London R-user group meeting

Claims reserving in R
The *ChainLadder* package

Markus Gesmann

31 March 2009
Claims reserving in insurance

- Insurers sell the promise to pay for future claims occurring over an agreed period for an upfront received premium.

- Unlike other industries insurers don’t know the production cost of their product.

- The estimated future claims have to be held in the reserves, one of the biggest liability items on an insurer’s balance sheet.
Reserving in insurance

- Reserves cover IBNR (Incurred But Not Reported) claims
- Reserves are usually estimated based on historical claims payment/reporting patterns
- The most popular method is called “chain ladder”
- In the past a point estimator for the reserves was sufficient
- New regulatory requirements (→ Solvency II) foster stochastic methods
Example

- Usually an insurance portfolio is split into 'homogeneous" classes of business, e.g. motor, marine, property, etc.
- Policies are aggregated by class and looked at in a triangle view of reported claims to forecast future claims developments.
Stochastic reserving

- Over recent years stochastic methods have been developed and published, but have been rarely used in practice
- Excel is the standard tool in the industry, but is not an ideal environment for implementing those stochastic methods
- Idea: Use R to implement stochastic reserving methods, and CRAN to share them
- Use the RExcel Add-in as a front end for Excel to use R functions
The *ChainLadder* package for R

- Started out of presentations given at the Institute of Actuaries on stochastic reserving
- Mack-, Munich- and Bootstrap-chain-ladder implemented
- Example spreadsheet shows how to use the functions with Excel using the RExcel Add-in
- Available from CRAN - sources and binaries
- Contribution most welcome!
The ChainLadder package

Agenda:

- Getting started
- ChainLadder package philosophy
- Examples for
  - MackChainLadder
  - MunichChainLadder
  - BootChainLadder

Current version: 0.1.2-11
Getting started

- Start R and type for
  - Installation:
    `install.packages("ChainLadder")`
  - Loading the package:
    `library(ChainLadder)`
  - Help:
    `?ChainLadder`
  - Examples:
    `example(ChainLadder)`
### Example data sets

- The *ChainLadder* package comes with some example data sets, e.g.

```r
> library(ChainLadder)
> RAA
```

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

Incurred claims development by origin year

> matplot(t(RAA), t="b")
Working with triangles

- Transform from cumulative to incremental
  
  ```
  incRAA <- cbind(RAA[,1], t(apply(RAA,1,diff)))
  ```

- Transform from incremental to cumulative
  
  ```
  cumRAA <- t(apply(incRAA,1, cumsum))
  ```

- Triangles to long format
  
  ```
  lRAA <- expand.grid(origin=as.numeric(dimnames(RAA)$origin), dev=as.numeric(dimnames(RAA)$dev))
  lRAA$value <- as.vector(RAA)
  ```

- Long format to triangle
  
  ```
  reshape(x, timevar="dev", idvar="origin", v.names="value", direction="wide")
  ```
ChainLadder package philosophy

- Use the linear regression function "lm" as much as possible and utilise its output
- The chain-ladder model for volume weighted average link ratios is expressed as a formula:
  \[ y \sim x + 0, \text{weights}=1/x \]
  and can easily be changed
- Provide tests for the model assumptions
Chain-ladder as linear regression

Chain-ladder can be regarded as weighted linear regression through the origin:

\[ x \leftarrow \text{RAA[,1]} \quad \# \text{dev. period 1} \]
\[ y \leftarrow \text{RAA[,2]} \quad \# \text{dev. period 2} \]

\[ \text{model} \leftarrow \text{lm}(y \sim x + 0, \text{weights}=1/x) \]

Call:
\[ \text{lm(formula} = y \sim x + 0, \text{weights} = 1/x) \]

Coefficients:
\[ x \]
\[ 2.999 \quad \text{chain-ladder link-ratio} \]
Full regression output

> summary(model)

Call:
  lm(formula = y ~ x + 0, weights = 1/x)

Residuals:
     Min      1Q  Median      3Q     Max
-95.54   -71.50   49.03    99.55  385.32

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
x             2.999     1.130    2.654   0.0291 *

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

Residual standard error: 167 on 8 degrees of freedom
Multiple R-squared:  0.4682,  Adjusted R-squared:  0.4017
F-statistic: 7.043 on 1 and 8 DF,  p-value: 0.02908
Chain-ladder using the \textit{“lm”} function

Idea: Create linear model for each development period

\begin{verbatim}
ChainLadder <- function(tri, weights=1/tri){
  n <- ncol(tri)
  myModel <- vector("list", (n-1))
  for(i in c(1:(n-1))){
    myModel[[i]] <- lm(y~x+0,
                       data.frame(x=tri[,i], y=tri[,i+1]),
                       weights=weights[,i])
  }
  return(myModel)
}
\end{verbatim}
Accessing regression statistics

