Claims reserving with R: ChainLadder-0.1.5-2 Package Vignette DRAFT

Markus Gesmann*

March 25, 2012

Abstract

The ChainLadder package provides various statistical methods which are typically used for the estimation of outstanding claims reserves in general insurance.

The package has implementations of the Mack-, Munich-, Bootstrap, and multi-variate chain-ladder methods, as well as the loss development factor curve fitting methods of Dave Clark and generalised linear model based reserving models.

This document is still in a draft stage. Any pointers which will help to iron out errors, clarify and make this document more helpful will be much appreciated.

*markus.gesmann@gmail.com

Contents

1 Introduction

1.1 Claims reserving in insurance

Unlike other industries the insurance industry 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 of their service, but rely on historical data analysis and judgement to derive 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 claims arising from casualty insurance can take a long time to settle. Claims can take years to materialise. A complex and costly example are the claims from asbestos liabilities. A research report by a working party of the Institute of Actuaries has estimated that the undiscounted cost of UK mesothelioma-related claims to the UK Insurance Market for the period 2009 to 2050 could be around £10bn [**?**]. The cost for aAsbestos related claims in the US for the worldwide insurance industry was estimate to be around $120bn in 2002 [**?**].

Thus, it should come to no surprise that the biggest item on the liability 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 out-standings claims), which are losses already reported to the insurance company and incurred but not reported (IBNR) claims.

Over the years several methods have been developed to estimate reserves for insurance claims, see [**?**], [**?**] for an overview. Changes in regulatory requirements, e.g. Solvency II[1] in Europe, have fostered further research into this topic, with a focus on stochastic and statistical techniques.

2 The `ChainLadder` package

2.1 Motivation

The `ChainLadder` [**?**] 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.

[1] See http://ec.europa.eu/internal_market/insurance/solvency/index_en.htm

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 extension
- many interfaces to data bases and other applications, such as MS Excel
- an established framework for documentation and testing
- 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 LaTeX and Sweave 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 functionallity of the `ChainLadder` package. The functions are discussed and explained in more detail in the respective help files and examples.

The `ChainLadder` package has implementations of the Mack-, Munich- and Bootstrap chain-ladder methods [?], [?], [?], [?]. Since version 0.1.3-3 it provides general multivariate chain ladder models by Wayne Zhang [?]. Version 0.1.4-0 introduced new functions on loss development factor (LDF) fitting methods and Cape Cod by Daniel Murphy following a paper by David Clark [?]. Version 0.1.5-0 has added loss reserving models within the generalized linear model framework following a paper by England and Verrall [?] implemented by Wayne Zhang.

The package also offers utility functions to convert quickly tables into triangles, triangles into tables, cumulative into incremental and incremental into cumulative triangles.

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

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

and can be executed via

[2] For an example see the project: Formatted Actuarial Vignettes in R, http://www.favir.net/

```
R> library(ChainLadder)
R> demo("demo name")
```

Additionally the ChainLadder package comes with example files which demonstrates how to the ChainLadder functions can be embedded in Excel and Word using the statconn interface[**?**].

For more information and examples see the project web site: http://code.google.com/p/chainladder/

2.3 Installation

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

```
R> install.packages('ChainLadder')
```

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

```
R> library(ChainLadder)

ChainLadder version 0.1.5-2 by:
Markus Gesmann <markus.gesmann@gmail.com>
Wayne Zhang <actuary_zhang@hotmail.com>
Daniel Murphy <danielmarkmurphy@gmail.com>

Type library(help='ChainLadder') or ?ChainLadder
to see overall documentation.

Type demo(ChainLadder) to get an idea of the functionality of this package.

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

Feel free to send us an email if you would like to be kept informed of
new versions or if you have any feedback, ideas, suggestions or would
like to collaborate.

More information is available on the ChainLadder project web-site:
http://code.google.com/p/chainladder/

To suppress this message use the statement:
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 origin period. An origin period could be the year the policy was sold, or the accident year. Of course the frequency doesn't have to be yearly, e.g. quarterly of monthly origin periods are also often used. 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
```

```
      dev
origin    1     2     3     4     5     6     7     8     9    10
  1981 5012  8269 10907 11805 13539 16181 18009 18608 18662 18834
  1982  106  4285  5396 10666 13782 15599 15496 16169 16704    NA
  1983 3410  8992 13873 16141 18735 22214 22863 23466    NA    NA
  1984 5655 11555 15766 21266 23425 26083 27067    NA    NA    NA
  1985 1092  9565 15836 22169 25955 26180    NA    NA    NA    NA
  1986 1513  6445 11702 12935 15852    NA    NA    NA    NA    NA
  1987  557  4020 10946 12314    NA    NA    NA    NA    NA    NA
  1988 1351  6947 13112    NA    NA    NA    NA    NA    NA    NA
  1989 3133  5395    NA    NA    NA    NA    NA    NA    NA    NA
  1990 2063    NA    NA    NA    NA    NA    NA    NA    NA    NA
```

