

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

Markus Gesmann* and Wayne Zhang†

July 16, 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	5
3	Using the ChainLadder package	6
3.1	Working with triangles	6
3.1.1	Plotting triangles	7
3.1.2	Transforming triangles between cumulative and incremental representation	7
3.1.3	Importing triangles from external data sources	9
3.1.4	Coping and pasting from MS Excel	12
3.2	Chain-ladder methods	12
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
3.3.1	The "triangles" class	25
3.3.2	Separate chain ladder ignoring correlations	26
3.3.3	Multivariate chain ladder using seemingly unrelated regressions	28
3.3.4	Other residual covariance estimation methods	29
3.3.5	Model with intercepts	32
3.3.6	Joint modeling of the paid and incurred losses	34
3.4	Clark's methods	35
3.4.1	Clark's Cap Cod method	35
3.4.2	Clark's LDF method	35
3.5	Generalised linear model methods	35

4 Using ChainLadder with RExcel and SWord	39
5 Further resources	40
5.1 Other insurance related R packages	40
5.2 Presentations	41
5.3 Further reading	41
6 Training and consultancy	41
References	43

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 judgment 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 estimate 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.

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

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

- 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 \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 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

```
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')
```

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

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)
```

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
```

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:

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

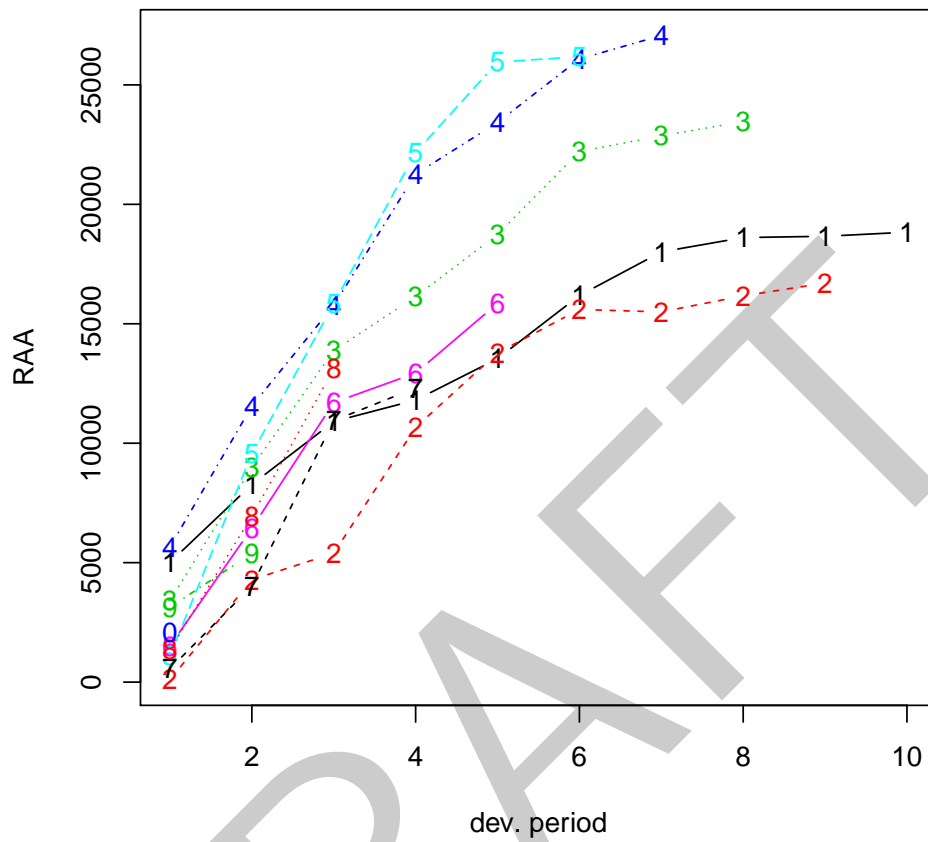


Figure 1: Claims development chart of the RAA triangle, with one line per origin period. Output of `plot(RAA)`

```
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
raa.inc[1,]	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,]
```

