Y_{i,1}, \ldots, Y_{i,t+1-k}) = D(Y_{i,k}^{-\delta/2}) \Sigma_k D(Y_{i,k}^{-\delta/2}). \quad (7)$$

losses of different accident years are independent. \quad (8)

$\epsilon_{i,k}$ are symmetrically distributed. \quad (9)

In the above, $D$ is the diagonal operator, and $\delta$ is a known positive value that controls how the variance depends on the mean (as weights). This model is referred to as the general multivariate chain ladder [GMCL] in [Zha10]. An important special case where $A_k = 0$ and $B_k$'s are diagonal is a naive generalization of the chain-ladder, often referred to as the multivariate chain-ladder [MCL] [PS05].

In the following, we first introduce the class "triangles", for which we have defined several utility functions. Indeed, any input triangles to the MultiChainLadder function will be converted to "triangles" internally. We then present loss reserving methods based on the MCL and GMCL models in turn.

### 4.6 The "triangles" class

Consider the two liability loss triangles from [MW08b]. It comes as a list of two matrices:

```r
R> str(liab)
List of 2
$ GeneralLiab: num [1:14, 1:14] 59966 49685 51914 84937 98921 ...
$ AutoLiab : num [1:14, 1:14] 114423 152296 144325 145904 170333 ...
```

We can convert a list to a "triangles" object using

```r
R> liab2 <- as(liab, "triangles")
R> class(liab2)
[1] "triangles"
attr(,"package")
[1] "ChainLadder"
```

We can find out what methods are available for this class:
For example, if we want to extract the last three columns of each triangle, we can use the "[" operator as follows:

```r
R> liab2[, 12:14, drop = TRUE]
```

An object of class "triangles"

[[1]]

[,1] [,2] [,3]
[1,] 540873 547696 549589
[2,] 563571 562795 NA
[3,] 602710 NA NA

[[2]]

[,1] [,2] [,3]
[1,] 391328 391537 391428
[2,] 485138 483974 NA
[3,] 540742 NA NA

The following combines two columns of the triangles to form a new matrix:

```r
R> cbind2(liab2[1:3, 12])
```

```
[,1] [,2]
[1,] 540873 391328
[2,] 563571 485138
[3,] 602710 540742
```

### 4.7 Separate chain-ladder ignoring correlations

The form of regression models used in estimating the development parameters is controlled by the `fit.method` argument. If we specify `fit.method = "OLS"`, the ordinary least squares will be used and the estimation of development factors for each triangle is independent of the others. In this case, the residual covariance matrix $\Sigma_k$ is diagonal. As a result, the multivariate model is equivalent to running multiple Mack chain-ladders separately.

```r
R> fit1 <- MultiChainLadder(liab, fit.method = "OLS")
R> lapply(summary(fit1)$report.summary, "[", 15, )
```

```
$`Summary Statistics for Triangle 1`
    Latest Dev.To.Date Ultimate   IBNR    S.E   CV
```
Total 11343397 0.6482 17498658 6155261 427289 0.0694

Summary Statistics for Triangle 2
Latest Dev.To.Date Ultimate IBNR S.E CV
Total 8759806 0.8093 10823418 2063612 162872 0.0789

Summary Statistics for Triangle 1+2
Latest Dev.To.Date Ultimate IBNR S.E CV
Total 20103203 0.7098 28322077 8218874 457278 0.0556

In the above, we only show the total reserve estimate for each triangle to reduce the output. The full summary including the estimate for each year can be retrieved using the usual summary function. By default, the summary function produces reserve statistics for all individual triangles, as well as for the portfolio that is assumed to be the sum of the two triangles. This behaviour can be changed by supplying the portfolio argument. See the documentation for details.

We can verify if this is indeed the same as the univariate Mack chain ladder. For example, we can apply the MackChainLadder function to each triangle:

```R
fit <- lapply(liab, MackChainLadder, est.sigma = "Mack")
R> # the same as the first triangle above
R> lapply(fit, function(x) t(summary(x)$Totals))

$GeneralLiab
Totals 11343397 0.6482 17498658 6155261 427289 0.06942

$AutoLiab
Totals 8759806 0.8093 10823418 2063612 162872 0.07893

The argument mse.method controls how the mean square errors are computed. By default, it implements the Mack method. An alternative method is the conditional re-sampling approach in [BBMW06], which assumes the estimated parameters are independent. This is used when mse.method = "Independence". For example, the following reproduces the result in [BBMW06]. Note that the first argument must be a list, even though only one triangle is used.

```R
(B1 <- MultiChainLadder(list(GenIns), fit.method = "OLS",
 mse.method = "Independence"))

$Summary Statistics for Input Triangle
Latest Dev.To.Date Ultimate IBNR S.E CV
1 3901463 1.0000 3901463 0 0 0.000

32
### 4.8 Multivariate chain-ladder using seemingly unrelated regressions

To allow correlations to be incorporated, we employ the seemingly unrelated regressions (see the package `systemfit`) that simultaneously model the two triangles in each development period. This is invoked when we specify `fit.method = "SUR"`:

```r
R> fit2 <- MultiChainLadder(liab, fit.method = "SUR")
R> lapply(summary(fit2)$report.summary, "+", 15)

$\text{Summary Statistics for Triangle 1}$

<table>
<thead>
<tr>
<th>Latest Dev.To.Date</th>
<th>Ultimate</th>
<th>IBNR</th>
<th>S.E</th>
<th>CV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>11343397</td>
<td>0.648</td>
<td>6151510</td>
<td>0.0682</td>
</tr>
</tbody>
</table>

$\text{Summary Statistics for Triangle 2}$

<table>
<thead>
<tr>
<th>Latest Dev.To.Date</th>
<th>Ultimate</th>
<th>IBNR</th>
<th>S.E</th>
<th>CV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>8759806</td>
<td>0.8095</td>
<td>162464</td>
<td>0.0788</td>
</tr>
</tbody>
</table>

$\text{Summary Statistics for Triangle 1+2}$

<table>
<thead>
<tr>
<th>Latest Dev.To.Date</th>
<th>Ultimate</th>
<th>IBNR</th>
<th>S.E</th>
<th>CV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>20103203</td>
<td>0.71</td>
<td>500607</td>
<td>0.061</td>
</tr>
</tbody>
</table>

We see that the portfolio prediction error is inflated to 500,607 from 457,278 in the separate development model ("OLS"). This is because of the positive correlation between the two triangles. The estimated correlation for each development period can be retrieved through the `residCor` function:

```r
R> round(unlist(residCor(fit2)), 3)
```

```
[1]  0.247  0.495  0.682  0.446  0.487  0.451  0.172  0.805  0.337  0.688
[11]  0.004  1.000  0.021
```

Similarly, most methods that work for linear models such as `coef`, `fitted`, `resid` and so on will also work. Since we have a sequence of models, the retrieved results
from these methods are stored in a list. For example, we can retrieve the estimated
development factors for each period as

\[
R> \text{do.call("rbind", coef(fit2))}
\]

\[
\begin{array}{ll}
\text{eq1_x[1]} & \text{eq2_x[2]} \\
[1,] & 3.227 2.2224 \\
[2,] & 1.719 1.2688 \\
[3,] & 1.352 1.1200 \\
[4,] & 1.179 1.0665 \\
[5,] & 1.106 1.0356 \\
[6,] & 1.055 1.0168 \\
[7,] & 1.026 1.0097 \\
[8,] & 1.015 1.0002 \\
[9,] & 1.012 1.0038 \\
[10,] & 1.006 0.9994 \\
[11,] & 1.005 0.9939 \\
[12,] & 1.005 0.9989 \\
[13,] & 1.003 0.9997 \\
\end{array}
\]

The smaller-than-one development factors after the 10-th period for the second
triangle indeed result in negative IBNR estimates for the first several accident years
in that triangle.

The package also offers the \text{plot} method that produces various summary and di-
agnostic figures:

\[
R> \text{parold} <- \text{par(mfrow = c(4, 2), mar = c(4, 4, 2, 1),}
\]
\text{mgp = c(1.3, 0.3, 0), tck = -0.02})
R> \text{plot(fit2, which.triangle = 1:2, which.plot = 1:4)}
R> \text{par(parold)}
\]

The resulting plots are shown in Figure 5. We use \text{which.triangle} to suppress
the plot for the portfolio, and use \text{which.plot} to select the desired types of plots.
See the documentation for possible values of these two arguments.

\subsection{Other residual covariance estimation methods}

Internally, the \text{MultiChainLadder} calls the \text{systemfit} function to fit the regression
models period by period. When SUR models are specified, there are several ways
to estimate the residual covariance matrix $\Sigma_k$. Available methods are "noDfCor",
"geomean", "max", and "Theil" with the default as "geomean". The method
"Theil" will produce unbiased covariance estimate, but the resulting estimate may
not be positive semi-definite. This is also the estimator used by [MW08b]. However,
this method does not work out of the box for the \text{liab} data, and is perhaps one
Figure 5: Summary and diagnostic plots from a MultiChainLadder object.
of the reasons [MW08b] used extrapolation to get the estimate for the last several periods.

Indeed, for most applications, we recommend the use of separate chain ladders for the tail periods to stabilize the estimation - there are few data points in the tail and running a multivariate model often produces extremely volatile estimates or even fails. To facilitate such an approach, the package offers the MultiChainLadder2 function, which implements a split-and-join procedure: we split the input data into two parts, specify a multivariate model with rich structures on the first part (with enough data) to reflect the multivariate dependencies, apply separate univariate chain-ladders on the second part, and then join the two models together to produce the final predictions. The splitting is determined by the "last" argument, which specifies how many of the development periods in the tail go into the second part of the split. The type of the model structure to be specified for the first part of the split model in MultiChainLadder2 is controlled by the type argument. It takes one of the following values: "MCL" - the multivariate chain-ladder with diagonal development matrix; "MCL+int" - the multivariate chain-ladder with additional intercepts; "GMCL-int" - the general multivariate chain-ladder without intercepts; and "GMCL" - the full general multivariate chain-ladder with intercepts and non-diagonal development matrix.

For example, the following fits the SUR method to the first part (the first 11 columns) using the unbiased residual covariance estimator in [MW08b], and separate chain-ladders for the rest:

```R
R> require(systemfit)
R> W1 <- MultiChainLadder2(liab, mse.method = "Independence",
control = systemfit.control(methodResidCov = "Theil"))
R> lapply(summary(W1)$report.summary, "[", 15, )
```

```R
$`Summary Statistics for Triangle 1`
 Latest Dev.To.Date Ultimate IBNR S.E CV
Total 11343397 0.6483 17497403 6154006 427041 0.0694

$`Summary Statistics for Triangle 2`
 Latest Dev.To.Date Ultimate IBNR S.E CV
Total 8759806 0.8095 10821034 2061228 162785 0.079

$`Summary Statistics for Triangle 1+2`
 Latest Dev.To.Date Ultimate IBNR S.E CV
Total 20103203 0.7099 28318437 8215234 505376 0.0615

Similarly, the iterative residual covariance estimator in [MW08b] can also be used, in which we use the control parameter maxiter to determine the number of iterations:

```R
R> for (i in 1:5){
    W2 <- MultiChainLadder2(liab, mse.method = "Independence",
```
control = systemfit.control(methodResidCov = "Theil", maxiter = i))
print(format(summary(W2)@report.summary[[3]][15, 4:5],
digits = 6, big.mark = ","))
}

IBNR S.E
Total 8,215,234 505,376
IBNR S.E
Total 8,215,357 505,443
IBNR S.E
Total 8,215,362 505,444
IBNR S.E
Total 8,215,362 505,444
IBNR S.E
Total 8,215,362 505,444

R> lapply(summary(W2)$report.summary, ",[", 15, )

$`Summary Statistics for Triangle 1`
  Latest Dev.To.Date Ultimate IBNR S.E CV
Total 11343397 0.6483 17497526 6154129 427074 0.0694

$`Summary Statistics for Triangle 2`
  Latest Dev.To.Date Ultimate IBNR S.E CV
Total 8759806 0.8095 10821039 2061233 162790 0.079

$`Summary Statistics for Triangle 1+2`
  Latest Dev.To.Date Ultimate IBNR S.E CV
Total 20103203 0.7099 28318565 8215362 505444 0.0615

We see that the covariance estimate converges in three steps. These are very similar to the results in [MW08b], the small difference being a result of the different approaches used in the last three periods.

Also note that in the above two examples, the argument control is not defined in the prototype of the MultiChainLadder. It is an argument that is passed to the systemfit function through the ... mechanism. Users are encouraged to explore how other options available in systemfit can be applied.

4.10 Model with intercepts

Consider the auto triangles from [Zha10]. It includes three automobile insurance triangles: personal auto paid, personal auto incurred, and commercial auto paid.

R> str(auto)
List of 3
$ PersonalAutoPaid : num [1:10, 1:10] 101125 102541 114932 114452 115597 ...
$ PersonalAutoIncurred: num [1:10, 1:10] 325423 323627 358410 405319 434065 ...
$ CommercialAutoPaid : num [1:10, 1:10] 19827 22331 22533 23128 25053 ...

It is a reasonable expectation that these triangles will be correlated. So we run an MCL model on them:

\begin{verbatim}
R> f0 <- MultiChainLadder2(auto, type = "MCL")
R> # show correlation- the last three columns have zero correlation
R> # because separate chain-ladders are used
R> print(do.call(cbind, residCor(f0)), digits = 3)
\end{verbatim}

\begin{verbatim}
(1,2) 0.327 -0.010 0.598 0.711 0.856 0.928 0 0 0
(1,3) 0.870 0.906 0.939 0.261 -0.060 0.911 0 0 0
(2,3) 0.198 -0.321 0.558 0.380 0.358 0.931 0 0 0
\end{verbatim}

However, from the residual plot, the first row in Figure 6, it is evident that the default mean structure in the MCL model is not adequate. Usually this is a common problem with the chain-ladder based models, owing to the missing of intercepts.

We can improve the above model by including intercepts in the SUR fit as follows:

\begin{verbatim}
R> f1 <- MultiChainLadder2(auto, type = "MCL+int")
\end{verbatim}

The corresponding residual plot is shown in the second row in Figure 6. We see that these residuals are randomly scattered around zero and there is no clear pattern compared to the plot from the MCL model.