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. Eventually all claims for a given origin period wil 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> plot(RAA)
```

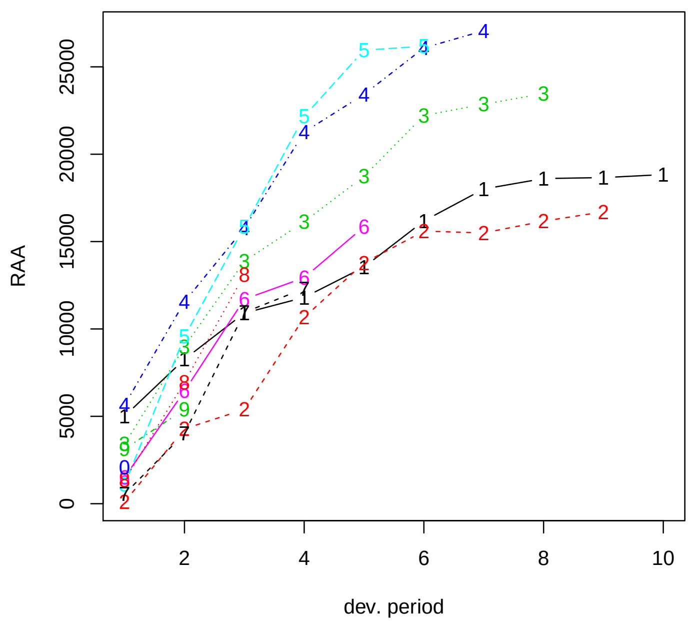


Figure 1: Claims development chart of the `RAA` triangle, with one line per origin period. Output of `plot(RAA)`

Setting the argument `lattice=TRUE` will produce individual plots for each origin period[3], see Figure 2.

```
R> 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] `ChainLadder` uses the `lattice` package for plotting the development of the origin years in separate panels.

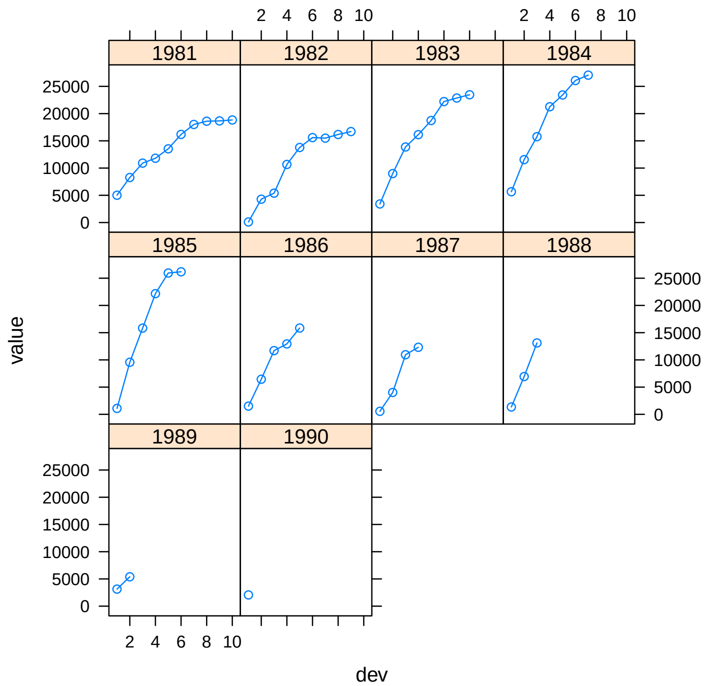


Figure 2: Claims development chart of the RAA triangle, with individual panels for each origin period. Output of `plot(RAA, lattice=TRUE)`

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,]
```

```
   1    2    3    4    5    6    7    8    9   10
5012 3257 2638  898 1734 2642 1828  599   54  172
```

```
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. R makes it easy to access data using SQL statements, e.g. via an ODBC connection[4] and the ChainLadder packages includes a demo to showcase how data can be imported from a MS Access data base, see:

```
R> demo(DatabaseExamples)
```

For more details see [**?**].