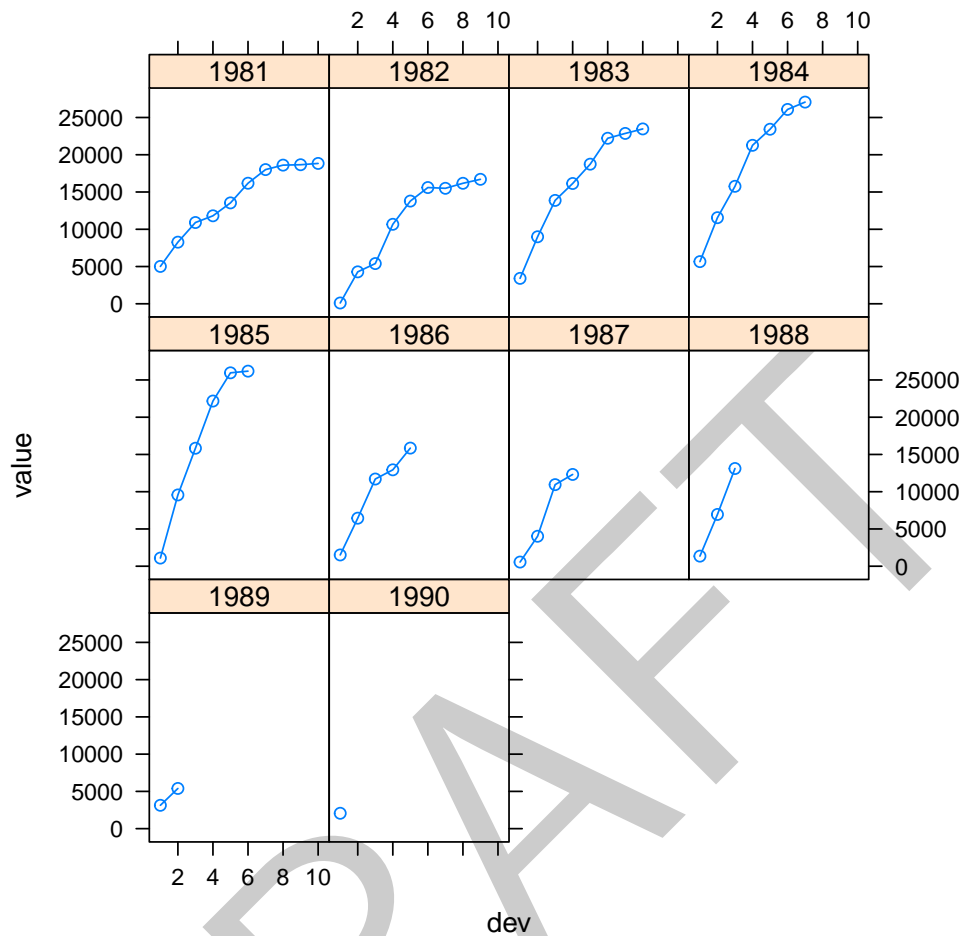



Figure 2: Claims development chart of the RAA triangle, with individual panels for each origin period. Output of `plot(RAA, lattice=TRUE)`

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:

⁴See the `RODBC` package

```
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	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
```

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.

```
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.

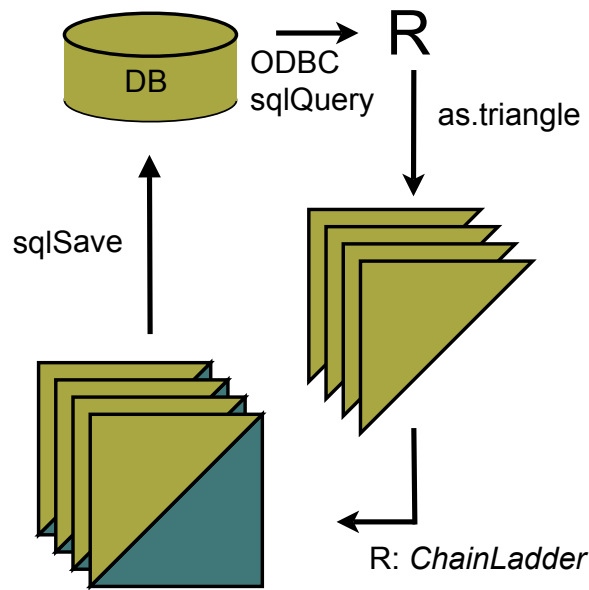


Figure 3: Flow chart of data between R and data bases.

3.1.4 Coping and pasting from MS Excel

Small data sets in Excel can be 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
```

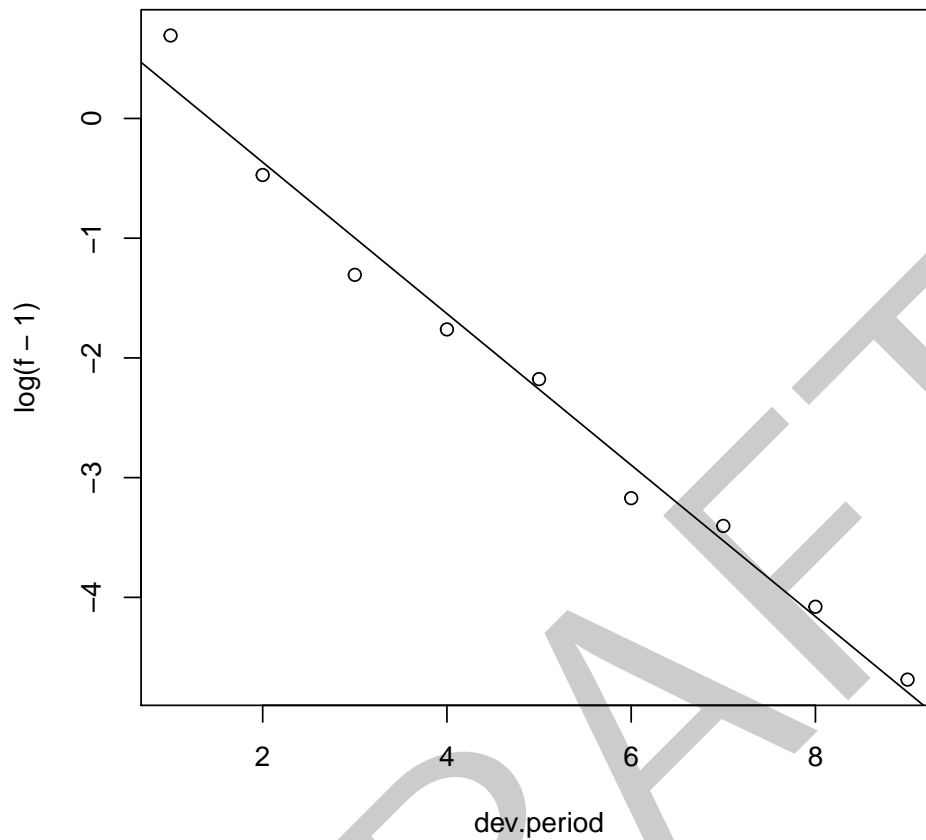
```
[1] 2.999 1.624 1.271 1.172 1.113 1.042 1.033 1.017 1.009
```