The default summary computes the portfolio estimates as the sum of all the triangles. This is not desirable because the first two triangles are both from the personal auto line. We can overwrite this via the portfolio argument. For example, the following uses the two paid triangles as the portfolio estimate:

\begin{verbatim}
R> lapply(summary(f1, portfolio = "1+3")@report.summary, 
\end{verbatim}

\begin{verbatim}
$`Summary Statistics for Triangle 1`
  Latest Dev.To.Date Ultimate IBNR S.E CV
Total 3290539 0.8537 3854572 564033 19089 0.0338

$`Summary Statistics for Triangle 2`
  Latest Dev.To.Date Ultimate IBNR S.E CV
Total 3710614 0.9884 3754197 43583 18839 0.4323
\end{verbatim}
Summary Statistics for Triangle 3
<table>
<thead>
<tr>
<th>Latest</th>
<th>Dev.To.Date</th>
<th>Ultimate</th>
<th>IBNR</th>
<th>S.E</th>
<th>CV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>1043851</td>
<td>0.7504</td>
<td>1391064</td>
<td>347213</td>
<td>27716</td>
</tr>
</tbody>
</table>

Summary Statistics for Triangle 1+3
<table>
<thead>
<tr>
<th>Latest</th>
<th>DevToDate</th>
<th>Ultimate</th>
<th>IBNR</th>
<th>S.E</th>
<th>CV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>4334390</td>
<td>0.8263</td>
<td>5245636</td>
<td>911246</td>
<td>38753</td>
</tr>
</tbody>
</table>

4.11 Joint modelling of the paid and incurred losses

Although the model with intercepts proved to be an improvement over the MCL model, it still fails to account for the structural relationship between triangles. In particular, it produces divergent paid-to-incurred loss ratios for the personal auto line:

```
R> ult <- summary(f1)$Ultimate
R> print(ult[, 1] /ult[, 2], 3)
```

```
1 2 3 4 5 6 7 8 9 10 Total
0.995 0.995 0.993 0.992 0.995 0.996 1.021 1.067 1.112 1.114 1.027
```

We see that for accident years 9-10, the paid-to-incurred loss ratios are more than 110%. This can be fixed by allowing the development of the paid/incurred triangles.
to depend on each other. That is, we include the past values from the paid triangle as predictors when developing the incurred triangle, and vice versa.

We illustrate this ignoring the commercial auto triangle. See the demo for a model that uses all three triangles. We also include the MCL model and the Munich chain-ladder as a comparison:

```r
R> da <- auto[1:2]
R> # MCL with diagonal development
R> M0 <- MultiChainLadder(da)
R> # non-diagonal development matrix with no intercepts
R> M1 <- MultiChainLadder2(da, type = "GMCL-int")
R> # Munich chain-ladder
R> M2 <- MunichChainLadder(da[[1]], da[[2]])
R> # compile results and compare projected paid to incurred ratios
R> r1 <- lapply(list(M0, M1), function(x){
    ult <- summary(x)@Ultimate
    ult[, 1] / ult[, 2]
  })
R> names(r1) <- c("MCL", "GMCL")
R> r2 <- summary(M2)[[1]][6]
R> r2 <- c(r2, summary(M2)[[2]][2, 3])
R> print(do.call(cbind, c(r1, list(MuCl = r2))) * 100, digits = 4)
```

```
     MCL GMCL  MuCl
 1  99.50  99.50  99.50
 2  99.49  99.49  99.55
 3  99.29  99.29 100.23
 4  99.20  99.20 100.23
 5  99.83  99.56 100.04
 6 100.43  99.66 100.03
 7 100.53  99.76  99.95
 8 111.24 100.02  99.81
 9 122.11 100.20  99.67
10 126.28 100.18  99.69
Total 105.58  99.68  99.88
```

### 5 Clark’s methods

The ChainLadder package contains functionality to carry out the methods described in the paper by David Clark [Cla03]. Using a longitudinal analysis approach, Clark assumes that losses develop according to a theoretical growth curve. The LDF method is a special case of this approach where the growth curve can

---

6This paper is on the CAS Exam 6 syllabus.
be considered to be either a step function or piecewise linear. Clark envisions a growth curve as measuring the percent of ultimate loss that can be expected to have emerged as of each age of an origin period. The paper describes two methods that fit this model.

The LDF method assumes that the ultimate losses in each origin period are separate and unrelated. The goal of the method, therefore, is to estimate parameters for the ultimate losses and for the growth curve in order to maximize the likelihood of having observed the data in the triangle.

The CapeCod method assumes that the apriori expected ultimate losses in each origin year are the product of earned premium that year and a theoretical loss ratio. The CapeCod method, therefore, need estimate potentially far fewer parameters: for the growth function and for the theoretical loss ratio.

One of the side benefits of using maximum likelihood to estimate parameters is that its associated asymptotic theory provides uncertainty estimates for the parameters. Observing that the reserve estimates by origin year are functions of the estimated parameters, uncertainty estimates of these functional values are calculated according to the Delta method, which is essentially a linearisation of the problem based on a Taylor series expansion.

The two functional forms for growth curves considered in Clark’s paper are the log-logistic function (a.k.a., the inverse power curve) and the Weibull function, both being two-parameter functions. Clark uses the parameters $\omega$ and $\theta$ in his paper. Clark’s methods work on incremental losses. His likelihood function is based on the assumption that incremental losses follow an over-dispersed Poisson (ODP) process.

### 5.1 Clark’s LDF method

Consider again the RAA triangle. Accepting all defaults, the Clark LDF Method would estimate total ultimate losses of 272,009 and a reserve (FutureValue) of 111,022, or almost twice the value based on the volume weighted average link ratios and loglinear fit in section 3.2.1 above.

```r
R> ClarkLDF(RAA)
```

<table>
<thead>
<tr>
<th>Origin</th>
<th>CurrentValue</th>
<th>Ldf</th>
<th>UltimateValue</th>
<th>FutureValue</th>
<th>StdError</th>
<th>CV%</th>
</tr>
</thead>
<tbody>
<tr>
<td>1981</td>
<td>18,834</td>
<td>1.216</td>
<td>22,906</td>
<td>4,072</td>
<td>2,792</td>
<td>68.6</td>
</tr>
<tr>
<td>1982</td>
<td>16,704</td>
<td>1.251</td>
<td>20,899</td>
<td>4,195</td>
<td>2,833</td>
<td>67.5</td>
</tr>
<tr>
<td>1983</td>
<td>23,466</td>
<td>1.297</td>
<td>30,441</td>
<td>6,975</td>
<td>4,050</td>
<td>58.1</td>
</tr>
<tr>
<td>1984</td>
<td>27,067</td>
<td>1.360</td>
<td>36,823</td>
<td>9,756</td>
<td>5,147</td>
<td>52.8</td>
</tr>
<tr>
<td>1985</td>
<td>26,180</td>
<td>1.451</td>
<td>37,996</td>
<td>11,816</td>
<td>5,858</td>
<td>49.6</td>
</tr>
<tr>
<td>1986</td>
<td>15,852</td>
<td>1.591</td>
<td>25,226</td>
<td>9,374</td>
<td>4,877</td>
<td>52.0</td>
</tr>
<tr>
<td>1987</td>
<td>12,314</td>
<td>1.829</td>
<td>22,528</td>
<td>10,214</td>
<td>5,206</td>
<td>51.0</td>
</tr>
<tr>
<td>1988</td>
<td>13,112</td>
<td>2.305</td>
<td>30,221</td>
<td>17,109</td>
<td>7,568</td>
<td>44.2</td>
</tr>
<tr>
<td>1989</td>
<td>5,395</td>
<td>3.596</td>
<td>19,399</td>
<td>14,004</td>
<td>7,506</td>
<td>53.6</td>
</tr>
</tbody>
</table>

41
Most of the difference is due to the heavy tail, 21.6%, implied by the inverse power curve fit. Clark recognizes that the log-logistic curve can take an unreasonably long length of time to flatten out. If according to the actuary’s experience most claims close as of, say, 20 years, the growth curve can be truncated accordingly by using the maxage argument:

\[ R > \text{ClarkLDF}(\text{RAA}, \text{maxage} = 20) \]