In this section we use data stored in a CSV-file[5] to demonstrate some typical operations you will want to carry out with data stored in data bases. 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)
```

```
  origin dev  value lob
1   1977   1 153638 ABC
2   1978   1 178536 ABC
3   1979   1 210172 ABC
4   1980   1 211448 ABC
5   1981   1 219810 ABC
6   1982   1 205654 ABC
```

```
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
```

[4]See the RODBC package

[5]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.

```
 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
  1984 5655 5900 4211 5500 2159 2658  984  NA  NA  NA
  1985 1092 8473 6271 6333 3786  225   NA  NA  NA  NA
  1986 1513 4932 5257 1233 2917   NA   NA  NA  NA  NA
  1987  557 3463 6926 1368   NA   NA   NA  NA  NA  NA
  1988 1351 5596 6165   NA   NA   NA   NA  NA  NA  NA
  1989 3133 2262   NA   NA   NA   NA   NA  NA  NA  NA
  1990 2063   NA   NA   NA   NA   NA   NA  NA  NA  NA
```

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> 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 particular helpful when you would like to store your results back into data base. Figure 3 gives you an idea of a potential data flow between R and data bases.

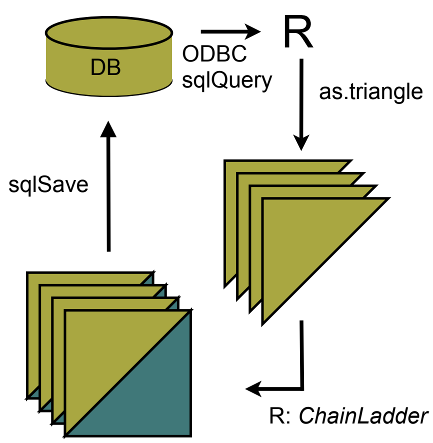


Figure 3: Flow chart of data between R and data bases.

3.1.4 Coping and pasting from MS Excel

Small data sets in Excel can be transfered to R backwards and forwards with via the clipboard under MS Windows.

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

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

Copying from R to Excel Suppose you would like to copy the RAA triangle into Excel, then the following statement would copy the data into the clipboard:

```
R> write.table(RAA, file="clipboard", sep="\t", na="")
```

Now you can paste the content into Excel. Please note that you can't copy lists structures from R to Excel.

3.2 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.

3.2.1 Basic idea

The age-to-age link ratios are calculated as the volume weighted average development ratios from one development period to the next of a cumulative loss development triangle $C_{ik}, i, k = 1, \dots, n$.

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

```
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

R> fullRAA <- RAA
R> for(k in 1:(n-1)){
+   fullRAA[(n-k+1):n, k+1] <- fullRAA[(n-k+1):n,k]*f[k]
+ }
R> round(fullRAA)

      dev
origin    1     2     3     4     5     6     7     8     9    10
  1981 5012  8269 10907 11805 13539 16181 18009 18608 18662 18834
  1982  106  4285  5396 10666 13782 15599 15496 16169 16704 16858
```

```
  1983 3410  8992 13873 16141 18735 22214 22863 23466 23863 24083
  1984 5655 11555 15766 21266 23425 26083 27067 27967 28441 28703
  1985 1092  9565 15836 22169 25955 26180 27278 28185 28663 28927
  1986 1513  6445 11702 12935 15852 17649 18389 19001 19323 19501
  1987  557  4020 10946 12314 14428 16064 16738 17294 17587 17749
  1988 1351  6947 13112 16664 19525 21738 22650 23403 23800 24019
  1989 3133  5395  8759 11132 13043 14521 15130 15634 15898 16045
  1990 2063  6188 10046 12767 14959 16655 17353 17931 18234 18402
```

In Ben Zehnwirth and Glenn Branett pointed out in [**?**] that the age-to-age link ratios can be regarded as the slope coefficients of a weighted linear regression through the origin, see also [**?**].

