

Claims reserving with R: ChainLadder-0.1.5-3 Package Vignette DRAFT

Markus Gesmann* and Wayne Zhang[†]

April 13, 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

[†] actuary_zhang@hotmail.com

Contents

1 Introduction	4
1.1 Claims reserving in insurance	4
2 The ChainLadder package	4
2.1 Motivation	4
2.2 Brief package overview	5
2.3 Installation	6
3 Using the ChainLadder package	7
3.1 Working with triangles	7
3.1.1 Plotting triangles	7
3.1.2 Transforming triangles between cumulative and incremental representation	9
3.1.3 Importing triangles from external data sources	10
3.1.4 Coping and pasting from MS Excel	12
3.2 Chain-ladder methods	13
3.2.1 Basic idea	13
3.2.2 Mack chain-ladder	16
3.2.3 Bootstrap chain-ladder	19
3.2.4 Munich chain-ladder	22
3.3 Multivariate chain-ladder	24
3.4 Clark's methods	30
3.4.1 Clark's Cap Cod method	30
3.4.2 Clark's LDF method	30
3.5 Generalised linear model methods	30
4 Using ChainLadder with RExcel and SWord	35
5 Further resources	36
5.1 Other insurance related R packages	36
5.2 Presentations	37
5.3 Further reading	37

6 Training and consultancy	38
References	40

DRAFT

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 [GBB⁺09]. The cost for asbestos related claims in the US for the worldwide insurance industry was estimated to be around \$120bn in 2002 [Mic02].

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 [Sch11], [PR02] for an overview. Changes in regulatory requirements, e.g. Solvency II¹ 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 [GMZ12] 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.

¹See 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 reports²
- in combination with other tools such as L^AT_EX 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 functionality 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 [Mac93a], [Mac99], [QM04], [EV99]. Since version 0.1.3-3 it provides general multivariate chain ladder models by Wayne Zhang [Zha10]. 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 [Cla03]. Version 0.1.5-0 has added loss reserving models within the generalized linear model framework following a paper by England and Verrall [EV99] 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

²For 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[BN07].

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 [Tea12b]. The installation was successful if the command `library(ChainLadder)` gives you the following message:

```
R> library(ChainLadder)
```

```
ChainLadder version 0.1.5-3 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 or 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 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> plot(RAA)
```

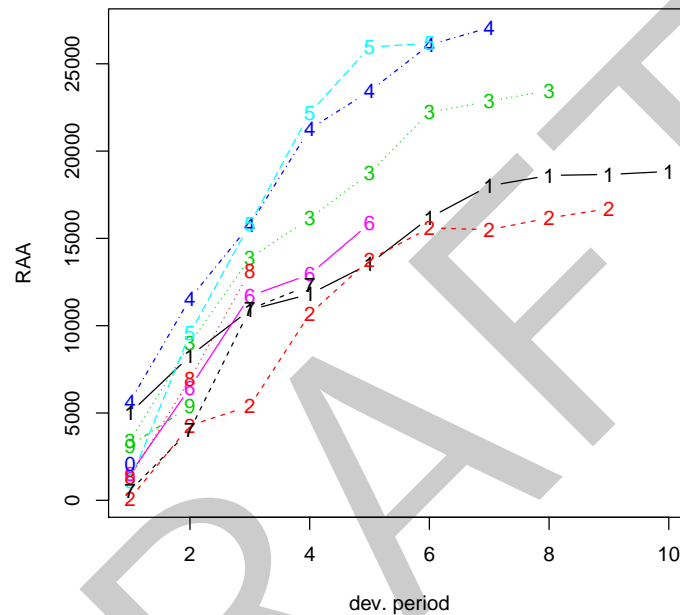


Figure 1: Claims development chart of the RAA triangle, with one line per origin period. Output of `plot(RAA)`

Setting the argument `lattice=TRUE` will produce individual plots for each origin period³, 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
```

³ChainLadder uses the `lattice` package for plotting the development of the origin years in separate panels.

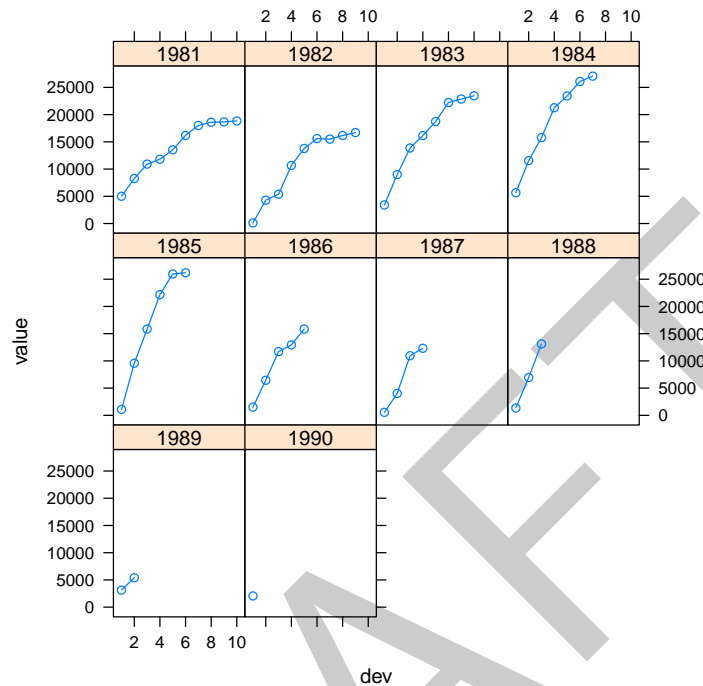


Figure 2: Claims development chart of the RAA triangle, with individual panels for each origin period. Output of `plot(RAA, lattice=TRUE)`

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 connection⁴ 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 [Tea12a].

In this section we use data stored in a CSV-file⁵ 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	GenerallLiab	:105
Median :	6	Median : 4.00	Median : 72468	M3IR5	:105

⁴See the RODBC package

⁵Please 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	1	2	3	4	5	6	7	8	9	10
origin											
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.

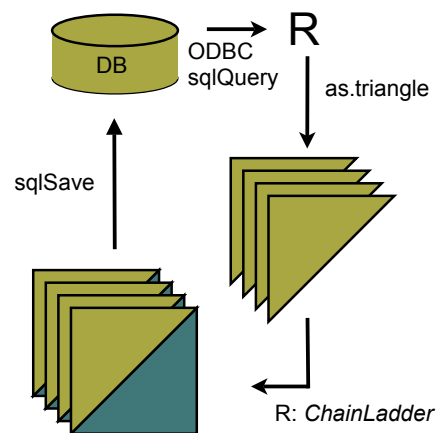


Figure 3: Flow chart of data between R and data bases.

3.1.4 Coping and pasting from MS Excel

Small data sets in Excel can be transferred 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 of a cumulative loss development triangle from one development period to the next $C_{ik}, i, k = 1, \dots, n$.

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