<table>
<thead>
<tr>
<th>Origin</th>
<th>CurrentValue</th>
<th>Ldf</th>
<th>UltimateValue</th>
<th>FutureValue</th>
<th>StdError</th>
<th>CV%</th>
</tr>
</thead>
<tbody>
<tr>
<td>1981</td>
<td>18,834</td>
<td>1.124</td>
<td>21,168</td>
<td>2,334</td>
<td>1,765</td>
<td>75.6</td>
</tr>
<tr>
<td>1982</td>
<td>16,704</td>
<td>1.156</td>
<td>19,314</td>
<td>2,610</td>
<td>1,893</td>
<td>72.6</td>
</tr>
<tr>
<td>1983</td>
<td>23,466</td>
<td>1.199</td>
<td>28,132</td>
<td>4,666</td>
<td>2,729</td>
<td>58.5</td>
</tr>
<tr>
<td>1984</td>
<td>27,067</td>
<td>1.257</td>
<td>34,029</td>
<td>6,962</td>
<td>3,559</td>
<td>51.1</td>
</tr>
<tr>
<td>1985</td>
<td>26,180</td>
<td>1.341</td>
<td>35,113</td>
<td>8,933</td>
<td>4,218</td>
<td>47.2</td>
</tr>
<tr>
<td>1986</td>
<td>15,852</td>
<td>1.471</td>
<td>23,312</td>
<td>7,460</td>
<td>3,775</td>
<td>50.6</td>
</tr>
<tr>
<td>1987</td>
<td>12,314</td>
<td>1.691</td>
<td>20,819</td>
<td>8,505</td>
<td>4,218</td>
<td>49.6</td>
</tr>
<tr>
<td>1988</td>
<td>13,112</td>
<td>2.130</td>
<td>27,928</td>
<td>14,816</td>
<td>6,300</td>
<td>42.5</td>
</tr>
<tr>
<td>1989</td>
<td>5,395</td>
<td>3.323</td>
<td>17,927</td>
<td>12,532</td>
<td>6,658</td>
<td>53.1</td>
</tr>
<tr>
<td>1990</td>
<td>2,063</td>
<td>11.454</td>
<td>23,629</td>
<td>21,566</td>
<td>15,899</td>
<td>73.7</td>
</tr>
<tr>
<td>Total</td>
<td>160,987</td>
<td></td>
<td>272,009</td>
<td>111,022</td>
<td>36,102</td>
<td>32.5</td>
</tr>
</tbody>
</table>

The Weibull growth curve tends to be faster developing than the log-logistic:

\[ R > \text{ClarkLDF}(\text{RAA}, \text{G} = \text{"weibull"}) \]

<table>
<thead>
<tr>
<th>Origin</th>
<th>CurrentValue</th>
<th>Ldf</th>
<th>UltimateValue</th>
<th>FutureValue</th>
<th>StdError</th>
<th>CV%</th>
</tr>
</thead>
<tbody>
<tr>
<td>1981</td>
<td>18,834</td>
<td>1.022</td>
<td>19,254</td>
<td>420</td>
<td>700</td>
<td>166.5</td>
</tr>
<tr>
<td>1982</td>
<td>16,704</td>
<td>1.037</td>
<td>17,317</td>
<td>613</td>
<td>855</td>
<td>139.5</td>
</tr>
<tr>
<td>1983</td>
<td>23,466</td>
<td>1.060</td>
<td>24,875</td>
<td>1,409</td>
<td>1,401</td>
<td>99.4</td>
</tr>
<tr>
<td>1984</td>
<td>27,067</td>
<td>1.098</td>
<td>29,728</td>
<td>2,661</td>
<td>2,037</td>
<td>76.5</td>
</tr>
<tr>
<td>1985</td>
<td>26,180</td>
<td>1.162</td>
<td>30,419</td>
<td>4,239</td>
<td>2,639</td>
<td>62.2</td>
</tr>
<tr>
<td>1986</td>
<td>15,852</td>
<td>1.271</td>
<td>20,151</td>
<td>4,299</td>
<td>2,549</td>
<td>59.3</td>
</tr>
<tr>
<td>1987</td>
<td>12,314</td>
<td>1.471</td>
<td>18,114</td>
<td>5,800</td>
<td>3,060</td>
<td>52.8</td>
</tr>
<tr>
<td>1988</td>
<td>13,112</td>
<td>1.883</td>
<td>24,692</td>
<td>11,580</td>
<td>4,867</td>
<td>42.0</td>
</tr>
<tr>
<td>1989</td>
<td>5,395</td>
<td>2.988</td>
<td>16,122</td>
<td>10,727</td>
<td>5,544</td>
<td>51.7</td>
</tr>
<tr>
<td>1990</td>
<td>2,063</td>
<td>9.815</td>
<td>23,927</td>
<td>18,185</td>
<td>12,929</td>
<td>71.1</td>
</tr>
<tr>
<td>Total</td>
<td>160,987</td>
<td></td>
<td>251,369</td>
<td>90,382</td>
<td>26,375</td>
<td>29.2</td>
</tr>
</tbody>
</table>

It is recommend to inspect the residuals to help assess the reasonableness of the model relative to the actual data.

Although there is some evidence of heteroscedasticity with increasing ages and fitted values, the residuals otherwise appear randomly scattered around a horizontal line.
5.2 Clark’s Cap Cod method

The RAA data set, widely researched in the literature, has no premium associated with it traditionally. Let’s assume a constant earned premium of 40000 each year, and a Weibull growth function:

\[ R> \text{ClarkCapeCod(RAA, Premium = 40000, } G = \text{"weibull"}) \]

<table>
<thead>
<tr>
<th>Origin</th>
<th>CurrentValue</th>
<th>Premium</th>
<th>ELR</th>
<th>FutureGrowthFactor</th>
<th>FutureValue</th>
<th>UltimateValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>1981</td>
<td>18,834</td>
<td>40,000</td>
<td>0.566</td>
<td>0.0192</td>
<td>436</td>
<td>19,270</td>
</tr>
<tr>
<td>1982</td>
<td>16,704</td>
<td>40,000</td>
<td>0.566</td>
<td>0.0320</td>
<td>725</td>
<td>17,429</td>
</tr>
<tr>
<td>1983</td>
<td>23,466</td>
<td>40,000</td>
<td>0.566</td>
<td>0.0525</td>
<td>1,189</td>
<td>24,655</td>
</tr>
</tbody>
</table>

As an exercise, the reader can confirm that the normal distribution assumption is rejected at the 5% level with the log-logistic curve.

\[ \text{Shapiro–Wilk p-value = 0.19684.} \]
The estimated expected loss ratio is 0.566. The total outstanding loss is about 10% higher than with the LDF method. The standard error, however, is lower, probably due to the fact that there are fewer parameters to estimate with the CapeCod method, resulting in less parameter risk.

A plot of this model shows similar residuals By Origin and Projected Age to those from the LDF method, a better spread By Fitted Value, and a slightly better q-q plot, particularly in the upper tail.

6 Generalised linear model methods

Recent years have also seen growing interest in using generalised linear models [GLM] for insurance loss reserving. The use of GLM in insurance loss reserving has many compelling aspects, e.g.,

- when over-dispersed Poisson model is used, it reproduces the estimates from chain-ladder;
- it provides a more coherent modelling framework than the Mack method;
- all the relevant established statistical theory can be directly applied to perform hypothesis testing and diagnostic checking;

The glmReserve function takes an insurance loss triangle, converts it to incremental losses internally if necessary, transforms it to the long format (see \texttt{as.data.frame})
and fits the resulting loss data with a generalised linear model where the mean structure includes both the accident year and the development lag effects. The function also provides both analytical and bootstrapping methods to compute the associated prediction errors. The bootstrapping approach also simulates the full predictive distribution, based on which the user can compute other uncertainty measures such as predictive intervals.

Only the Tweedie family of distributions are allowed, that is, the exponential family that admits a power variance function $V(\mu) = \mu^p$. The variance power $p$ is specified in the var.power argument, and controls the type of the distribution. When the Tweedie compound Poisson distribution $1 < p < 2$ is to be used, the user has the option to specify var.power = NULL, where the variance power $p$ will be estimated from the data using the cplm package [Zha12].