```
R> lmCL <- function(i, Triangle){
+   lm(y~x+0, weights=1/Triangle[,i],
+      data=data.frame(x=Triangle[,i], y=Triangle[,i+1]))
+ }
R> sapply(lapply(c(1:(n-1)), lmCL, RAA), coef)

    x     x     x     x     x     x     x     x     x
2.999 1.624 1.271 1.172 1.113 1.042 1.033 1.017 1.009

R> demo(ChainLadder)
```

3.2.2 Mack chain-ladder

Following Mack [**?**] let C_{ik} denote the cumulative loss amounts of origin period (e.g. accident year) $i = 1, \dots, m$, 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: $E[F_{ik}|C_{i1}, C_{i2}, \dots, C_{ik}] = f_k$ with $F_{ik} = \dfrac{C_{i,k+1}}{C_{ik}}$ (2)

CL2: $Var(\dfrac{C_{i,k+1}}{C_{ik}}|C_{i1}, C_{i2}, \dots, C_{ik}) = \dfrac{\sigma_k^2}{w_{ik} C_{ik}^{\alpha}}$ (3)

CL3: $\{C_{i1}, \dots, C_{in}\}, \{C_{j1}, \dots, C_{jn}\}$, are independent for origin period $i \neq j$ (4)

with $w_{ik} \in [0;1], \alpha \in \{0, 1, 2\}$. If these assumptions are 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: `lm(y ~ x  + 0, 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.

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

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

     Latest Dev.To.Date Ultimate   IBNR Mack.S.E CV(IBNR)
1981 18,834       1.000   18,834      0        0      NaN
1982 16,704       0.991   16,858    154      206    1.339
1983 23,466       0.974   24,083    617      623    1.010
1984 27,067       0.943   28,703  1,636      747    0.457
1985 26,180       0.905   28,927  2,747    1,469    0.535
1986 15,852       0.813   19,501  3,649    2,002    0.549
1987 12,314       0.694   17,749  5,435    2,209    0.406
1988 13,112       0.546   24,019 10,907    5,358    0.491
1989  5,395       0.336   16,045 10,650    6,333    0.595
1990  2,063       0.112   18,402 16,339   24,566    1.503

              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
```

Access the loss development factors and the full triangle

```
R> mack$f
```

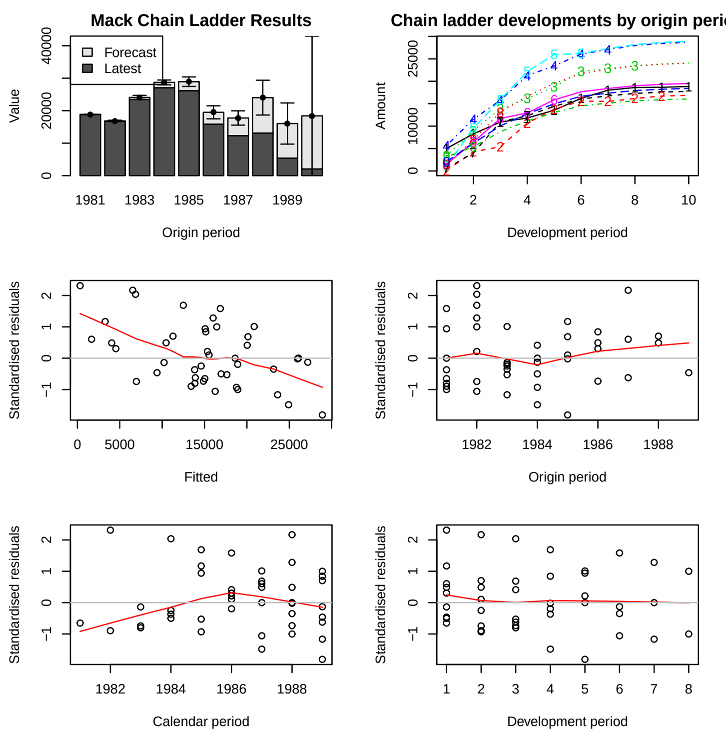
```
 [1] 2.999 1.624 1.271 1.172 1.113 1.042 1.033 1.017 1.009 1.000
```

```
R> mack$FullTriangle
```