```
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
```

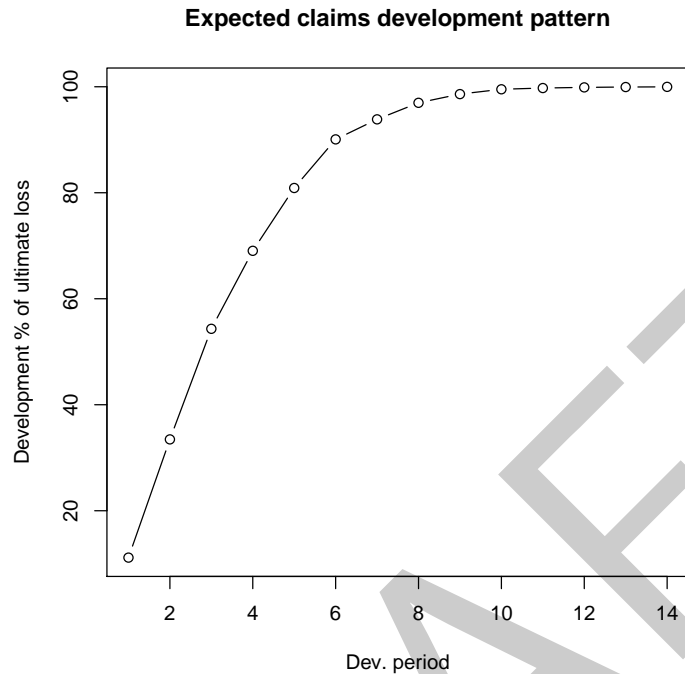
```
R> tail <- exp(co[1] + c((n + 1):(n + 100)) * co[2]) + 1
R> f.tail <- prod(tail)
R> f.tail
```

```
[1] 1.005
```



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

```
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")
```



The link ratios are then applied to the latest know cumulative claims amount to forecast the next development period.

```
R> f <- c(f, f.tail)
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> fullRAA[,n] <- fullRAA[,n]*f[n]
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	18928
1982	106	4285	5396	10666	13782	15599	15496	16169	16704	16942
1983	3410	8992	13873	16141	18735	22214	22863	23466	23863	24204
1984	5655	11555	15766	21266	23425	26083	27067	27967	28441	28847
1985	1092	9565	15836	22169	25955	26180	27278	28185	28663	29072
1986	1513	6445	11702	12935	15852	17649	18389	19001	19323	19599
1987	557	4020	10946	12314	14428	16064	16738	17294	17587	17838
1988	1351	6947	13112	16664	19525	21738	22650	23403	23800	24139
1989	3133	5395	8759	11132	13043	14521	15130	15634	15898	16125

1990 2063 6188 10046 12767 14959 16655 17353 17931 18234 18495

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

Since the early 1990s several papers have been published to embed the simple chain-ladder method into a statistical framework. Ben Zehnwirth and Glenn Branett point out in [Zx00] 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> 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
```

3.2.2 Mack chain-ladder

Thomas Mack published in 1993 [Mac93b] an article which allows to estimate the standard errors of the chain-ladder forecast without assuming a distribution under certain constrain to the data.

Following Mack [Mac99] 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:

$$\text{CL1: } E[F_{ik}|C_{i1}, C_{i2}, \dots, C_{ik}] = f_k \text{ with } F_{ik} = \frac{C_{i,k+1}}{C_{ik}} \quad (2)$$

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

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

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: $\text{lm}(y \sim x + 0, \text{weights}=w/x^{(2-\alpha)})$, where y is the vector of claims at development period $k + 1$ and x is the vector of claims at development period k .