For example, the following fits the over-dispersed Poisson model and spells out the estimated reserve information:

R> # load data
R> data(GenIns)
R> GenIns <- GenIns / 1000
R> # fit Poisson GLM
R> (fit1 <- glmReserve(GenIns))

<table>
<thead>
<tr>
<th>Latest</th>
<th>Dev.To.Date</th>
<th>Ultimate</th>
<th>IBNR</th>
<th>S.E</th>
<th>CV</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5339</td>
<td>0.98252</td>
<td>5434</td>
<td>95</td>
<td>110.1 1.1589</td>
</tr>
<tr>
<td>3</td>
<td>4909</td>
<td>0.91263</td>
<td>5379</td>
<td>470</td>
<td>216.0 0.4597</td>
</tr>
<tr>
<td>4</td>
<td>4588</td>
<td>0.86599</td>
<td>5298</td>
<td>710</td>
<td>260.9 0.3674</td>
</tr>
<tr>
<td>5</td>
<td>3873</td>
<td>0.79725</td>
<td>4858</td>
<td>985</td>
<td>303.6 0.3082</td>
</tr>
<tr>
<td>6</td>
<td>3692</td>
<td>0.72235</td>
<td>5111</td>
<td>1419</td>
<td>375.0 0.2643</td>
</tr>
<tr>
<td>7</td>
<td>3483</td>
<td>0.61527</td>
<td>5661</td>
<td>2178</td>
<td>495.4 0.2274</td>
</tr>
<tr>
<td>8</td>
<td>2864</td>
<td>0.42221</td>
<td>6784</td>
<td>3920</td>
<td>790.0 0.2015</td>
</tr>
<tr>
<td>9</td>
<td>1363</td>
<td>0.24162</td>
<td>5642</td>
<td>4279</td>
<td>1046.5 0.2446</td>
</tr>
<tr>
<td>10</td>
<td>344</td>
<td>0.06922</td>
<td>4970</td>
<td>4626</td>
<td>1980.1 0.4280</td>
</tr>
<tr>
<td>total</td>
<td>30457</td>
<td>0.61982</td>
<td>49138</td>
<td>18681</td>
<td>2945.7 0.1577</td>
</tr>
</tbody>
</table>

We can also extract the underlying GLM model by specifying type = "model" in the summary function:

R> summary(fit1, type = "model")

Call:
glm(formula = value ~ factor(origin) + factor(dev), family = fam,
    data = ldaFit, offset = offset)

Deviance Residuals:

Min 1Q Median 3Q Max
-14.701 -3.913 -0.688 3.675 15.633

Coefficients:

                         Estimate Std. Error t value Pr(>|t|)
(Intercept)               5.59865  0.17292  32.38  < 2e-16 ***
factor(origin)2            0.33127  0.15354   2.16   0.0377 *
factor(origin)3            0.32112  0.15772   2.04   0.0492 *
factor(origin)4            0.30596  0.16074   1.90   0.0650 .
factor(origin)5            0.21932  0.16797   1.31   0.1999
factor(origin)6            0.27008  0.17076   1.58   0.1225
factor(origin)7            0.37221  0.17445   2.13   0.0398 *
factor(origin)8            0.55333  0.18653   2.97   0.0053 **
factor(origin)9            0.36893  0.23918   1.54   0.1317
factor(origin)10           0.24203  0.42756   0.57   0.5749
factor(dev)2               0.91253  0.14885   6.13   4.7e-07 ***
factor(dev)3               0.95883  0.15257   6.28   2.9e-07 ***
factor(dev)4               1.02600  0.15688   6.54   1.3e-07 ***
factor(dev)5               0.43528  0.18391   2.37   0.0234 *
factor(dev)6               0.08006  0.21477   0.37   0.7115
factor(dev)7  -0.00638  0.23829  -0.03  0.9788
factor(dev)8  -0.39445  0.31029  -1.27  0.2118
factor(dev)9   0.00938  0.32025   0.03  0.9768
factor(dev)10  -1.37991  0.89669  -1.54  0.1326

---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for Tweedie family taken to be 52.6)

Null deviance: 10699 on 54 degrees of freedom
Residual deviance: 1903 on 36 degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 4

Similarly, we can fit the Gamma and a compound Poisson GLM reserving model by changing the var.power argument:

R> # Gamma GLM
R> (fit2 <- glmReserve(GenIns, var.power = 2))

<table>
<thead>
<tr>
<th>Latest</th>
<th>Dev.To.Date</th>
<th>Ultimate</th>
<th>IBNR</th>
<th>S.E</th>
<th>CV</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5339</td>
<td>0.98288</td>
<td>5432</td>
<td>93</td>
<td>45.17</td>
</tr>
<tr>
<td>3</td>
<td>4909</td>
<td>0.91655</td>
<td>5366</td>
<td>447</td>
<td>160.56</td>
</tr>
<tr>
<td>4</td>
<td>4588</td>
<td>0.88248</td>
<td>5199</td>
<td>611</td>
<td>177.62</td>
</tr>
<tr>
<td>5</td>
<td>3873</td>
<td>0.79611</td>
<td>4865</td>
<td>992</td>
<td>254.47</td>
</tr>
<tr>
<td>6</td>
<td>3692</td>
<td>0.71757</td>
<td>5145</td>
<td>1453</td>
<td>351.33</td>
</tr>
<tr>
<td>7</td>
<td>3483</td>
<td>0.61440</td>
<td>5669</td>
<td>2186</td>
<td>526.29</td>
</tr>
<tr>
<td>8</td>
<td>2864</td>
<td>0.43870</td>
<td>6529</td>
<td>3665</td>
<td>941.32</td>
</tr>
<tr>
<td>9</td>
<td>1363</td>
<td>0.24854</td>
<td>5485</td>
<td>4122</td>
<td>1175.95</td>
</tr>
<tr>
<td>10</td>
<td>344</td>
<td>0.07078</td>
<td>4860</td>
<td>4516</td>
<td>1667.39</td>
</tr>
<tr>
<td>total</td>
<td>30457</td>
<td>0.62742</td>
<td>48543</td>
<td>18086</td>
<td>2702.71</td>
</tr>
</tbody>
</table>

R> # compound Poisson GLM (variance function estimated from the data):
R> #(fit3 <- glmReserve(GenIns, var.power = NULL))

By default, the formulaic approach is used to compute the prediction errors. We can also carry out bootstrapping simulations by specifying mse.method = "bootstrap" (note that this argument supports partial match):

R> set.seed(11)
R> (fit5 <- glmReserve(GenIns, mse.method = "boot"))

<table>
<thead>
<tr>
<th>Latest</th>
<th>Dev.To.Date</th>
<th>Ultimate</th>
<th>IBNR</th>
<th>S.E</th>
<th>CV</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5339</td>
<td>0.98252</td>
<td>5434</td>
<td>95</td>
<td>108.7</td>
</tr>
</tbody>
</table>
When bootstrapping is used, the resulting object has three additional components - "sims.par", "sims.reserve.mean", and "sims.reserve.pred" that store the simulated parameters, mean values and predicted values of the reserves for each year, respectively.

\[
\begin{array}{cccccc}
3 & 4909 & 0.91263 & 5379 & 470 & 207.8 & 0.4421 \\
4 & 4588 & 0.86599 & 5298 & 710 & 270.4 & 0.3809 \\
5 & 3873 & 0.79725 & 4858 & 985 & 306.6 & 0.3112 \\
6 & 3692 & 0.72235 & 5111 & 1419 & 387.6 & 0.2732 \\
7 & 3483 & 0.61527 & 5661 & 2178 & 493.3 & 0.2265 \\
8 & 2864 & 0.42221 & 6784 & 3920 & 813.1 & 0.2074 \\
9 & 1363 & 0.24162 & 5642 & 4279 & 1077.3 & 0.2518 \\
10 & 344 & 0.06922 & 4970 & 4626 & 1986.3 & 0.4294 \\
\text{total} & 30457 & 0.61982 & 49138 & 18681 & 2955.4 & 0.1582 \\
\end{array}
\]