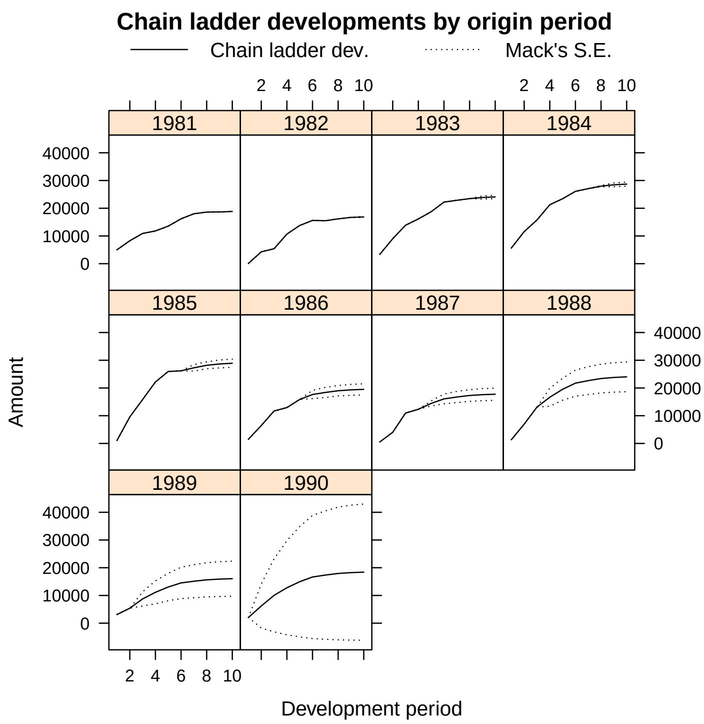
```
      dev
origin    1     2     3     4     5     6     7     8     9    10
  1981 5012  8269 10907 11805 13539 16181 18009 18608 18662 18834
  1982  106  4285  5396 10666 13782 15599 15496 16169 16704 16858
  1983 3410  8992 13873 16141 18735 22214 22863 23466 23863 24083
  1984 5655 11555 15766 21266 23425 26083 27067 27967 28441 28703
  1985 1092  9565 15836 22169 25955 26180 27278 28185 28663 28927
  1986 1513  6445 11702 12935 15852 17649 18389 19001 19323 19501
```

```
1987  557  4020 10946 12314 14428 16064 16738 17294 17587 17749
1988 1351  6947 13112 16664 19525 21738 22650 23403 23800 24019
1989 3133  5395  8759 11132 13043 14521 15130 15634 15898 16045
1990 2063  6188 10046 12767 14959 16655 17353 17931 18234 18402
```

```
R> plot(mack)
```




```
R> plot(mack, lattice=TRUE)
```



3.2.3 Bootstrap chain-ladder

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

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

     Latest Mean Ultimate Mean IBNR SD IBNR IBNR 75% IBNR 95%
1981 18,834        18,834         0       0        0        0
1982 16,704        16,871       167     712      162    1,473
1983 23,466        24,128       662   1,309    1,137    3,150
1984 27,067        28,751     1,684   1,890    2,611    5,234
1985 26,180        28,998     2,818   2,290    4,085    7,276
1986 15,852        19,584     3,732   2,557    5,197    8,655
1987 12,314        17,880     5,566   3,125    7,232   11,289
1988 13,112        24,101    10,989   4,946   13,808   20,408
1989  5,395        16,373    10,978   6,402   14,406   23,372
1990  2,063        20,096    18,033  14,037   26,103   43,671
```

```
                 Totals
Latest:         160,987
Mean Ultimate:  215,616
Mean IBNR:       54,629
SD IBNR:         18,839
Total IBNR 75%:  66,387
Total IBNR 95%:  87,517
```

```
R> plot(B)
R> # Compare to MackChainLadder
R> MackChainLadder(RAA)
```

```
MackChainLadder(Triangle = RAA)

     Latest Dev.To.Date Ultimate   IBNR Mack.S.E CV(IBNR)
1981 18,834       1.000   18,834      0        0      NaN
1982 16,704       0.991   16,858    154      143    0.928
1983 23,466       0.974   24,083    617      592    0.959
1984 27,067       0.943   28,703  1,636      713    0.436
1985 26,180       0.905   28,927  2,747    1,452    0.529
1986 15,852       0.813   19,501  3,649    1,995    0.547
1987 12,314       0.694   17,749  5,435    2,204    0.405
1988 13,112       0.546   24,019 10,907    5,354    0.491
1989  5,395       0.336   16,045 10,650    6,332    0.595
1990  2,063       0.112   18,402 16,339   24,566    1.503