```
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

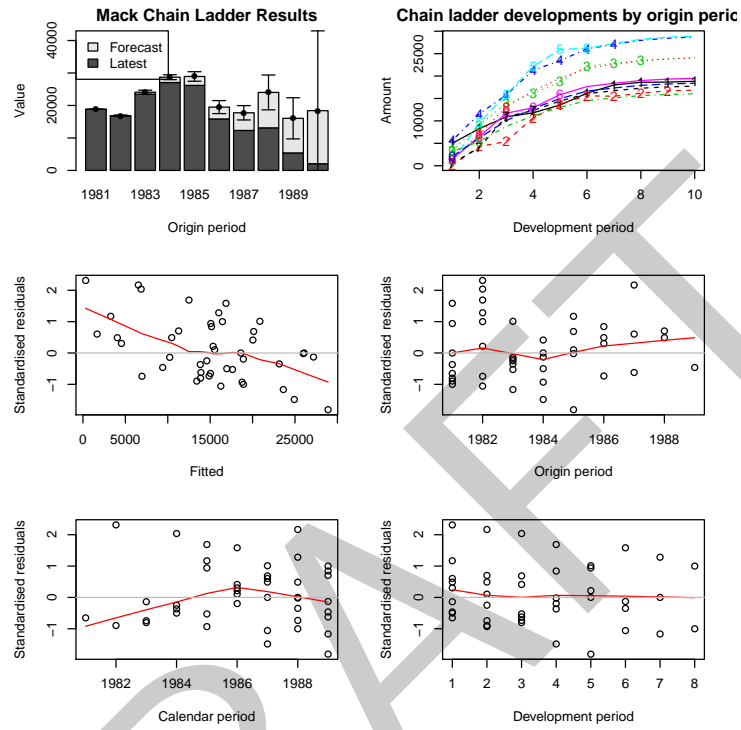
```
R> mack$f
```

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

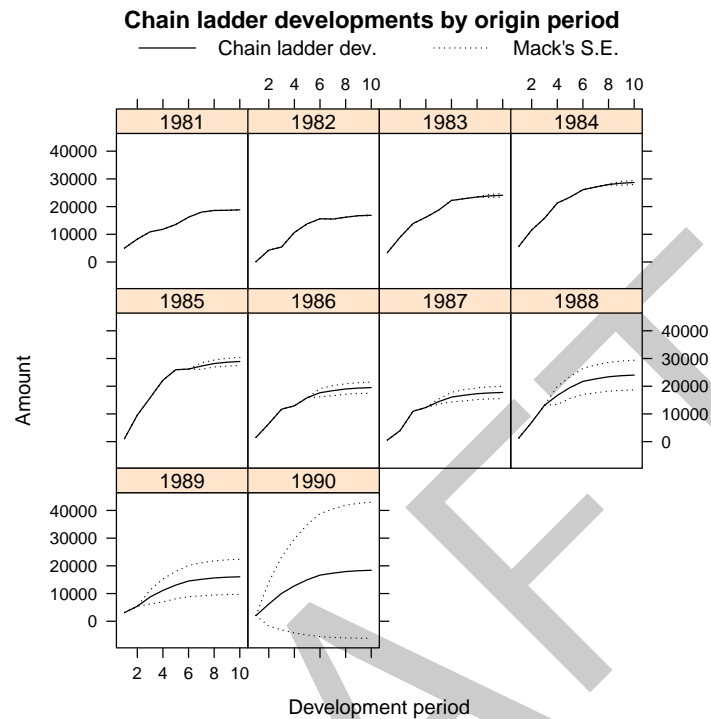
```
R> mack$FullTriangle
```

	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		16,820		116		662		129		1,254
1983	23,466		24,077		611		1,224		1,106		3,026
1984	27,067		28,737		1,670		1,935		2,662		5,067
1985	26,180		29,061		2,881		2,385		4,099		7,161
1986	15,852		19,573		3,721		2,567		5,102		8,587
1987	12,314		17,829		5,515		3,080		7,404		11,256
1988	13,112		24,462		11,350		5,194		14,370		21,283
1989	5,395		16,472		11,077		6,132		14,933		22,293
1990	2,063		19,707		17,644		13,159		25,222		41,327

```

Totals
Latest:      160,987
Mean Ultimate: 215,574
Mean IBNR:   54,587
SD IBNR:     18,395
Total IBNR 75%: 64,929
Total IBNR 95%: 87,655

```

```

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      0
1982    128.8    1254    2810    3481
1983   1105.6   3026   4600   5354
1984   2661.8   5067   7804   8905
1985   4098.9   7161  10165  11113
1986   5101.8   8587  11541  12452
1987   7404.0  11256  14952  16579

```

1988	14369.7	21283	26130	28868
1989	14932.9	22293	27964	29910
1990	25221.5	41327	56069	57605

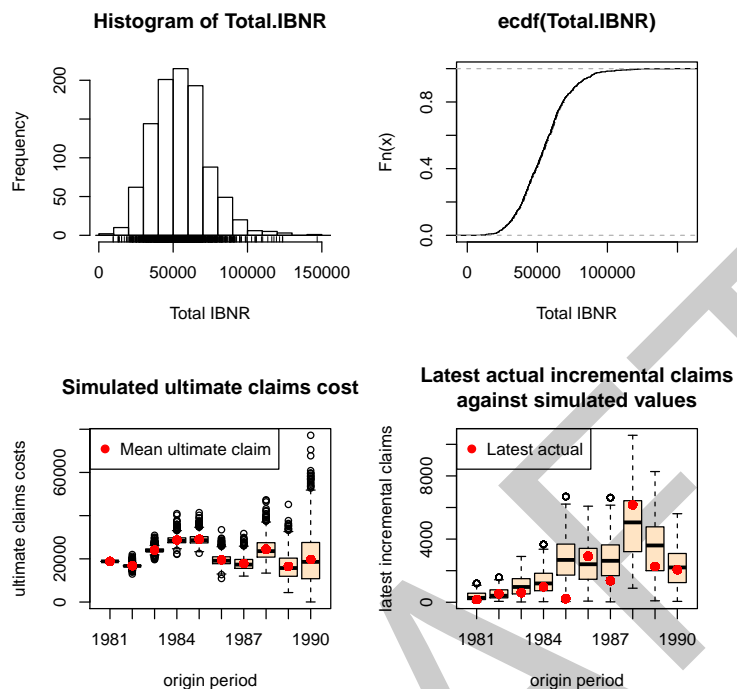
\$Totals

	Totals
IBNR 75%:	64929
IBNR 95%:	87655
IBNR 99%:	105968
IBNR 99.5%:	117800

```
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
10.847099	0.360747
(0.011414)	(0.008071)

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



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
```