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
```

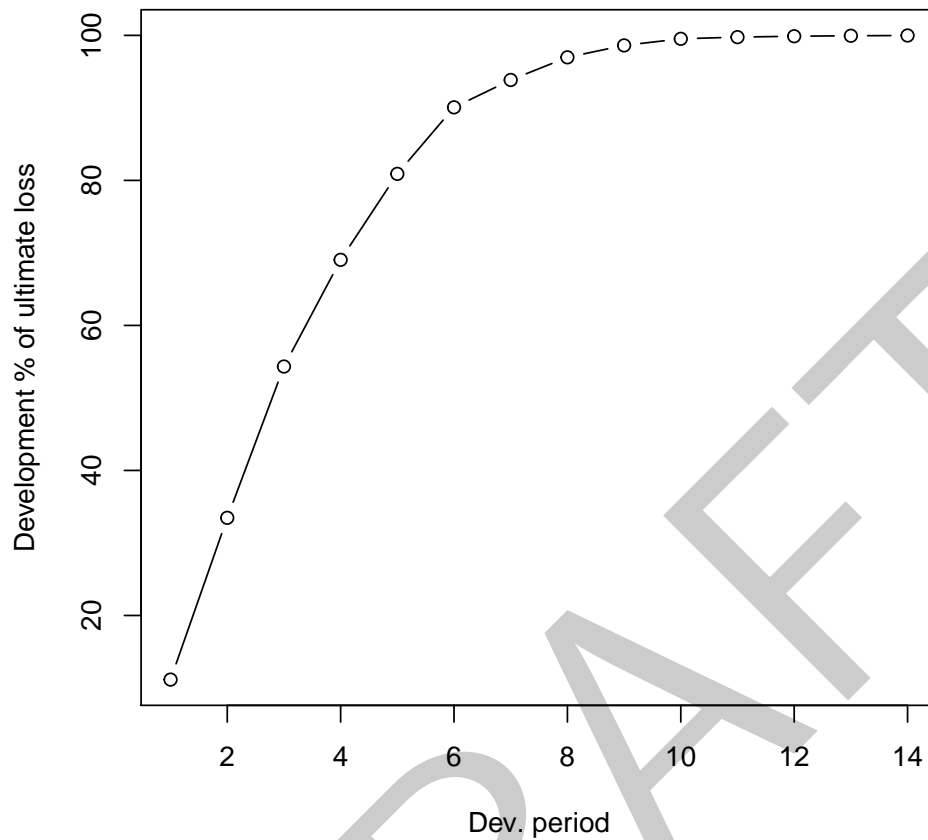
Log-linear extrapolation of age-to-age factors



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")
```

Expected claims development pattern



The link ratios are then applied to the latest known 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 Barnett 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