CL <- ChainLadder(RAA)

# Get chain-ladder link-ratios
sapply(CL, coef)
# 2.999359 1.623523 1.270888 1.171675 1.113385
# 1.041935 1.033264 1.016936 1.009217

# Get residual standard errors
sapply(lapply(CL, summary), "[[", "sigma")
# 166.983470  33.294538  26.295300   7.824960  10.928818
#   6.389042   1.159062   2.807704        NaN

# Get R squared values
sapply(lapply(ChainLadder(RAA), summary), "[[", "r.squared")
# 0.4681832  0.9532872  0.9704743  0.9976576  0.9959779
# 0.9985933  0.9999554  0.9997809  1.0000000
Mack-chain-ladder

Mack’s chain-ladder method calculates the standard error for the reserves estimates.

The method works for a cumulative triangle $C_{ik}$ if the following assumptions are hold:

1. All accident years are independent

\[
E \left[ \frac{C_{i,k+1}}{C_{ik}} \mid C_{i1}, C_{i2}, \ldots, C_{ik} \right] = f_k
\]

\[
\text{Var} \left( \frac{C_{i,k+1}}{C_{ik}} \mid C_{i1}, C_{i2}, \ldots, C_{ik} \right) = \frac{\sigma_k^2}{C_{ik}}
\]

All accident years are independent
MackChainLadder

Usage:
MackChainLadder(Triangle,
weights = 1/Triangle,
tail=FALSE,
est.sigma="log-linear")

- Triangle: cumulative claims triangle
- weights: default (1/Triangle) volume weighted CL
- tail: estimator for the tail
- est.sigma: Estimator for sigma_{n-1}
**MackChainLadder example**

```r
library(ChainLadder)
M <- MackChainLadder(Triangle = RAA, est.sigma = "Mack")
M
```

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

**Totals**

- **Latest:** 160,987.00
- **Ultimate:** 213,122.23
- **IBNR:** 52,135.23
- **Mack S.E.:** 26,909.01
- **CV(IBNR):** 0.52
The residual plots show the standardised residuals against fitted values, origin period, calendar period and development period.

All residual plot should show no pattern or direction for Mack's method to be applicable.

Pattern in any direction can be the result of trends and require further investigations.

plot(M)
Munich-chain-ladder

- Munich-chain-ladder (MCL) is an extension of Mack’s method that reduces the gap between IBNR projections based on paid (P) and incurred (I) losses
  - Mack has to be applicable to both triangles
- MCL adjusts the chain-ladder link-ratios depending if the momentary (P/I) ratio is above or below average
- MCL uses the correlation of residuals between P vs. (I/P) and I vs. (P/I) chain-ladder link-ratio to estimate the correction factor
Munich-chain-ladder example

P/I triangle

Full P/I triangle using chain ladder

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

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
**MunichChainLadder**

**Usage:**

MunichChainLadder(Paid, Incurred,

   
est.sigmaP = "log-linear",
est.sigmaI = "log-linear",
tailP=FALSE, tailI=FALSE)

- Paid: cumulative paid claims triangle
- Incurred: cumulative incurred claims triangle
- est.sigmaP, est.sigmaI: Estimator for sigma_{n-1}
- tailP, tailI: estimator for the tail
MunichChainLadder example

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

<table>
<thead>
<tr>
<th>Latest</th>
<th>Latest Paid</th>
<th>Latest Incurred</th>
<th>Latest P/I Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2,131</td>
<td>2,174</td>
<td>0.980</td>
</tr>
<tr>
<td>2</td>
<td>2,348</td>
<td>2,454</td>
<td>0.957</td>
</tr>
<tr>
<td>3</td>
<td>4,494</td>
<td>4,644</td>
<td>0.968</td>
</tr>
<tr>
<td>4</td>
<td>5,850</td>
<td>6,142</td>
<td>0.952</td>
</tr>
<tr>
<td>5</td>
<td>4,648</td>
<td>4,852</td>
<td>0.958</td>
</tr>
<tr>
<td>6</td>
<td>4,010</td>
<td>4,406</td>
<td>0.910</td>
</tr>
<tr>
<td>7</td>
<td>2,044</td>
<td>5,022</td>
<td>0.407</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ult. Paid</th>
<th>Ult. Incurred</th>
<th>Ult. P/I Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,131</td>
<td>2,174</td>
<td>0.980</td>
</tr>
<tr>
<td>2,383</td>
<td>2,444</td>
<td>0.975</td>
</tr>
<tr>
<td>4,597</td>
<td>4,629</td>
<td>0.993</td>
</tr>
<tr>
<td>6,119</td>
<td>6,176</td>
<td>0.991</td>
</tr>
<tr>
<td>4,937</td>
<td>4,950</td>
<td>0.997</td>
</tr>
<tr>
<td>4,656</td>
<td>4,665</td>
<td>0.998</td>
</tr>
<tr>
<td>7,549</td>
<td>7,650</td>
<td>0.987</td>
</tr>
</tbody>
</table>