We can thus compute the quantiles of the predictions based on the simulated samples in the "sims.reserve.pred" element as:

\[
R> \text{pr} <- \text{as.data.frame(fit5$sims.reserve.pred)} \\
R> \text{qv} <- c(0.025, 0.25, 0.5, 0.75, 0.975) \\
R> \text{res.q} <- t(\text{apply(pr, 2, quantile, qv)}) \\
R> \text{print(format(round(res.q), big.mark = ",", quote = FALSE))}
\]

\[
\begin{array}{cccccc}
2.5% & 25% & 50% & 75% & 97.5% \\
2 & 0 & 40 & 91 & 175 & 375 \\
3 & 125 & 339 & 463 & 612 & 941 \\
4 & 290 & 551 & 720 & 910 & 1,330 \\
5 & 507 & 797 & 991 & 1,195 & 1,686 \\
6 & 812 & 1,194 & 1,424 & 1,691 & 2,305 \\
7 & 1,345 & 1,873 & 2,161 & 2,536 & 3,258 \\
8 & 2,595 & 3,427 & 3,951 & 4,517 & 5,742 \\
9 & 2,462 & 3,555 & 4,222 & 4,988 & 6,879 \\
10 & 809 & 3,394 & 4,505 & 5,904 & 9,236 \\
\end{array}
\]

The full predictive distribution of the simulated reserves for each year can be visualized easily:

\[
R> \text{library(ggplot2)} \\
R> \text{prm} <- \text{reshape(pr, varying=list(names(pr)), v.names = "reserve",}
\]

48
```r
R> gg <- ggplot(prm, aes(reserve))
R> gg <- gg + geom_density(aes(fill = year), alpha = 0.3) +
    facet_wrap(~year, nrow = 2, scales = "free") +
    theme(legend.position = "none")
R> print(gg)
```

![Figure 7: The predictive distribution of loss reserves for each year based on bootstrapping.](image)

### 7 Paid-incurred chain model

The Paid-incurred chain model was published by Merz and Wüthrich in 2010 [MW10]. It combines claims payments and incurred losses information in a mathematically rigorous and consistent way to get a unified ultimate loss prediction.

#### 7.1 Model assumptions

The model assumptions for the Log-Normal PIC Model are the following:

- Conditionally, given \( \Theta = (\Phi_0, \ldots, \Phi_I, \Psi_0, \ldots, \Psi_{I-1}, \sigma_0, \ldots, \sigma_{I-1}, \tau_0, \ldots, \tau_{I-1}) \)
  we have
the random vector \((\xi_{0,0}, ..., \xi_{I,I}, \zeta_{0,0}, ..., \zeta_{I,I-1})\) has multivariate Gaussian distribution with uncorrelated components given by
\[
\xi_{i,j} \sim N(\Phi_j, \sigma_j^2), \\
\zeta_{k,l} \sim N(\Psi_l, \tau_l^2);
\]
- cumulative payments are given by the recursion
\[
P_{i,j} = P_{i,j-1} \exp(\xi_{i,j}),
\]
with initial value \(P_{i,0} = \exp(\xi_{i,0})\);
- incurred losses \(I_{i,j}\) are given by the backwards recursion
\[
I_{i,j-1} = I_{i,j} \exp(-\zeta_{i,j-1}),
\]
with initial value \(I_{i,I} = P_{i,I}\).
- The components of \(\Theta\) are independent and \(\sigma_j, \tau_j > 0\) for all \(j\).

### 7.2 Parameter estimation

Parameters \(\Theta\) in the model are in general not known and need to be estimated from observations. They are estimated in a Bayesian framework. In the Bayesian PIC model they assume that the previous assumptions hold true with deterministic \(\sigma_0, ..., \sigma_J\) and \(\tau_0, ..., \tau_{J-1}\) and
\[
\Phi_m \sim N(\phi_m, s_m^2), \\
\Psi_n \sim N(\psi_n, t_n^2).
\]

This is not a full Bayesian approach but has the advantage to give analytical expressions for the posterior distributions and the prediction uncertainty.

The Paid-incurred Chain model is implemented in the ChainLadder package via the function \texttt{PaidIncurredChain}. As an example we apply the function to the USAA paid and incurred triangles:

```r
R> library(ChainLadder)
R> PIC <- PaidIncurredChain(USAApaid, USAAincurred)
R> PIC
$Ult.Loss.Origin
  [,1]
[1,] 983113
[2,] 1078697
[3,] 1145761
[4,] 1245171
[5,] 1371964
[6,] 1433857
```

50
$\text{Ult.Loss}$

[1] 11405008

$\text{Res.Origin}$

[,1]
[1,]  965.3
[2,] 3159.6
[3,]  7386.4
[4,] 18521.4
[5,]  47232.4
[6,] 113727.4
[7,] 230663.7
[8,]  443903.3
[9,]  778393.5

$\text{Res.Tot}$

[1] 1643953

$s.e.$

[1] 113940

We can access the reserves by origin year via

\begin{verbatim}
R> PIC$Res.Origin
\end{verbatim}

[,1]
[1,]  965.3
[2,] 3159.6
[3,]  7386.4
[4,] 18521.4
[5,]  47232.4
[6,] 113727.4
[7,] 230663.7
[8,]  443903.3
[9,]  778393.5

and the total reserve via

\begin{verbatim}
R> PIC$Res.Tot
\end{verbatim}

[1] 1643953

s.e. is the square root of mean square error of prediction for the total ultimate loss.
It’s important to notice that the model is implemented in the special case of non-informative priors for \( \Phi_m \) and \( \Psi_n \); this means that we let \( s^2_m \rightarrow \infty \) and \( t^2_n \rightarrow \infty \).

8 One year claims development result

The stochastic claims reserving methods considered above predict the lower (unknown) triangle and assess the uncertainty of this prediction. For instance, Mack’s uncertainty formula quantifies the total prediction uncertainty of the chain-ladder predictor over the entire run-off of the outstanding claims. Modern solvency considerations, such as Solvency II, require a second view of claims reserving uncertainty. This second view is a short-term view because it requires assessments of the one-year changes of the claims predictions when one updates the available information at the end of each accounting year. At time \( t \geq n \) we have information

\[
D_t = \{ C_{i,k}; i + k \leq t + 1 \}.
\]

This motivates the following sequence of predictors for the ultimate claim \( C_{i,K} \) at times \( t \geq n \)

\[
\hat{C}^{(t)}_{i,K} = \mathbb{E}[C_{i,K}|D_t].
\]

The one year claims development results (CDR), see Merz-Wüthrich [MW08a, MW14], consider the changes in these one year updates, that is,

\[
\text{CDR}_{i,t+1} = \hat{C}^{(t)}_{i,K} - \hat{C}^{(t+1)}_{i,K}.
\]

The tower property of conditional expectation implies that the CDRs are on average 0, that is, \( \mathbb{E}[\text{CDR}_{i,t+1}|D_t] = 0 \) and the Merz-Wüthrich formula [MW08a, MW14] assesses the uncertainty of these predictions measured by the following conditional mean square error of prediction (MSEP)

\[
\text{mse}_{\text{CDR}_{i,t+1}|D_t}(0) = \mathbb{E} \left[ (\text{CDR}_{i,t+1} - 0)^2 | D_t \right].
\]

The major difficulty in the evaluation of the conditional MSEP is the quantification of parameter estimation uncertainty.

8.1 CDR functions

The one year claims development result (CDR) can be estimate via the generic CDR function for objects of MackChainLadder and BootChainLadder.