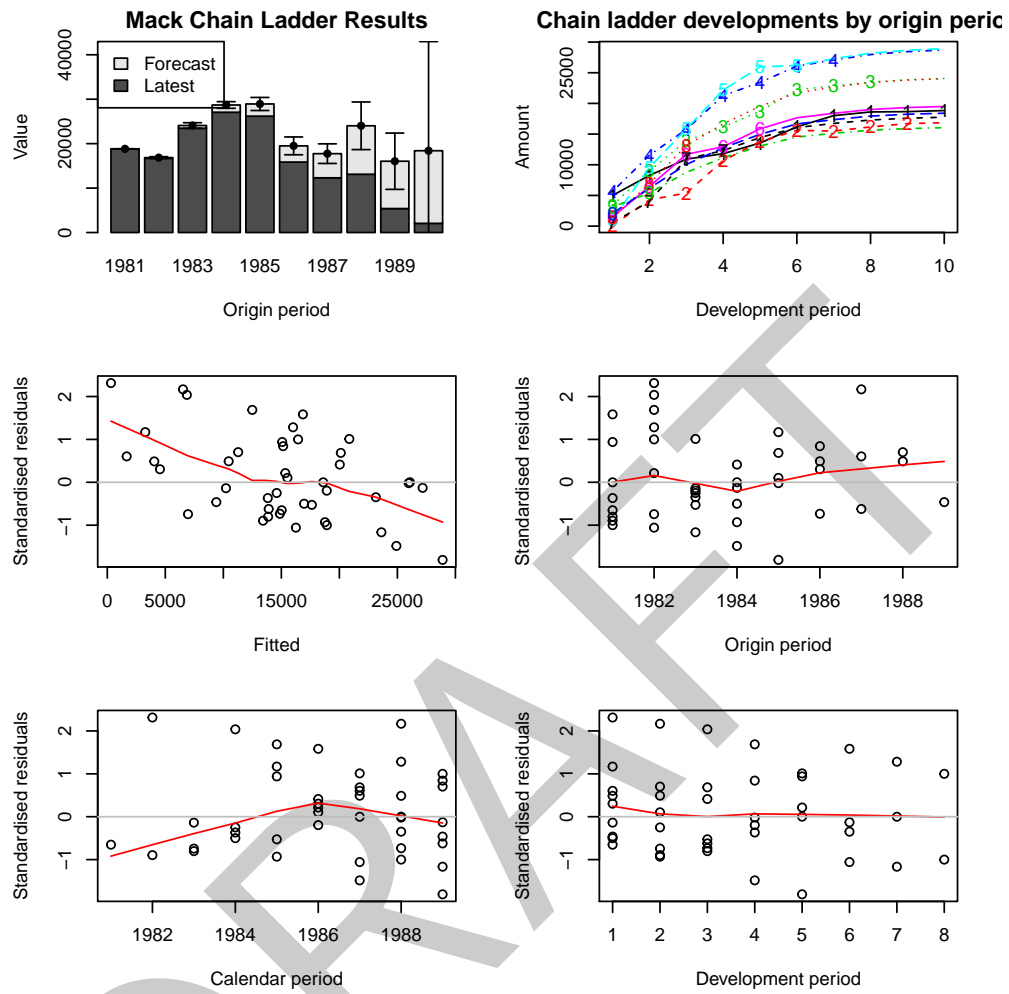
```
R> mack$f
```

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

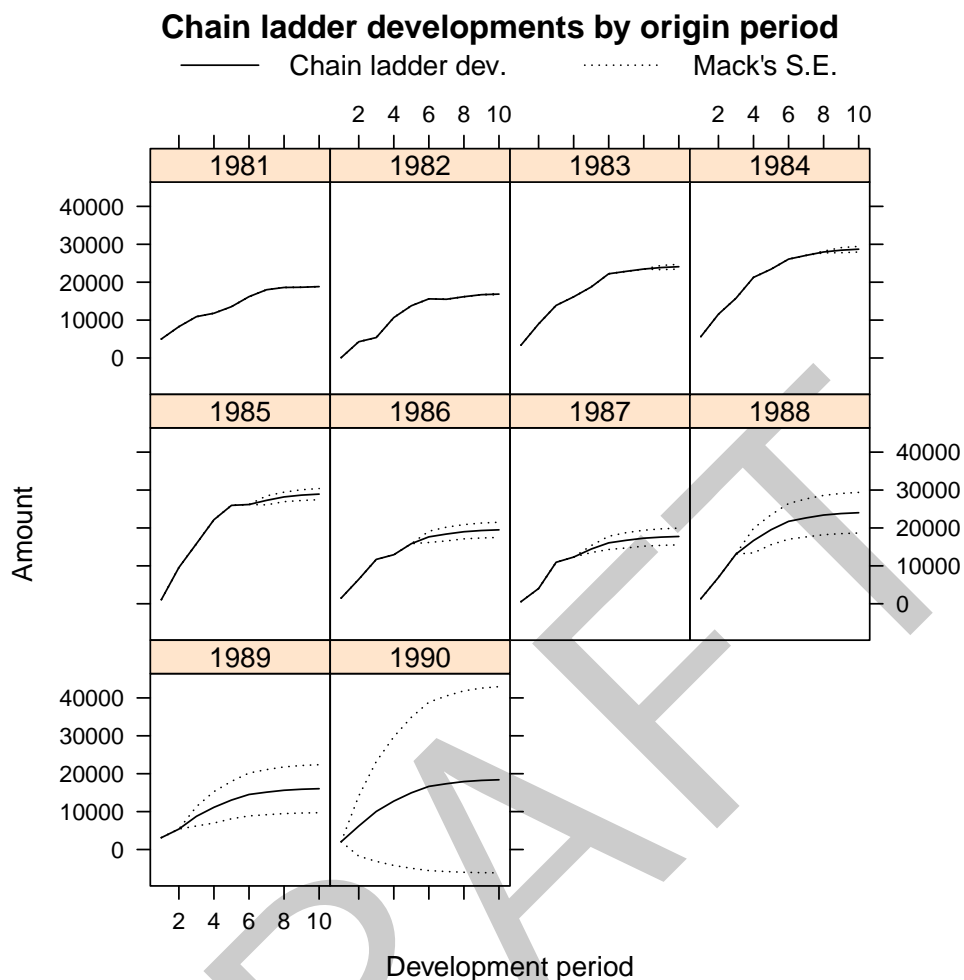
```
R> mack$FullTriangle
```

	dev	1	2	3	4	5	6	7	8	9	10
origin											
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		16,856		152		652		188		1,214
1983	23,466		24,185		719		1,334		1,229		3,284