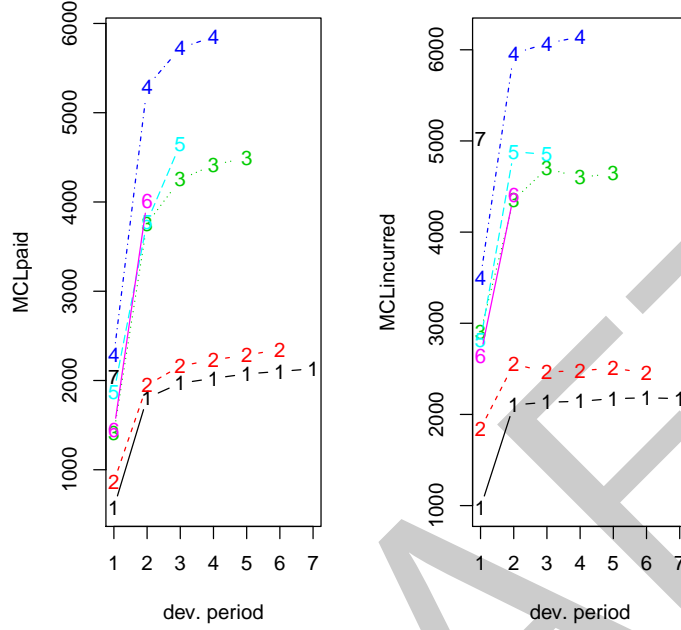
```
MunichChainLadder(Paid = MCLpaid, Incurred = MCLincurred, est.sigmaP = 0.1,
  est.sigmaI = 0.1)
```

	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

The Mack chain ladder technique can be generalized to the multivariate situation where multiple reserving triangles are modeled simultaneously. The advantage of the multivariate modeling is that correlations among different triangles can be modeled, which will lead to more accurate uncertainty assessment. There has been considerable development in this area, but most of the chain-ladder-based models can be summarized as sequential seemingly unrelated regressions (see Zhang 2010).

Denote $Y_{i,k} = (Y_{i,k}^{(1)}, \dots, Y_{i,k}^{(N)})$ 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. Zhang (2010) 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 a development matrix. Assump-

tions for this model are:

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

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

$$\text{losses of different accident years are independent.} \quad (8)$$

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

In the above, D is the diagonal operator, and δ is a known positive value that controls how the variance depends on the mean (as weights).

The simplest case where $A_k = 0$ and B_k are diagonal is a naive generalization of the chain ladder, often referred to as the multivariate chain ladder (see Pröhl and Schmidt 2005). The following shows an application of this case. The following is a data set used in Merz and Wüthrich (2008):

```
R> data(liab)
R> # this is a list of two triangles
R> length(liab)
```

```
[1] 2
```

```
R> dim(liab[[1]])
```

```
[1] 14 14
```

If we specify `fit.method = "OLS"`, the ordinary least squares will be used in estimating the development factors. As a result, the multivariate model is equivalent to running two Mack chain ladders separately.

```
R> fit1 <- MultiChainLadder(liab, fit.method = "OLS")
R> (s1 <- summary(fit1))
```

```
$`Summary Statistics for Triangle 1`
      Latest Dev.To.Date Ultimate      IBNR      S.E      CV
1      549,589      1.0000  549,589         0         0 0.0000
2      562,795      0.9966  564,740      1,945      1,743 0.8961
3      602,710      0.9911  608,104      5,394      7,354 1.3633
4      784,632      0.9867  795,248     10,616      9,042 0.8518
5      768,373      0.9806  783,593     15,220     11,181 0.7346
6      811,100      0.9690  837,088     25,988     16,781 0.6457
7      896,728      0.9551  938,861     42,133     19,690 0.4673
8     1,022,241      0.9308 1,098,200     75,959     23,344 0.3073
9     1,019,303      0.8826 1,154,902    135,599     29,585 0.2182
10    1,141,750      0.7976 1,431,409    289,659     37,492 0.1294
```

11	1,174,196	0.6766	1,735,433	561,237	57,623	0.1027
12	1,032,684	0.4998	2,065,991	1,033,307	89,488	0.0866
13	772,971	0.2905	2,660,561	1,887,590	193,210	0.1024
14	204,325	0.0898	2,274,941	2,070,616	282,960	0.1367
Total	11,343,397	0.6482	17,498,658	6,155,261	427,289	0.0694

\$`Summary Statistics for Triangle 2`

	Latest	Dev.To.Date	Ultimate	IBNR	S.E	CV
1	391,428	1.000	391,428	0	0	0.0000
2	483,974	1.000	483,839	-135	604	-4.4825
3	540,742	1.001	540,002	-740	1,436	-1.9416
4	485,016	0.998	486,227	1,211	2,912	2.4043
5	507,752	0.998	508,744	992	3,202	3.2284
6	549,693	0.994	552,825	3,132	5,418	1.7298
7	635,452	0.994	639,113	3,661	6,221	1.6993
8	648,365	0.985	658,410	10,045	7,483	0.7449
9	663,152	0.969	684,719	21,567	9,123	0.4230
10	790,901	0.935	845,543	54,642	16,191	0.2963
11	844,159	0.877	962,734	118,575	26,742	0.2255
12	915,109	0.783	1,169,260	254,151	36,736	0.1445
13	909,066	0.617	1,474,514	565,448	53,398	0.0944
14	394,997	0.277	1,426,060	1,031,063	126,613	0.1228
Total	8,759,806	0.809	10,823,418	2,063,612	162,872	0.0789

\$`Summary Statistics for Triangle 1+2`

	Latest	Dev.To.Date	Ultimate	IBNR	S.E	CV
1	941,017	1.000	941,017	0	0	0.0000
2	1,046,769	0.998	1,048,579	1,810	1,845	1.0190
3	1,143,452	0.996	1,148,107	4,655	7,493	1.6097
4	1,269,648	0.991	1,281,475	11,827	9,499	0.8032
5	1,276,125	0.988	1,292,337	16,212	11,631	0.7174
6	1,360,793	0.979	1,389,913	29,120	17,634	0.6056
7	1,532,180	0.971	1,577,973	45,793	20,650	0.4509
8	1,670,606	0.951	1,756,610	86,004	24,514	0.2850
9	1,682,455	0.915	1,839,620	157,165	30,960	0.1970
10	1,932,651	0.849	2,276,952	344,301	40,838	0.1186
11	2,018,355	0.748	2,698,167	679,812	63,526	0.0934
12	1,947,793	0.602	3,235,251	1,287,458	96,735	0.0751
13	1,682,037	0.407	4,135,075	2,453,038	200,453	0.0817
14	599,322	0.162	3,701,001	3,101,679	309,995	0.0999
Total	20,103,203	0.710	28,322,077	8,218,874	457,278	0.0556