Totals

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

Munich-chain-ladder forecasts based on paid and incurred losses
1. Munich Chain Ladder Results

2. Munich Chain Ladder vs. Standard Chain Ladder

3. Paid residual plot

4. Incurred residual plot

1. MCL forecasts on P and I
2. Comparison of Ultimate P/I ratios of MCL and Mack
3. I/P link-ratio residuals against P link-ratio residuals
4. P/I link-ratio residuals against I link-ratio residuals

plot(MCL)
Bootstrap-chain-ladder

- **BootChainLadder** uses a two-stage approach.
  1. Calculate the scaled Pearson residuals and bootstrap R times to forecast future incremental claims payments via the standard chain-ladder method.
  2. Simulate the process error with the bootstrap value as the mean and using an assumed process distribution.

- The set of reserves obtained in this way forms the predictive distribution, from which summary statistics such as mean, prediction error or quantiles can be derived.
BootChainLadder

Usage:
BootChainLadder(Triangle, R = 999,
    process.distr=c("gamma",
    "od.pois"))

- Triangle: cumulative claims triangle
- R: Number of resampled bootstraps
- process.distr: Assumed process distribution
BootChainLadder example

```r
set.seed(1)
BootChainLadder(Triangle = RAA, R = 999, process.distr = "od.pois")
```

<table>
<thead>
<tr>
<th>Year</th>
<th>Latest</th>
<th>Mean</th>
<th>Ultimate</th>
<th>Mean</th>
<th>IBNR</th>
<th>SD</th>
<th>IBNR 75%</th>
<th>IBNR 95%</th>
</tr>
</thead>
<tbody>
<tr>
<td>1981</td>
<td>18,834</td>
<td></td>
<td>18,834</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1982</td>
<td>16,704</td>
<td>16,921</td>
<td>217</td>
<td>710</td>
<td>253</td>
<td>1,597</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1983</td>
<td>23,466</td>
<td>24,108</td>
<td>642</td>
<td>1,340</td>
<td>1,074</td>
<td>3,205</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1984</td>
<td>27,067</td>
<td>28,739</td>
<td>1,672</td>
<td>1,949</td>
<td>2,679</td>
<td>4,980</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1985</td>
<td>26,180</td>
<td>29,077</td>
<td>2,897</td>
<td>2,467</td>
<td>4,149</td>
<td>7,298</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1986</td>
<td>15,852</td>
<td>19,611</td>
<td>3,759</td>
<td>2,447</td>
<td>4,976</td>
<td>8,645</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1987</td>
<td>12,314</td>
<td>17,724</td>
<td>5,410</td>
<td>3,157</td>
<td>7,214</td>
<td>11,232</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1988</td>
<td>13,112</td>
<td>24,219</td>
<td>11,107</td>
<td>5,072</td>
<td>14,140</td>
<td>20,651</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1989</td>
<td>5,395</td>
<td>16,119</td>
<td>10,724</td>
<td>6,052</td>
<td>14,094</td>
<td>21,817</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1990</td>
<td>2,063</td>
<td>18,714</td>
<td>16,651</td>
<td>13,426</td>
<td>24,459</td>
<td>42,339</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Totals

- **Latest**: 160,987
- **Mean Ultimate**: 214,066
- **Mean IBNR**: 53,079
- **SD IBNR**: 18,884
- **Total IBNR 75%**: 64,788
- **Total IBNR 95%**: 88,037
1. Histogram of simulated total IBNR

2. Empirical distribution of total IBNR

3. Box-whisker plot of simulated ultimate claims cost by origin period

4. Test if latest actual incremental loss could come from simulated distribution of claims cost
Generic Methods

- **Mack-, Munich-, BootChainLadder**
  - names: gives the individual elements back
  - summary: summary by origin and totals
  - print: nice formatted output
  - plot: plot overview of the results

- **MackChainLadder**
  - residuals: chain-ladder residuals

- **BootChainLadder**
  - mean: mean IBNR by origin and totals
  - quantile: gives quantiles of the simulation back
  - residuals: chain-ladder residuals
More help