1984	27,067	28,737	1,670	1,784	2,633	4,931
1985	26,180	29,098	2,918	2,377	4,263	7,496
1986	15,852	19,566	3,714	2,543	5,066	8,736
1987	12,314	18,094	5,780	3,190	7,695	11,726
1988	13,112	24,170	11,058	5,071	13,741	20,874
1989	5,395	16,284	10,889	5,850	14,662	21,808
1990	2,063	19,249	17,186	13,753	24,853	42,590

```
Totals
Latest:      160,987
Mean Ultimate: 215,073
Mean IBNR:    54,086
SD IBNR:      18,926
Total IBNR 75%: 65,824
Total IBNR 95%: 86,377
```

```
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	188.4	1214	2859	3291
1983	1229.4	3284	5573	6301
1984	2632.8	4931	7004	7781
1985	4262.7	7496	10010	10559
1986	5065.7	8736	11240	12049
1987	7695.5	11726	16056	17124
1988	13741.0	20874	26721	28222
1989	14661.5	21808	27900	29201
1990	24852.8	42590	60794	63953

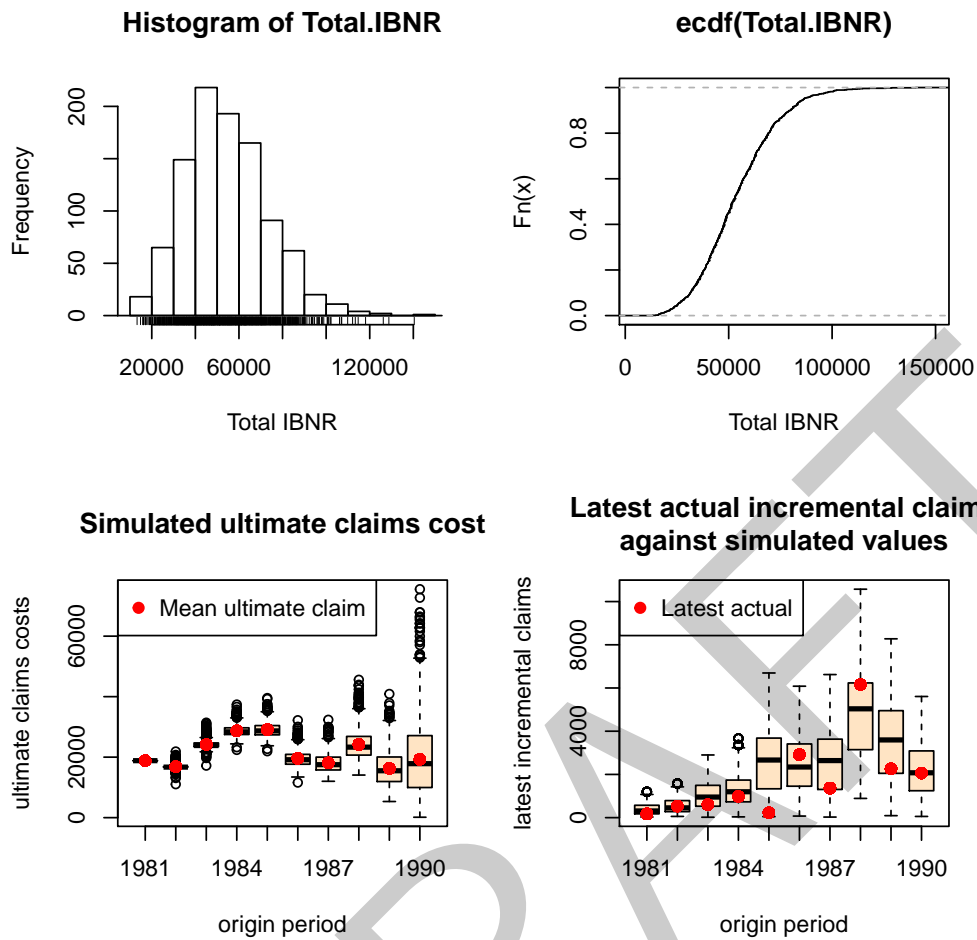
\$Totals

	Totals
IBNR 75%:	65824
IBNR 95%:	86377
IBNR 99%:	106152
IBNR 99.5%:	113380

```
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
10.83376	0.37057
(0.01172)	(0.00829)

```
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)

```

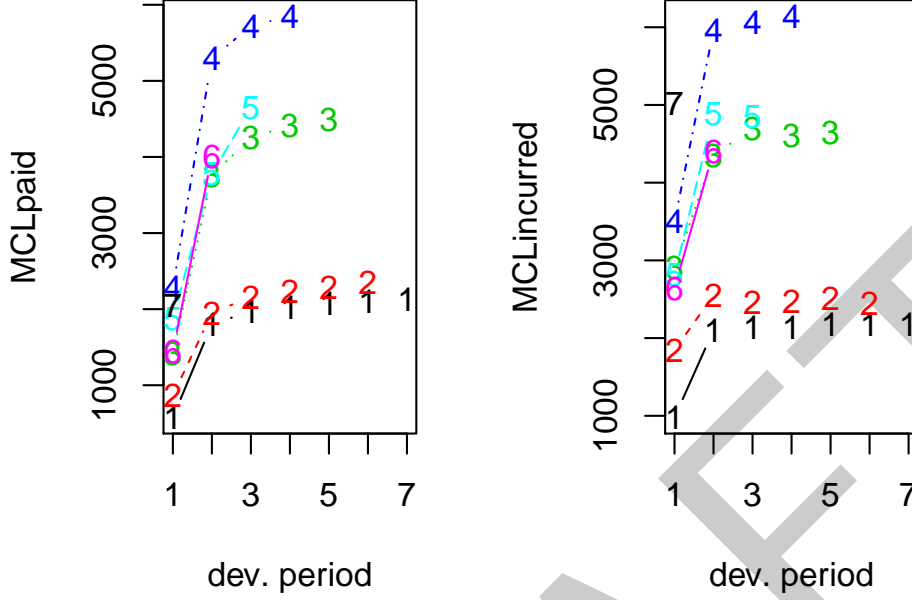
	Latest Paid	Latest Incurred	Latest P/I Ratio	Ult. Paid	Ult. Incurred
1	2,131	2,174	0.980	2,131	2,174
2	2,348	2,454	0.957	2,383	2,444
3	4,494	4,644	0.968	4,597	4,629
4	5,850	6,142	0.952	6,119	6,176
5	4,648	4,852	0.958	4,937	4,950
6	4,010	4,406	0.910	4,656	4,665
7	2,044	5,022	0.407	7,549	7,650