By default, the summary 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 behavior 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:

```
R> MackChainLadder(liab[[1]], est.sigma = "Mack")
```

```
MackChainLadder(Triangle = liab[[1]], est.sigma = "Mack")
```

	Latest	Dev.To.Date	Ultimate	IBNR	Mack.S.E	CV(IBNR)
1	549,589	1.0000	549,589	0	0	NaN
2	562,795	0.9966	564,740	1,945	1,743	0.8961
3	602,710	0.9911	608,104	5,394	7,354	1.3633
4	784,632	0.9867	795,248	10,616	9,042	0.8518
5	768,373	0.9806	783,593	15,220	11,181	0.7346
6	811,100	0.9690	837,088	25,988	16,781	0.6457
7	896,728	0.9551	938,861	42,133	19,690	0.4673
8	1,022,241	0.9308	1,098,200	75,959	23,344	0.3073
9	1,019,303	0.8826	1,154,902	135,599	29,585	0.2182
10	1,141,750	0.7976	1,431,409	289,659	37,492	0.1294
11	1,174,196	0.6766	1,735,433	561,237	57,623	0.1027
12	1,032,684	0.4998	2,065,991	1,033,307	89,488	0.0866
13	772,971	0.2905	2,660,561	1,887,590	193,210	0.1024
14	204,325	0.0898	2,274,941	2,070,616	282,960	0.1367

```
Totals
Latest: 11,343,397.00
Dev: 0.65
Ultimate: 17,498,658.29
IBNR: 6,155,261.29
Mack S.E.: 427,288.99
CV(IBNR): 0.07
```

The `model.summary` slot in the summary stores the fitted parameters in the regression fit for each development period:

```
R> names(s1)
```

```
[1] "Triangles"      "FullTriangles"  "S.E.Full"       "S.E.Est.Full"
[5] "S.E.Proc.Full"  "Ultimate"       "IBNR"           "S.E.Ult"
[9] "S.E.Est.Ult"    "S.E.Proc.Ult"   "report.summary" "coefficients"
[13] "coefCov"        "residCov"       "rstandard"      "fitted.values"
[17] "residCor"       "model.summary"  "portfolio"
```

```
R> head(s1$model.summary)
```

```
      Estimate Std. Error t value Pr(>|t|) dev
eq1_x[[1]]    3.235    0.113591  28.48 2.182e-12  1
```

eq2_x[[2]]	2.226	0.065316	34.08	2.593e-13	1
eq1_x[[1]]	1.720	0.043880	39.21	3.599e-13	2
eq2_x[[2]]	1.269	0.011148	113.87	0.000e+00	2
eq1_x[[1]]	1.354	0.016523	81.92	1.776e-15	3
eq2_x[[2]]	1.120	0.007803	143.59	0.000e+00	3

To allow correlations to be incorporated, we employ the seemingly unrelated regressions that simultaneously model the two triangles. We now specify `fit.method = "SUR"`:

```
R> fit2 <- MultiChainLadder(liab, fit.method = "SUR")
R> (s2 <- summary(fit2))
```

```
$`Summary Statistics for Triangle 1`
      Latest Dev.To.Date Ultimate      IBNR      S.E      CV
1      549,589      1.0000   549,589         0         0 0.0000
2      562,795      0.9966   564,740      1,945     1,743 0.8961
3      602,710      0.9913   608,013      5,303     6,720 1.2673
4      784,632      0.9868   795,128     10,496     8,154 0.7769
5      768,373      0.9805   783,649     15,276    10,497 0.6871
6      811,100      0.9688   837,214     26,114    16,269 0.6230
7      896,728      0.9544   939,595     42,867    19,154 0.4468
8     1,022,241      0.9301  1,099,087     76,846    22,726 0.2957
9     1,019,303      0.8818  1,155,890    136,587    29,055 0.2127
10    1,141,750      0.7970  1,432,561    290,811    36,849 0.1267
11    1,174,196      0.6761  1,736,765    562,569    57,024 0.1014
12    1,032,684      0.4999  2,065,835   1,033,151    88,941 0.0861
13      772,971      0.2907  2,658,834   1,885,863   192,733 0.1022
14      204,325      0.0901  2,268,006   2,063,681   282,477 0.1369
Total 11,343,397      0.6484 17,494,907  6,151,510   419,293 0.0682
```

```
$`Summary Statistics for Triangle 2`
      Latest Dev.To.Date Ultimate      IBNR      S.E      CV
1      391,428      1.000   391,428         0         0 0.0000
2      483,974      1.000   483,839      -135         604 -4.4846
3      540,742      1.001   540,020      -722     1,324 -1.8323
4      485,016      0.998   486,242     1,226     2,868  2.3394
5      507,752      0.998   508,744         992     3,158  3.1844
6      549,693      0.994   552,877     3,184     5,388  1.6923
7      635,452      0.994   639,272     3,820     6,187  1.6193
8      648,365      0.985   658,591    10,226     7,454  0.7289
9      663,152      0.968   684,957    21,805     9,097  0.4172
10     790,901      0.935   846,012    55,111    16,173  0.2935
11     844,159      0.876   963,052   118,893    26,734  0.2249
12     915,109      0.783  1,169,300   254,191    36,722  0.1445
```

13	909,066	0.617	1,473,826	564,760	53,370	0.0945
14	394,997	0.278	1,423,182	1,028,185	126,538	0.1231
Total	8,759,806	0.809	10,821,341	2,061,535	162,464	0.0788