               Totals
Latest:    160,987.00
Dev:             0.76
Ultimate:  213,122.23
IBNR:       52,135.23
Mack S.E.:  26,880.74
CV(IBNR):        0.52
```

```
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
1982      162     1473     2676       3736
1983     1137     3150     4881       5343
1984     2611     5234     7787       8750
1985     4085     7276     9035      10279
1986     5197     8655    11467      12828
1987     7232    11289    14998      17448
```

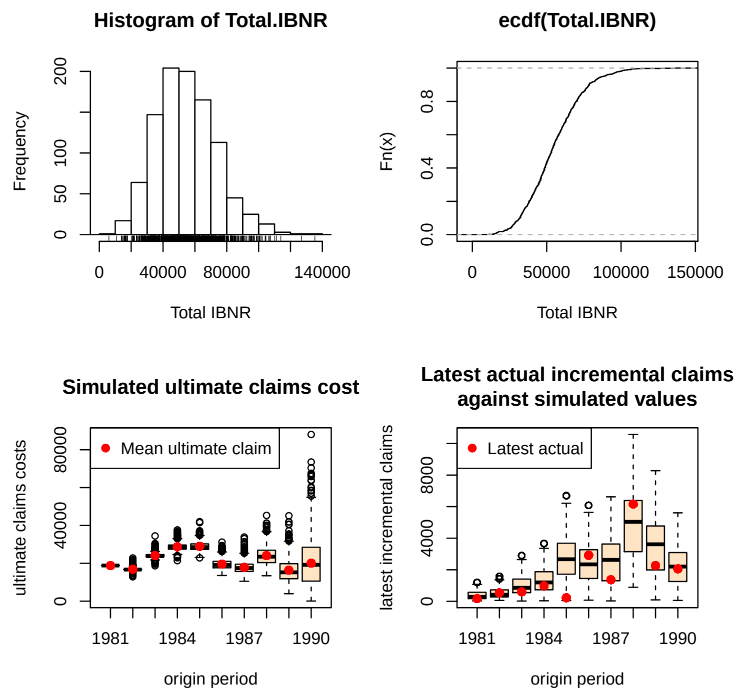
```
1988    13808    20408    24122      26579
1989    14406    23372    29622      31804
1990    26103    43671    58361      64509

$Totals
            Totals
IBNR 75%:    66387
IBNR 95%:    87517
IBNR 99%:   104153
IBNR 99.5%: 107375

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

    meanlog      sdlog
  10.843433    0.376148
 ( 0.011901) ( 0.008415)

R> curve(plnorm(x,fit$estimate["meanlog"], fit$estimate["sdlog"]), col="red", add=TRUE)
R>
```



3.2.4 Munich chain-ladder