Further, the tweedieReserve function offers also the option to estimate the one year CDR, by setting the argument \texttt{rereserving}=TRUE.

For example, to reproduce the results of [MW14] use:
```r
R> M <- MackChainLadder(MW2014, est.sigma="Mack")
R> cdrM <- CDR(M)
R> round(cdrM, 1)

<table>
<thead>
<tr>
<th></th>
<th>IBNR</th>
<th>CDR(1)</th>
<th>S.E.</th>
<th>Mack.S.E.</th>
<th>S.E.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
</tr>
<tr>
<td>3</td>
<td>10.1</td>
<td>2.5</td>
<td>2.6</td>
<td>2.6</td>
<td>2.6</td>
</tr>
<tr>
<td>4</td>
<td>21.2</td>
<td>16.7</td>
<td>16.9</td>
<td>16.9</td>
<td>16.9</td>
</tr>
<tr>
<td>5</td>
<td>117.7</td>
<td>156.4</td>
<td>157.3</td>
<td>157.3</td>
<td>157.3</td>
</tr>
<tr>
<td>6</td>
<td>223.3</td>
<td>137.7</td>
<td>207.2</td>
<td>207.2</td>
<td>207.2</td>
</tr>
<tr>
<td>7</td>
<td>361.8</td>
<td>171.2</td>
<td>261.9</td>
<td>261.9</td>
<td>261.9</td>
</tr>
<tr>
<td>8</td>
<td>469.4</td>
<td>70.3</td>
<td>292.3</td>
<td>292.3</td>
<td>292.3</td>
</tr>
<tr>
<td>9</td>
<td>653.5</td>
<td>271.6</td>
<td>390.6</td>
<td>390.6</td>
<td>390.6</td>
</tr>
<tr>
<td>10</td>
<td>1008.8</td>
<td>310.1</td>
<td>502.1</td>
<td>502.1</td>
<td>502.1</td>
</tr>
<tr>
<td>11</td>
<td>1011.9</td>
<td>103.4</td>
<td>486.1</td>
<td>486.1</td>
<td>486.1</td>
</tr>
<tr>
<td>12</td>
<td>1406.7</td>
<td>632.6</td>
<td>806.9</td>
<td>806.9</td>
<td>806.9</td>
</tr>
<tr>
<td>13</td>
<td>1492.9</td>
<td>315.0</td>
<td>793.9</td>
<td>793.9</td>
<td>793.9</td>
</tr>
<tr>
<td>14</td>
<td>1917.6</td>
<td>406.1</td>
<td>891.7</td>
<td>891.7</td>
<td>891.7</td>
</tr>
<tr>
<td>15</td>
<td>2458.2</td>
<td>285.2</td>
<td>916.5</td>
<td>916.5</td>
<td>916.5</td>
</tr>
<tr>
<td>16</td>
<td>3384.3</td>
<td>668.2</td>
<td>1106.1</td>
<td>1106.1</td>
<td>1106.1</td>
</tr>
<tr>
<td>17</td>
<td>9596.6</td>
<td>733.2</td>
<td>1295.7</td>
<td>1295.7</td>
<td>1295.7</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>24134.9</td>
<td>1842.9</td>
<td>3233.7</td>
<td>3233.7</td>
</tr>
</tbody>
</table>

To review the full claims development picture set the argument dev="all":

```r
R> cdrAll <- CDR(M, dev="all")
R> round(cdrAll, 1)

<table>
<thead>
<tr>
<th></th>
<th>IBNR</th>
<th>CDR(1)</th>
<th>S.E.</th>
<th>CDR(2)</th>
<th>S.E.</th>
<th>CDR(3)</th>
<th>S.E.</th>
<th>CDR(4)</th>
<th>S.E.</th>
<th>CDR(5)</th>
<th>S.E.</th>
<th>CDR(6)</th>
<th>S.E.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>0.4</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>3</td>
<td>10.1</td>
<td>2.5</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>4</td>
<td>21.2</td>
<td>16.7</td>
<td>2.4</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>5</td>
<td>117.7</td>
<td>156.4</td>
<td>16.4</td>
<td>2.4</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>6</td>
<td>223.3</td>
<td>137.7</td>
<td>154.0</td>
<td>16.1</td>
<td>2.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>7</td>
<td>361.8</td>
<td>171.2</td>
<td>131.0</td>
<td>148.0</td>
<td>15.5</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>8</td>
<td>469.4</td>
<td>70.3</td>
<td>185.0</td>
<td>141.8</td>
<td>160.9</td>
<td>16.8</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>9</td>
<td>653.5</td>
<td>271.6</td>
<td>61.8</td>
<td>178.0</td>
<td>136.9</td>
<td>155.7</td>
<td>16.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>10</td>
<td>1008.8</td>
<td>310.1</td>
<td>59.0</td>
<td>180.4</td>
<td>138.6</td>
<td>158.1</td>
<td>150.1</td>
<td>170.9</td>
<td>131.4</td>
<td>176.6</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>11</td>
<td>1011.9</td>
<td>103.4</td>
<td>260.0</td>
<td>53.0</td>
<td>170.9</td>
<td>131.4</td>
<td>176.6</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>12</td>
<td>1406.7</td>
<td>632.6</td>
<td>302.2</td>
<td>268.7</td>
<td>52.8</td>
<td>176.6</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>13</td>
<td>1492.9</td>
<td>315.0</td>
<td>572.1</td>
<td>86.6</td>
<td>273.0</td>
<td>242.8</td>
<td>45.4</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>14</td>
<td>1917.6</td>
<td>406.1</td>
<td>313.3</td>
<td>573.0</td>
<td>84.4</td>
<td>273.1</td>
<td>243.1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>15</td>
<td>2458.2</td>
<td>285.2</td>
<td>395.5</td>
<td>305.3</td>
<td>560.8</td>
<td>80.1</td>
<td>267.1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>16</td>
<td>3384.3</td>
<td>668.2</td>
<td>271.7</td>
<td>380.2</td>
<td>293.3</td>
<td>540.6</td>
<td>75.7</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>17</td>
<td>9596.6</td>
<td>733.2</td>
<td>645.4</td>
<td>261.0</td>
<td>367.0</td>
<td>282.8</td>
<td>522.9</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>24134.9</td>
<td>1842.9</td>
<td>1208.3</td>
<td>1071.1</td>
<td>901.1</td>
<td>785.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>
```
<table>
<thead>
<tr>
<th></th>
<th>CDR(7)S.E.</th>
<th>CDR(8)S.E.</th>
<th>CDR(9)S.E.</th>
<th>CDR(10)S.E.</th>
<th>CDR(11)S.E.</th>
<th>CDR(12)S.E.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>4</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>5</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>6</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>7</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>8</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>9</td>
<td>2.3</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>10</td>
<td>16.6</td>
<td>2.4</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>11</td>
<td>150.4</td>
<td>15.7</td>
<td>2.3</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>12</td>
<td>135.8</td>
<td>155.6</td>
<td>16.3</td>
<td>2.3</td>
<td>0.3</td>
<td>0.0</td>
</tr>
<tr>
<td>13</td>
<td>159.7</td>
<td>122.9</td>
<td>141.1</td>
<td>14.8</td>
<td>2.1</td>
<td>0.3</td>
</tr>
<tr>
<td>14</td>
<td>44.1</td>
<td>159.9</td>
<td>123.0</td>
<td>141.4</td>
<td>14.8</td>
<td>2.1</td>
</tr>
<tr>
<td>15</td>
<td>237.9</td>
<td>42.3</td>
<td>156.4</td>
<td>120.4</td>
<td>138.5</td>
<td>14.5</td>
</tr>
<tr>
<td>16</td>
<td>257.3</td>
<td>229.3</td>
<td>39.9</td>
<td>150.8</td>
<td>116.1</td>
<td>133.6</td>
</tr>
<tr>
<td>17</td>
<td>71.8</td>
<td>248.8</td>
<td>221.7</td>
<td>38.1</td>
<td>145.9</td>
<td>112.3</td>
</tr>
<tr>
<td>Total</td>
<td>525.2</td>
<td>476.3</td>
<td>366.4</td>
<td>269.3</td>
<td>245.0</td>
<td>180.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>CDR(13)S.E.</th>
<th>CDR(14)S.E.</th>
<th>CDR(15)S.E.</th>
<th>CDR(16)S.E.</th>
<th>CDR(17)S.E.</th>
<th>Mack.S.E.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.4</td>
</tr>
<tr>
<td>3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>2.6</td>
</tr>
<tr>
<td>4</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>16.9</td>
</tr>
<tr>
<td>5</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>157.3</td>
</tr>
<tr>
<td>6</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>207.2</td>
</tr>
<tr>
<td>7</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>261.9</td>
</tr>
<tr>
<td>8</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>292.3</td>
</tr>
<tr>
<td>9</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>390.6</td>
</tr>
<tr>
<td>10</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>502.1</td>
</tr>
<tr>
<td>11</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>486.1</td>
</tr>
<tr>
<td>12</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>806.9</td>
</tr>
<tr>
<td>13</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>793.9</td>
</tr>
<tr>
<td>14</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>891.7</td>
</tr>
<tr>
<td>15</td>
<td>2.1</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>916.5</td>
</tr>
<tr>
<td>16</td>
<td>14.0</td>
<td>2.0</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>1106.1</td>
</tr>
<tr>
<td>17</td>
<td>129.3</td>
<td>13.5</td>
<td>1.9</td>
<td>0.3</td>
<td>0.0</td>
<td>1295.7</td>
</tr>
<tr>
<td>Total</td>
<td>130.1</td>
<td>13.7</td>
<td>2.0</td>
<td>0.3</td>
<td>0.0</td>
<td>3233.7</td>
</tr>
</tbody>
</table>

See the help files to CDR and tweedieReserve for more details.

## 9 Model Validation with tweedieReserve

Model validation is one of the key activities when an insurance company goes through the Internal Model Approval Process with the regulator. This section
gives some examples how the arguments of the `tweedieReserve` function can be used to validate a stochastic reserving model. The argument `design.type` allows us to test different regression structures. The classic over-dispersed Poisson (ODP) model uses the following structure:

\[ Y \sim \text{as.factor}(OY) + \text{as.factor}(DY), \]

(i.e. `design.type=c(1,1,0)`). This allows, together with the log link, to achieve the same results of the (volume weighted) chain-ladder model, thus the same model implied assumptions. A common model shortcoming is when the residuals plotted by calendar period start to show a pattern, which chain-ladder isn’t capable to model. In order to overcome this, the user could be then interested to change the regression structure in order to try to strip out these patterns [GS05]. For example, a regression structure like:

\[ Y \sim \text{as.factor}(DY) + \text{as.factor}(CY), \]

i.e. `design.type=c(0,1,1)` could be considered instead. This approach returns the same results of the arithmetic separation method, modelling explicitly inflation parameters between consequent calendar periods. Another interesting assumption is the assumed underlying distribution. The ODP model assumes the following:

\[ P_{i,j} \sim \text{ODP}(m_{i,j}, \phi \cdot m_{i,j}), \]

which is a particular case of a Tweedie distribution, with \( p \) parameter equals to 1. Generally speaking, for any random variable \( Y \) that obeys a Tweedie distribution, the variance \( \mathbb{V}[Y] \) relates to the mean \( \mathbb{E}[Y] \) by the following law:

\[ \mathbb{V}[Y] = a \cdot \mathbb{E}[Y]^p, \]

where \( a \) and \( p \) are positive constants. The user is able to test different \( p \) values through the `var.power` function argument. Besides, in order to validate the Tweedie’s \( p \) parameter, it could be interesting to plot the likelihood profile at defined \( p \) values (through the `p.check` argument) for a given dataset and a regression structure. This could be achieved setting the `p.optim=TRUE` argument, as follows:

```R
R> p_profile <- tweedieReserve(MW2008, p.optim=TRUE,
                              p.check=c(0,1.1,1.2,1.3,1.4,1.5,2,3),
                              design.type=c(0,1,1),
                              rerereserving=FALSE,
                              bootstrap=0,
                              progressBar=FALSE)
R> # 0 1.1 1.2 1.3 1.4 1.5 2 3
R> # .........Done.
R> # MLE of p is between 0 and 1, which is impossible.
R> # Instead, the MLE of p has been set to NA.
```
This example shows, see Figure 8, that the MLE of $p$ seems to be between 0 and 1, which is not possible as Tweedie models aren’t defined for $0 < p < 1$, thus the Error message. But, despite this, we can conclude that overall a value $p=1$ could be reasonable for this dataset and the chosen regression function, as it seems to be near the MLE. Other sensitivities could be run on:

- Bootstrap type (parametric / semi-parametric), via the `bootstrap` argument
- Bias adjustment (if using semi-parametric bootstrap), via the `boot.adj` argument

Please refer to `help(tweedieReserve)` for additional information.

10 Using ChainLadder with RExcel and SWord

The ChainLadder package comes with example files which demonstrate how its functions can be embedded in Excel and Word using the `statconn` interface. The spreadsheet is located in the Excel folder of the package. The R command

```r
R> system.file("Excel", package="ChainLadder")
```

will tell you the exact path to the directory. To use the spreadsheet you will need the RExcel-Add-in. The package also provides an example SWord file, demonstrating how the functions of the package can be integrated
into a MS Word file via SWord [BN07]. Again you find the Word file via the command:

```
R> system.file("SWord", package="ChainLadder")
```

The package comes with several demos to provide you with an overview of the package functionality, see

```
R> demo(package="ChainLadder")
```

## 11 Further resources

Other useful documents and resources to get started with R in the context of actuarial work:

- Introduction to R for Actuaries [DS06].
- Computational Actuarial Science with R [Cha14]
- Modern Actuarial Risk Theory – Using R [KGDD01]
- An Actuarial Toolkit [MSH+06].
- Mailing list R-SIG-insurance⁸: Special Interest Group on using R in actuarial science and insurance

### 11.1 Other insurance related R packages

Below is a list of further R packages in the context of insurance. The list is by no-means complete, and the CRAN Task Views 'Empirical Finance' and Probability Distributions will provide links to additional resources. Please feel free to contact us with items to be added to the list.

- cplm: Likelihood-based and Bayesian methods for fitting Tweedie compound Poisson linear models [Zha12].
- lossDev: A Bayesian time series loss development model. Features include skewed-t distribution with time-varying scale parameter, Reversible Jump MCMC for determining the functional form of the consumption path, and a structural break in this path [LS11].
- DCL: Claims Reserving under the Double Chain-Ladder Model. Statistical modelling and forecasting in claims reserving in non-life insurance under the Double Chain-Ladder framework by [MNV12].
- favir: Formatted Actuarial Vignettes in R. FAViR lowers the learning curve of the R environment. It is a series of peer-reviewed Sweave papers that use a consistent style [Esc11].
- actuar: Loss distributions modelling, risk theory (including ruin theory), simulation of compound hierarchical models and credibility theory [DGP08].

⁸https://stat.ethz.ch/mailman/listinfo/r-sig-insurance
– fitdistrplus: Help to fit of a parametric distribution to non-censored or censored data [DMPDD10].
– mondate: R package to keep track of dates in terms of months [Mur11].
– MRMR: Multivariate Regression Models for Reserving [Fan13].

References