\$`Summary Statistics for Triangle 1+2`

	Latest	Dev.To.Date	Ultimate	IBNR	S.E	CV
1	941,017	1.000	941,017	0	0	0.0000
2	1,046,769	0.998	1,048,579	1,810	1,851	1.0221
3	1,143,452	0.996	1,148,032	4,580	7,859	1.7158
4	1,269,648	0.991	1,281,370	11,722	9,545	0.8143
5	1,276,125	0.987	1,292,393	16,268	12,133	0.7458
6	1,360,793	0.979	1,390,091	29,298	18,913	0.6455
7	1,532,180	0.970	1,578,868	46,688	22,448	0.4808
8	1,670,606	0.951	1,757,679	87,073	25,913	0.2976
9	1,682,455	0.914	1,840,846	158,391	33,294	0.2102
10	1,932,651	0.848	2,278,572	345,921	45,253	0.1308
11	2,018,355	0.748	2,699,816	681,461	72,050	0.1057
12	1,947,793	0.602	3,235,135	1,287,342	112,187	0.0871
13	1,682,037	0.407	4,132,660	2,450,623	222,927	0.0910
14	599,322	0.162	3,691,189	3,091,867	342,127	0.1107
Total	20,103,203	0.710	28,316,248	8,213,045	500,607	0.0610

We see that the portfolio prediction error is inflated to 500,607 from 457,278 due to the positive correlations between the two triangles. The estimated correlation for each development period can be retrieved through the `residCor` function:

```
R> unlist(residCor(fit2))
```

```
[1] 0.247283 0.495379 0.682483 0.446450 0.486991 0.450551 -0.171639
[8] 0.804882 0.336744 0.687983 -0.003797 1.000000 0.020989
```

Similarly, most methods that work for linear models such as `coef`, `fitted`, `resid` and so on will also work. For example, we can retrieve the estimated development factors for the first two periods as

```
R> coef(fit2)[1:2]
```

```
[[1]]
eq1_x[[1]] eq2_x[[2]]
  3.227    2.222
```

```
[[2]]
eq1_x[[1]] eq2_x[[2]]
  1.719    1.269
```

The plot function can also be used to generate various summary and diagnostic figures:

We use the `which.triangle` to suppress the plot for the portfolios, and use the `which.plot` to select the desired types of plots. See the documentation for possible values for these two arguments.

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

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 modeling 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 `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 method 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.

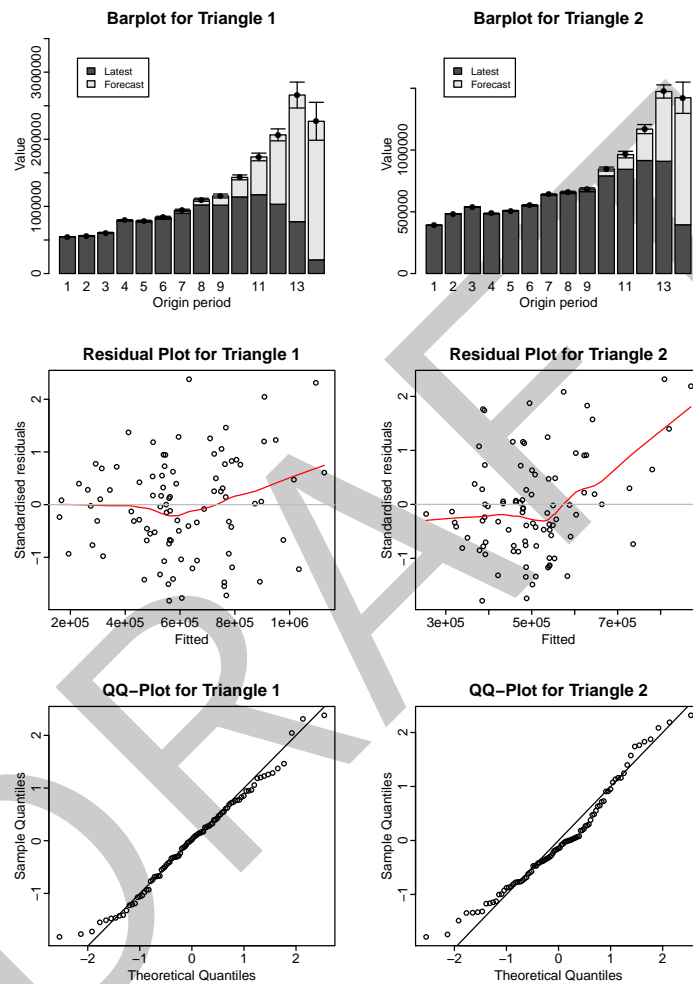
For example, the following fits the over-dispersed Poisson model and spells out the estimated reserve information:

```
R> # load data
R> data(GenIns)
```

```

R> parold <- par(mfrow = c(4, 2), mar = c(4, 4, 2, 1),
+               mgp = c(1.3, 0.3, 0), tck = -0.02)
R> plot(fit2, which.triangle = 1:2, which.plot = c(1, 3, 4), legend.cex = 0.8)
R> par(parold)

```



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

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

We can also extract the underlying GLM model by specify 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

(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))

	Latest	Dev.To.Date	Ultimate	IBNR	S.E	CV
2	5339	0.98288	5432	93	45.17	0.4857
3	4909	0.91655	5356	447	160.56	0.3592
4	4588	0.88248	5199	611	177.62	0.2907
5	3873	0.79611	4865	992	254.47	0.2565
6	3692	0.71757	5145	1453	351.33	0.2418
7	3483	0.61440	5669	2186	526.29	0.2408
8	2864	0.43870	6529	3665	941.32	0.2568
9	1363	0.24854	5485	4122	1175.95	0.2853
10	344	0.07078	4860	4516	1667.39	0.3692
total	30457	0.62742	48543	18086	2702.71	0.1494