- See example on project web page
- Read documentation on CRAN: http://cran.r-project.org/web/packages/ChainLadder/ChainLadder.pdf
- Read help pages in R:
  - ?MackChainLadder
  - ?MunichChainLadder
  - ?BootChainLadder
- Follow examples in R:
  - example(MackChainLadder)
  - example(MunichChainLadder)
  - example(BootChainLadder)
Conclusions

- R is ideal for reserving
  - Built-in functions for statistical modelling
  - Powerful language for data manipulations
  - Fantastic graphical capabilities for analysis and presentation
- RExcel add-in allows to share R functions with colleagues without R knowledge
- rcom allows to control of MS Office from R
- Easy to set-up connections to databases (ODBC)
- Effective knowledge transfer - plain text files
Example *rcom*: Control MS Office from R

- Using the *rcom* R-package you can write output from R into MS Office application
  - Example: Run *MackChainLadder* and put the plot into a new PowerPoint slide

```r
library(ChainLadder)
R <- MackChainLadder(RAA)
myfile=tempfile()
win.metafile(file=myfile)
plot(R)
dev.off()

# library(rcom)
ppt<-comCreateObject("Powerpoint.Application")
comSetProperty(ppt,"Visible",TRUE)
myPresColl<-comGetProperty(ppt,"Presentations")
myPres<-comInvoke(myPresColl,"Add")
mySlides<-comGetProperty(myPres,"Slides")
mySlide<-comInvoke(mySlides,"Add",1,12)
myShapes<-comGetProperty(mySlide,"Shapes")
myPicture<-comInvoke(myShapes,"AddPicture",myfile,0,1,100,10)
```
References

library(ChainLadder)
## Example data
RAA
## Triangle development plot
matplot(t(RAA), t="b")
## Chain ladder ratio
x <- RAA[,1]
y <- RAA[,2]
model <- lm(y~x+0, weights=1/x)
## Full model output
summary(model)
## Simple chain ladder function
ChainLadder <- function(Triangle, weights=1/Triangle){
  n <- ncol(Triangle)
  myModel <- vector("list", (n-1))
  for(i in c(1:(n-1))){
    dev.data <- data.frame(x=Triangle[,i],
                           y=Triangle[,i+1])
    myModel[[i]] <- lm(y~x+0, weights=weights[,i],
                       data=dev.data)
  }
  return(myModel)
}
CL <- ChainLadder(RAA)
# Get chain-ladder link-ratios
sapply(CL, coef)
# 2.999359 1.623523 1.270888 1.171675 1.113385
# 1.041935 1.033264 1.016936 1.009217
## Get residual standard errors
sapply(lapply(CL, summary), "sigma")
# 166.983470 33.294538 26.299300 7.824960 10.928818
# 6.389042 1.159062 2.807704 NaN
## Get R squared values
sapply(lapply(ChainLadder(RAA), summary), "r.squared")
# 0.9998132 0.9532872 0.9704743 0.9976576 0.9959779
# 0.9985933 0.9999554 0.997809 1.000000

## MunichChainLadder
MCL <- MunichChainLadder(Paid = MCLpaid, Incurred = MCLincurred, est.sigmaP = 0.1, est.sigmaI = 0.1)
MCL
plot(MCL)
## BootChainLadder
set.seed(1)
B <- BootChainLadder(Triangle = RAA, R = 999, process.distr = "od.pois")
plot(B) quantile(B, probs=(0.75, 0.995))
## IBNR distribution fit
fit <- fitdist(B$IBNR.Totals[B$IBNR.Totals>0], "lognormal")
fit
## Fancy 3 plot, requires package "rgl"
library(rgl)
MCL=MackChainLadder(GenIns)
FT <- MCL$FullTriangle
FTpSE <- FT+MCL$Mack.S.E
FTmSE <- FT-MCL$Mack.S.E
mSE <- data.frame(as.table(FTmSE))
points3d(xyz.coords(x=as.numeric(as.character(mSE$origin)), y=as.numeric(as.character(mSE$dev)), z=mSE$Freq), size=2)
pSE <- data.frame(as.table(FTpSE))
points3d(xyz.coords(x=as.numeric(as.character(pSE$origin)), y=as.numeric(as.character(pSE$dev)), z=pSE$Freq), size=2)
# IBNR distribution fit
fit <- fitdist(B$IBNR.Totals[B$IBNR.Totals>0], "lognormal")
fit
curve(plnorm(x,fit$estimate["meanlog"], fit$estimate["sdlog"]), col="red", add=TRUE)
## MunichChainLadder
MCL <- MunichChainLadder(Paid = MCLpaid, Incurred = MCLincurred, est.sigmaP = 0.1, est.sigmaI = 0.1)
MCL
plot(MCL)