```
R> MCLpaid
```

```
      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> MCLincurred
```

```
      dev
origin    1    2    3    4    5    6    7
     1  978 2104 2134 2144 2174 2182 2174
     2 1844 2552 2466 2480 2508 2454   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
```

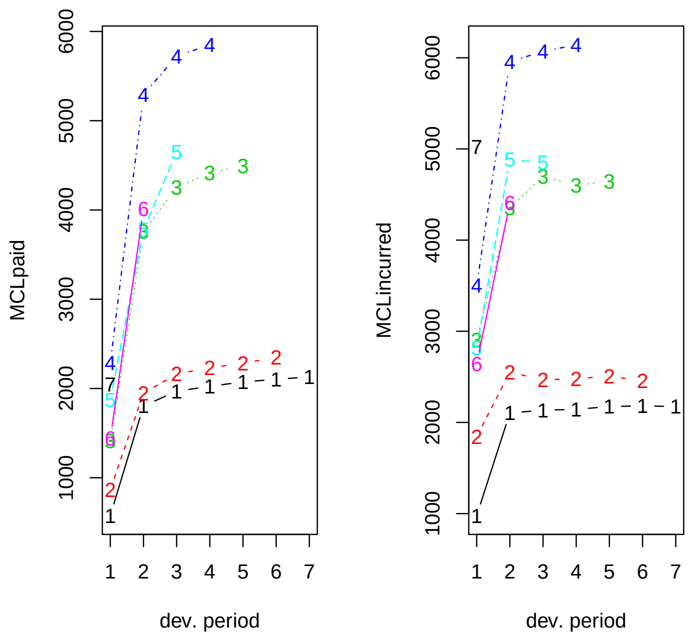
```
R> op <- par(mfrow=c(1,2))
R> plot(MCLpaid)
R> plot(MCLincurred)
R> par(op)
R> # Following the example in Quarg's (2004) paper:
R> MCL <- MunichChainLadder(MCLpaid, MCLincurred, est.sigmaP=0.1, est.sigmaI=0.1)
R> MCL
```

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

  Latest Paid Latest Incurred Latest P/I Ratio Ult. Paid Ult. Incurred
1       2,131           2,174            0.980     2,131         2,174
2       2,348           2,454            0.957     2,383         2,444
3       4,494           4,644            0.968     4,597         4,629
4       5,850           6,142            0.952     6,119         6,176
5       4,648           4,852            0.958     4,937         4,950
6       4,010           4,406            0.910     4,656         4,665
7       2,044           5,022            0.407     7,549         7,650
  Ult. P/I Ratio
1          0.980
2          0.975
3          0.993
4          0.991
5          0.997
6          0.998
7          0.987

Totals
            Paid Incurred P/I Ratio
Latest:   25,525   29,694      0.86
Ultimate: 32,371   32,688      0.99
```

```
R> plot(MCL)
```



3.3 Multivariate chain-ladder

3.4 Clark's methods

3.4.1 Clark's Cap Cod method

3.4.2 Clark's LDF method

3.5 Generalised linear model methods

4 Using `ChainLadder` with RExcel and SWord

The spreadsheet is located in the Excel folder of the package. The R command

```
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 [**?**]. 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")
```

5 Further resources

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

- Introduction to R for Actuaries [**?**].
- An Actuarial Toolkit [**?**].
- The book *Modern Actuarial Risk Theory – Using R* [**?**]
- Actuar package vignettes: http://cran.r-project.org/web/packages/actuar/index.html
- Mailing list R-SIG-insurance[6]: Special Interest Group on using R in actuarial science and insurance

5.1 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 *'Emperical 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`: Monte Carlo EM algorithms and Bayesian methods for fitting Tweedie compound Poisson linear models [**?**].
- `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 [**?**].
- `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 [**?**].

[6] https://stat.ethz.ch/mailman/listinfo/r-sig-insurance

- `actuar`: Loss distributions modelling, risk theory (including ruin theory), simulation of compound hierarchical models and credibility theory [**?**].
- `fitdistrplus`: Help to fit of a parametric distribution to non-censored or censored data [**?**].
- `mondate`: R packackge to keep track of dates in terms of months [**?**].
- `lifecontingencies`: Package to perform actuarial evaluation of life contingencies [**?**].

5.2 Presentations

Over the years the contributors of the `ChainLadder` package have given numerous presentations and most of those are still available online:

- Bayesian Hierarchical Models in Property-Casualty Insurance, Wayne Zhang, 2011
- ChainLadder at the Predictive Modelling Seminar, Institute of Actuaries, November 2010, Markus Gesmann, 2011
- Reserve variability calculations, CAS spring meeting, San Diego, Jimmy Curcio Jr., Markus Gesmann and Wayne Zhang, 2010
- The ChainLadder package, working with databases and MS Office interfaces, presentation at the "R you ready?" workshop , Institute of Actuaries, Markus Gesmann, 2009
- The ChainLadder package, London R user group meeting, Markus Gesmann, 2009
- Introduction to R, Loss Reserving with R, Stochastic Reserving and Modelling Seminar, Institute of Actuaries, Markus Gesmann, 2008
- Loss Reserving with R , CAS meeting, Vincent Goulet, Markus Gesmann and Daniel Murphy, 2008
- The ChainLadder package R-user conference Dortmund, Markus Gesmann, 2008

5.3 Further reading

Other papers and presentation which cited `ChainLadder` : [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**]

6 Training and consultancy

Please contact us if you would like to discuss tailored training or consultancy.