R> # compound Poisson GLM (variance function estimated from the data):

R> (fit3 <- glmReserve(GenIns, var.power = NULL))

	Latest	Dev.To.Date	Ultimate	IBNR	S.E	CV
2	5339	0.98270	5433	94	91.6	0.9745
3	4909	0.91331	5375	466	186.5	0.4003
4	4588	0.86780	5287	699	223.7	0.3201
5	3873	0.79709	4859	986	264.8	0.2685
6	3692	0.72164	5116	1424	333.2	0.2340
7	3483	0.61505	5663	2180	452.9	0.2078
8	2864	0.42365	6761	3897	754.6	0.1936
9	1363	0.24231	5626	4263	1019.5	0.2391
10	344	0.06943	4955	4611	1911.0	0.4144
total	30457	0.62058	49078	18621	2831.5	0.1521

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"))
```

	Latest	Dev.To.Date	Ultimate	IBNR	S.E	CV
2	5339	0.98252	5434	95	105.4	1.1098
3	4909	0.91263	5379	470	216.1	0.4597
4	4588	0.86599	5298	710	266.6	0.3755
5	3873	0.79725	4858	985	307.5	0.3122
6	3692	0.72235	5111	1419	376.3	0.2652
7	3483	0.61527	5661	2178	496.1	0.2278
8	2864	0.42221	6784	3920	812.9	0.2074
9	1363	0.24162	5642	4279	1050.9	0.2456
10	344	0.06922	4970	4626	2004.1	0.4332
total	30457	0.61982	49138	18681	2959.4	0.1584

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.

```
R> names(fit5)

[1] "call"          "summary"        "Triangle"
[4] "FullTriangle"  "model"          "sims.par"
[7] "sims.reserve.mean" "sims.reserve.pred"
```

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

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

	2.5%	25%	50%	75%	97.5%
2	0	34	82	170	376
3	136	337	470	615	987
4	279	556	719	917	1,302
5	506	797	972	1,197	1,674
6	774	1,159	1,404	1,666	2,203

```

7  1,329 1,877 2,210 2,547 3,303
8  2,523 3,463 3,991 4,572 5,713
9  2,364 3,593 4,310 5,013 6,531
10 913 3,354 4,487 5,774 9,165

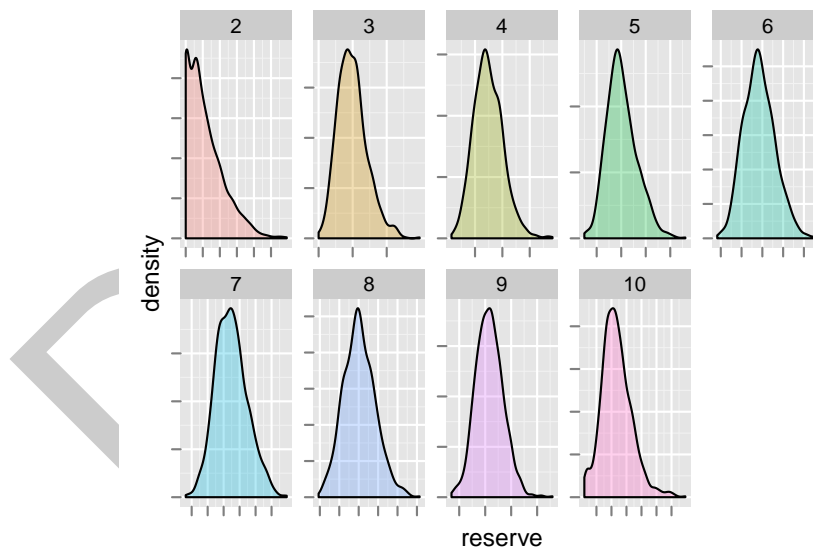
```

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

```

R> library(ggplot2)
R> library(reshape2)
R> prm <- melt(pr)
R> names(prm) <- c("year", "reserve")
R> gg <- ggplot(prm, aes(reserve))
R> gg <- gg + geom_density(aes(fill = year), alpha = 0.3) +
+   facet_wrap(~year, nrow = 2, scales = "free") +
+   opts(axis.text.x = theme_blank(),
+         axis.text.y = theme_blank(),
+         legend.position = "none")
R> print(gg)

```



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 [BN07]. 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")
```

5 Further resources

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

- Introduction to R for Actuaries [DS06].
- An Actuarial Toolkit [MSH⁺06].
- The book *Modern Actuarial Risk Theory – Using R* [KGDD01]
- Actuar package vignettes: <http://cran.r-project.org/web/packages/actuar/index.html>
- Mailing list [R-SIG-insurance](https://stat.ethz.ch/mailman/listinfo/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.

- `cp1m`: Monte Carlo EM algorithms and Bayesian methods for fitting Tweedie compound Poisson linear models [Zha11].
- `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].

⁶<https://stat.ethz.ch/mailman/listinfo/r-sig-insurance>

- `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].
- `fitdistrplus`: Help to fit of a parametric distribution to non-censored or censored data [DMPDD10].
- `mondate`: R packackge to keep track of dates in terms of months [Mur11].
- `lifecontingencies`: Package to perform actuarial evaluation of life contingencies [Spe11].

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 : [Orr07], [Nic09], [Zha10], [MNNV10], [Sch10], [MNV10], [Esc11], [Spe11]

6 Training and consultancy

Please contact [us](#) if you would like to discuss tailored training or consultancy.