	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 setting where multiple reserving triangles are modeled and developed simultaneously. The advantage of the multivariate modeling is that correlations among different triangles can be modeled, which will lead to more accurate uncertainty assessment. Reserving methods that explicitly model the between-triangle contemporaneous correlations can be found in [PS05, MW08]. 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 summarized as sequential seemingly unrelated regressions [Zha10]. We note another strand of multivariate loss reserving builds 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 $Y_{i,k} = (Y_{i,k}^{(1)}, \dots, Y_{i,k}^{(N)})$ 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. [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:

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

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

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

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

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). This model is referred to as the general multivariate chain ladder [GMCL] in [Zha10]. A 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.

3.3.1 The "triangles" class

Consider the two liability loss triangles from [MW08]. It comes as a list of two matrices :

```
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> liab2 <- as(liab, "triangles")
R> class(liab2)
```

```
[1] "triangles"
attr("package")
[1] "ChainLadder"
```

We can find out what methods are available for this class:

```
R> showMethods(classes = "triangles")
```

For example, if we want to extract the last three columns of each triangle, we can use the "[" operator as follows:

```
R> # use drop = TRUE to remove rows that are all NA's
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> cbind2(liab2[1:3, 12])
```

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

3.3.2 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 Σ_k is diagonal. As a result, the multivariate model is equivalent to running multiple Mack chain ladders separately.

```
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 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. 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`

```
Latest: Dev: Ultimate: IBNR: Mack S.E.: CV(IBNR):
Totals 11343397 0.6482 17498658 6155261 427289 0.06942
```

`$AutoLiab`

```
Latest: Dev: Ultimate: IBNR: Mack S.E.: CV(IBNR):
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 Buchwalder et al. (2006), 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	3,901,463	1.0000	3,901,463	0	0	0.000
2	5,339,085	0.9826	5,433,719	94,634	75,535	0.798
3	4,909,315	0.9127	5,378,826	469,511	121,700	0.259
4	4,588,268	0.8661	5,297,906	709,638	133,551	0.188
5	3,873,311	0.7973	4,858,200	984,889	261,412	0.265
6	3,691,712	0.7223	5,111,171	1,419,459	411,028	0.290
7	3,483,130	0.6153	5,660,771	2,177,641	558,356	0.256
8	2,864,498	0.4222	6,784,799	3,920,301	875,430	0.223
9	1,363,294	0.2416	5,642,266	4,278,972	971,385	0.227
10	344,014	0.0692	4,969,825	4,625,811	1,363,385	0.295
Total	34,358,090	0.6478	53,038,946	18,680,856	2,447,618	0.131

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

```
$`Summary Statistics for Triangle 1`
      Latest Dev.To.Date Ultimate   IBNR   S.E   CV
Total 11343397      0.6484 17494907 6151510 419293 0.0682
```

```
$`Summary Statistics for Triangle 2`
      Latest Dev.To.Date Ultimate   IBNR   S.E   CV
Total 8759806      0.8095 10821341 2061535 162464 0.0788
```

```
$`Summary Statistics for Triangle 1+2`
      Latest Dev.To.Date Ultimate   IBNR   S.E   CV
Total 20103203      0.71 28316248 8213045 500607 0.061
```

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> 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> do.call("rbind", coef(fit2))
```

```
      eq1_x[[1]] 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	1.0039
[12,]	1.005	0.9989
[13,]	1.003	0.9997

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 `plot` method that produces various summary and diagnostic figures:

```
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 = 1:4)
R> par(parold)
```

The resulting plots are shown in Figure 4. We use `which.triangle` to suppress the plot for the portfolio, and use `which.plot` to select the desired types of plots. See the documentation for possible values of these two arguments.

3.3.4 Other residual covariance estimation methods

Internally, the `MultiChainLadder` calls the `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 Σ_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 [MW08]. However, this method does not work out of the box for the `liab` data, and is perhaps one of the reasons [MW08] 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 [MW08], and separate chain ladders for the rest:

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

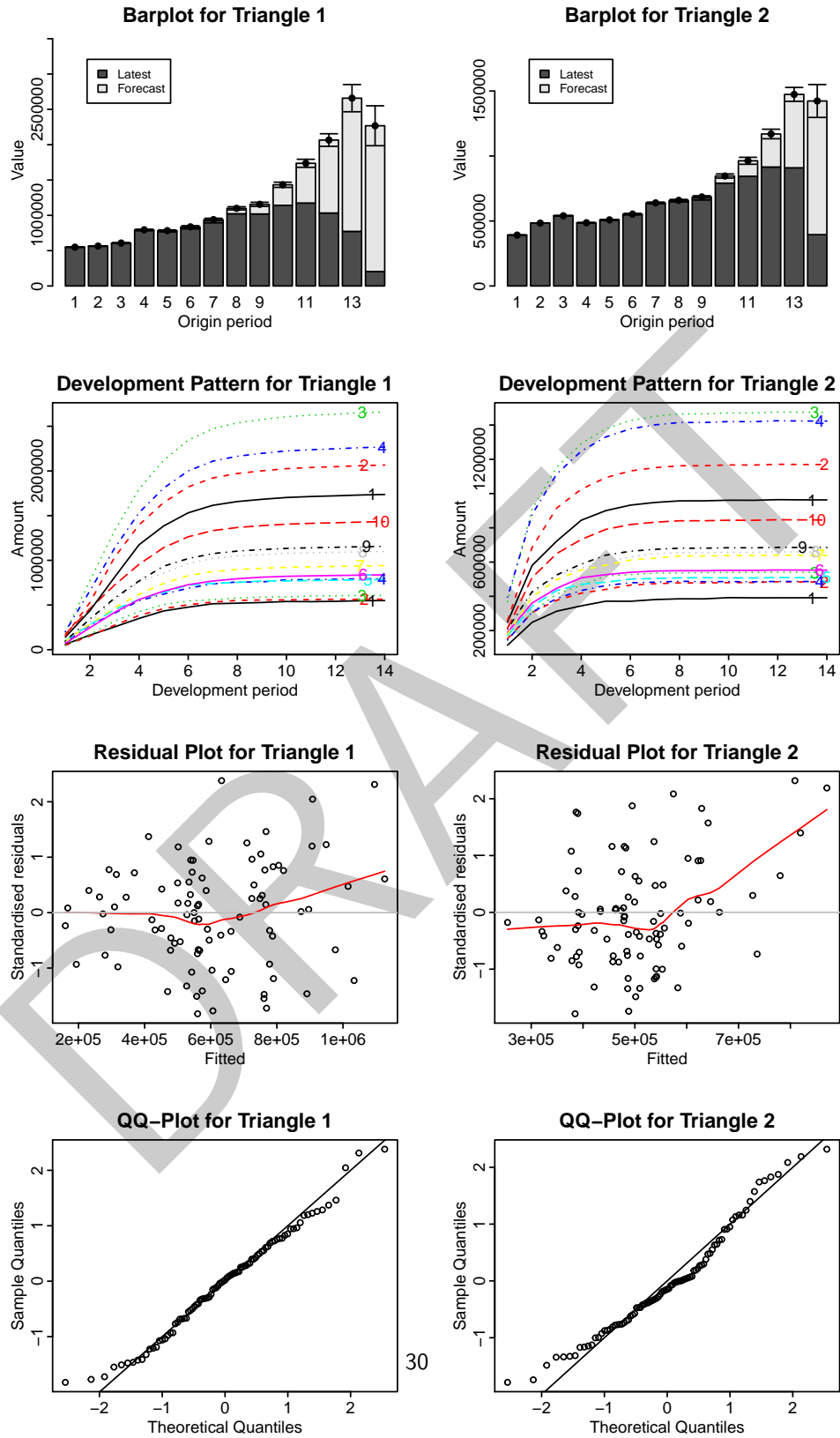


Figure 4: Summary and diagnostic plots from a MultiChainLadder object.

```
$`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 [MW08] can also be used, in which we use the control parameter `maxiter` to determine the number of iterations:

```
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 [MW08], 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. The users are encouraged to explore how other options available in `systemfit` can be applied.

3.3.5 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 a MCL model on them:

```
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)
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]
(1,2)	0.327	-0.0101	0.598	0.711	0.8565	0.928	0	0	0
(1,3)	0.870	0.9064	0.939	0.261	-0.0607	0.911	0	0	0
(2,3)	0.198	-0.3217	0.558	0.380	0.3586	0.931	0	0	0

However, from the residual plot, the first row in Figure 5, 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:

```
R> f1 <- MultiChainLadder2(auto, type = "MCL+int")
```

The corresponding residual plot is shown in the second row in Figure 5. 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:

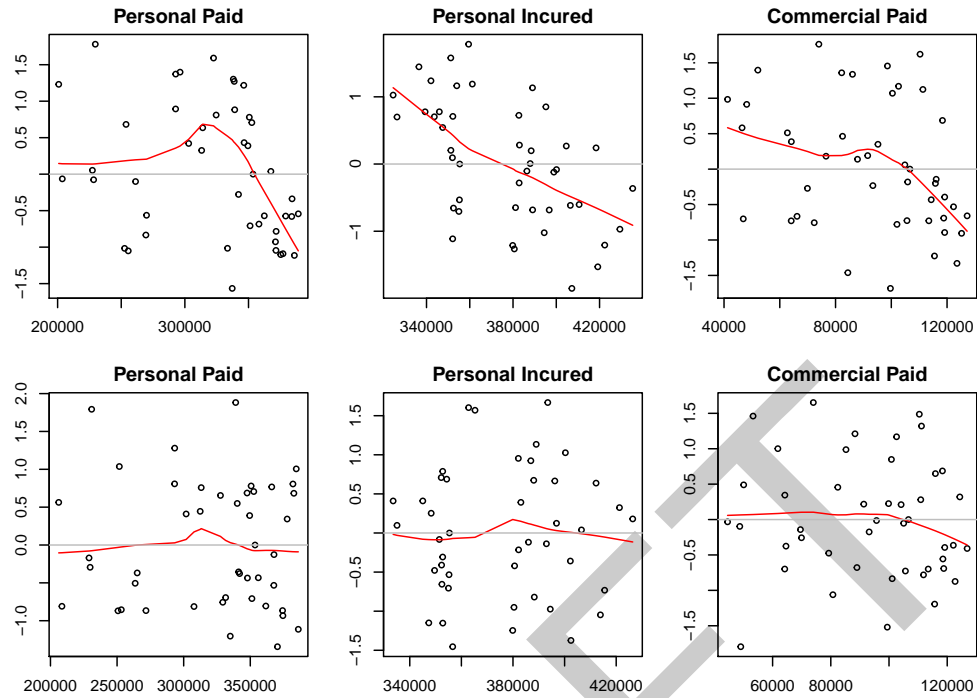


Figure 5: Residual plots for the MCL model (first row) and the GMCL (MCL+int) model (second row) for the auto data.