References

- [BN07] Thomas Baier and Erich Neuwirth. Excel :: Com :: R. *Computational Statistics*, 22(1), April 2007. Physica Verlag.
- [Cla03] David R. Clark. *LDF Curve-Fitting and Stochastic Reserving: A Maximum Likelihood Approach*. Casualty Actuarial Society, 2003. CAS Fall Forum.
- [DGP08] C Dutang, V. Goulet, and M. Pigeon. actuar: An R package for actuarial science. *Journal of Statistical Software*, 25(7), 2008.
- [DMPDD10] Marie Laure Delignette-Muller, Regis Pouillot, Jean-Baptiste Denis, and Christophe Dutang. *fitdistrplus: help to fit of a parametric distribution to non-censored or censored data*, 2010. R package version 0.1-3.
- [DS06] Nigel De Silva. An introduction to r: Examples for actuaries. <http://toolkit.pbwiki.com/RToolkit>, 2006.
- [Esc11] Benedict Escoto. *favir: Formatted Actuarial Vignettes in R*, 0.5-1 edition, January 2011.
- [EV99] Peter England and Richard Verrall. Analytic and bootstrap estimates of prediction errors in claims reserving. *Mathematics and Economics*, Vol. 25:281 – 293, 1999.
- [GBB⁺09] Brian Gravelsons, Matthew Ball, Dan Beard, Robert Brooks, Naomi Couchman, Brian Gravelsons, Charlie Kefford, Darren Michaels, Patrick Nolan, Gregory Overton, Stephen Robertson-Dunn, Emiliano Ruffini, Graham Sandhouse, Jerome Schilling, Dan Sykes, Peter Taylor, Andy Whiting, Matthew Wilde, and John Wilson. B12: Uk asbestos working party update 2009. <http://www.actuaries.org.uk/research-and-resources/documents/b12-uk-asbestos-working-party-update-2009-5mb>, October 2009. Presented at the General Insurance Convention.
- [GMZ12] Markus Gesmann, Dan Murphy, and Wayne Zhang. *ChainLadder: Mack-, Bootstrap and Munich-chain-ladder methods for insurance claims reserving*, 2012. R package version 0.1.5-2.
- [KGDD01] R. Kaas, M. Goovaerts, J. Dhaene, and M. Denuit. *Modern actuarial risk theory*. Kluwer Academic Publishers, Dordrecht, 2001.

- [LS11] Christopher W. Laws and Frank A. Schmid. *lossDev: Robust Loss Development Using MCMC*, 2011. R package version 3.0.0-1.
- [Mac93a] Thomas Mack. Distribution-free calculation of the standard error of chain ladder reserve estimates. *Astin Bulletin*, Vol. 23:213 – 25, 1993.
- [Mac93b] Thomas Mack. Distribution-free calculation of the standard error of chain ladder reserve estimates. *ASTIN Bulletin*, 23:213–225, 1993.
- [Mac99] Thomas Mack. The standard error of chain ladder reserve estimates: Recursive calculation and inclusion of a tail factor. *Astin Bulletin*, Vol. 29(2):361 – 266, 1999.
- [Mic02] Darren Michaels. APH: how the love carnal and silicone implants nearly destroyed Lloyd's (slides). <http://www.actuaries.org.uk/research-and-resources/documents/aph-how-love-carnal-and-silicone-implants-nearly-destroyed-lloyds-s>, December 2002. Presented at the Younger Members' Convention.
- [MNNV10] Maria Dolores Martinez Miranda, Bent Nielsen, Jens Perch Nielsen, and Richard Verrall. *Cash flow simulation for a model of outstanding liabilities based on claim amounts and claim numbers*. CASS, September 2010.
- [MNV10] Maria Dolores Martinez Miranda, Jens Perch Nielsen, and Richard Verrall. *Double Chain Ladder*. ASTIN, Colloquia Madrid edition, 2010.
- [MSH⁺06] Trevor Maynard, Nigel De Silva, Richard Holloway, Markus Gesmann, Sie Lau, and John Harnett. An actuarial toolkit. introducing The Toolkit Manifesto. <http://www.actuaries.org.uk/sites/all/files/documents/pdf/actuarial-toolkit.pdf>, 2006. General Insurance Convention.
- [Mur94] Daniel Murphy. Unbiased loss development factors. *PCAS*, 81:154 – 222, 1994.
- [Mur11] Dan Murphy. *mondate: Keep track of dates in terms of months*, 2011. R package version 0.9.8.24.
- [Nic09] Luke Nichols. *Multimodel Inference for Reserving*. Australian Prudential Regulation Authority (APRA), December 2009.
- [Orr07] James Orr. *A Simple Multi-State Reserving Model*. ASTIN, Colloquia Orlando edition, 2007.
- [PR02] P.D.England and R.J.Verrall. Stochastic claims reserving in general insurance. *British Actuarial Journal*, 8:443–544, 2002.

- [QM04] Gerhard Quarg and Thomas Mack. Munich chain ladder. Munich Re Group, 2004.
- [Sch10] Ernesto Schirmacher. Reserve variability calculations, chain ladder, R, and Excel. <http://www.casact.org/affiliates/cane/0910/schirmacher.pdf>, September 2010. Presentation at the Casualty Actuaries of New England (CANE) meeting.
- [Sch11] Klaus D. Schmidt. A bibliography on loss reserving. <http://www.math.tu-dresden.de/sto/schmidt/dsvm/reserve.pdf>, 2011.
- [Spe11] Giorgio Alfredo Spedicato. *Introduction to lifecontingencies Package*. StatisticalAdvisor Inc, 0.0.4 edition, November 2011.
- [Tea12a] R Development Core Team. *R Data Import/Export*. R Foundation for Statistical Computing, 2012. ISBN 3-900051-10-0.
- [Tea12b] R Development Core Team. *R Installation and Administration*. R Foundation for Statistical Computing, 2012. ISBN 3-900051-09-7.
- [Zha10] Y. Zhang. A general multivariate chain ladder model. *Insurance: Mathematics and Economics*, 46:588 – 599, 2010.
- [Zha11] Wayne Zhang. *cplm: Tweedie compound Poisson linear models*, 2011. R package version 0.4-1.
- [Zx00] Ben Zehnwirth and Glenn xbBarnett. Best estimates for reserves. *Proceedings of the CAS*, LXXXVII(167), November 2000.