```
R> lapply(summary(f1, portfolio = "1+3")@report.summary, "[", 11, )
```

```
$`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
```

```
$`Summary Statistics for Triangle 3`
      Latest Dev.To.Date Ultimate IBNR  S.E  CV
Total 1043851      0.7504  1391064 347213 27716 0.0798
```

```
$`Summary Statistics for Triangle 1+3`
      Latest Dev.To.Date Ultimate IBNR  S.E  CV
Total 4334390      0.8263  5245636 911246 38753 0.0425
```

3.3.6 Joint modeling 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> 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	103.53	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

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 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 `cp1m` 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))
```

	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)
```

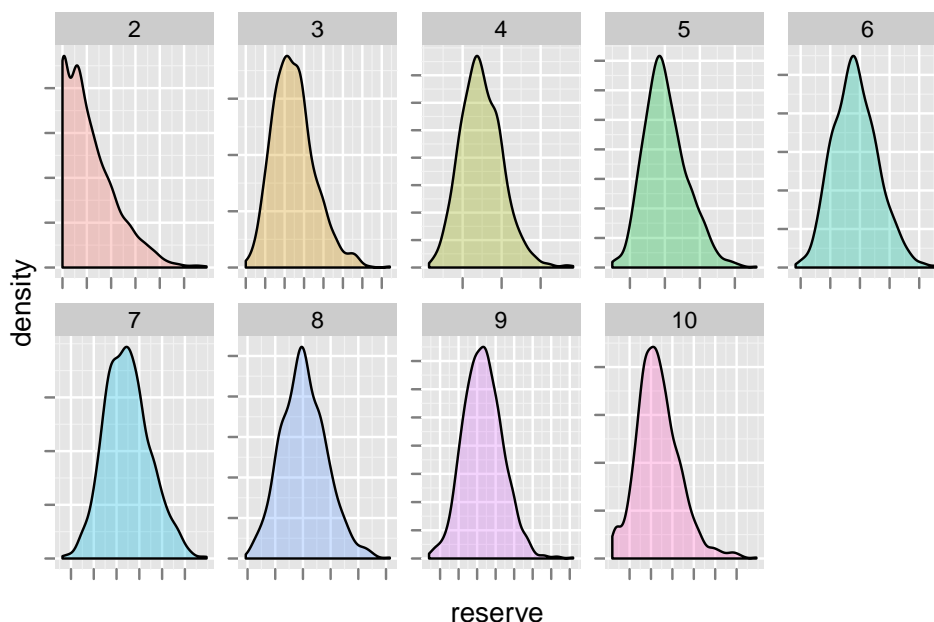


Figure 6: The predictive distribution of loss reserves for each year based on bootstrapping.

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 [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].
- `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].

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

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 presentations 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

- [BBMW06] M. Buchwalder, H. Bühlmann, M. Merz, and M.V Wüthrich. The mean square error of prediction in the chain ladder reserving method (mack and murphy revisited). *North American Actuarial Journal*, 36:521–542, 2006.
- [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.
- [MW08] Michael Merz and Mario V. Wüthrich. Prediction error of the multivariate chain ladder reserving method. *North American Actuarial Journal*, 12:175–197, 2008.
- [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.
- [PS05] Carsten Pröhl and Klaus D. Schmidt. Multivariate chain-ladder. *Dresdner Schriften zur Versicherungsmathematik*, 2005.
- [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.
- [ZDG12] Yanwei Zhang, Vanja Dukic, and James Guszczka. A bayesian nonlinear model for forecasting insurance loss payments. *Journal of the Royal Statistical Society, Series A*, 175:637–656, 2012.
- [Zha10] Yanwei Zhang. A general multivariate chain ladder model. *Insurance: Mathematics and Economics*, 46:588 – 599, 2010.
- [Zha12] Yanwei Zhang. Likelihood-based and bayesian methods for tweedie compound poisson linear mixed models. *Statistics and Computing*, 2012. forthcoming.
- [Zx00] Ben Zehnwrith and Glenn xBarnett. Best estimates for reserves. *Proceedings of the CAS*, LXXXVII(167), November 2000.