The Actuarial Profession
making financial sense of the future

R you ready?
One-Day Workshop
24 July, Staple Inn, London

Claims reserving in R
The *ChainLadder* package

Markus Gesmann

Contact: markus.gesmann@gmail.com

Content

- Introduction
- *ChainLadder* package
- R and databases
- R and MS Office interfaces

Double click the paperclip symbol in the bottom of this slide to access all the R code of this presentation.

Claims reserving in insurance

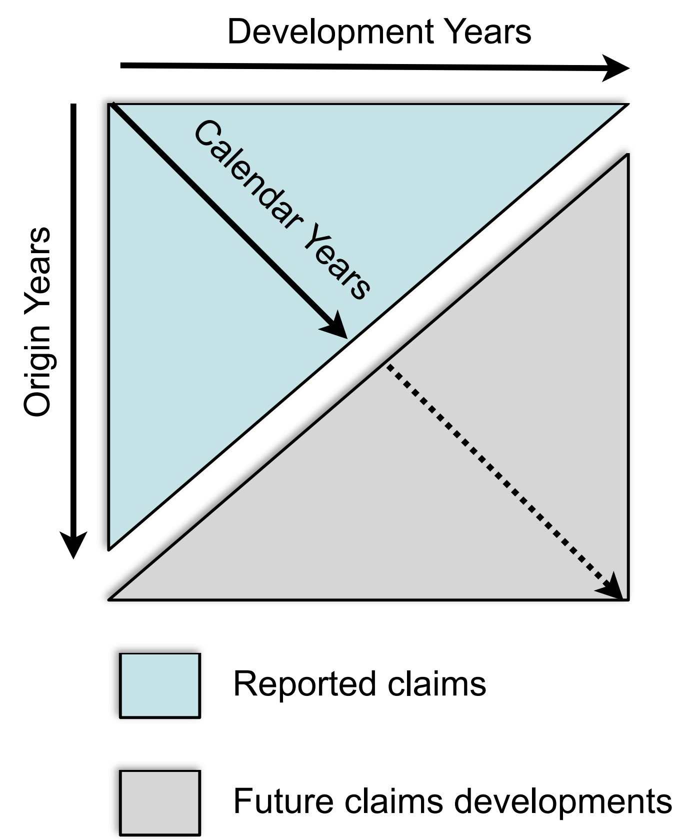
- **Insurers** sell the **promise** to pay for future claims occurring over an agreed period for an upfront received premium
- Unlike other industries insurers **don’t know** the production **cost** of their product
- The **estimated future claims** have to be held in the **reserves**, one of the biggest 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
- In the past a point estimator for the reserves was sufficient
- New regulatory requirements ($\rightarrow$ Solvency II) foster stochastic methods

Typical scenario

- 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
- Excel is often the standard tool, 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

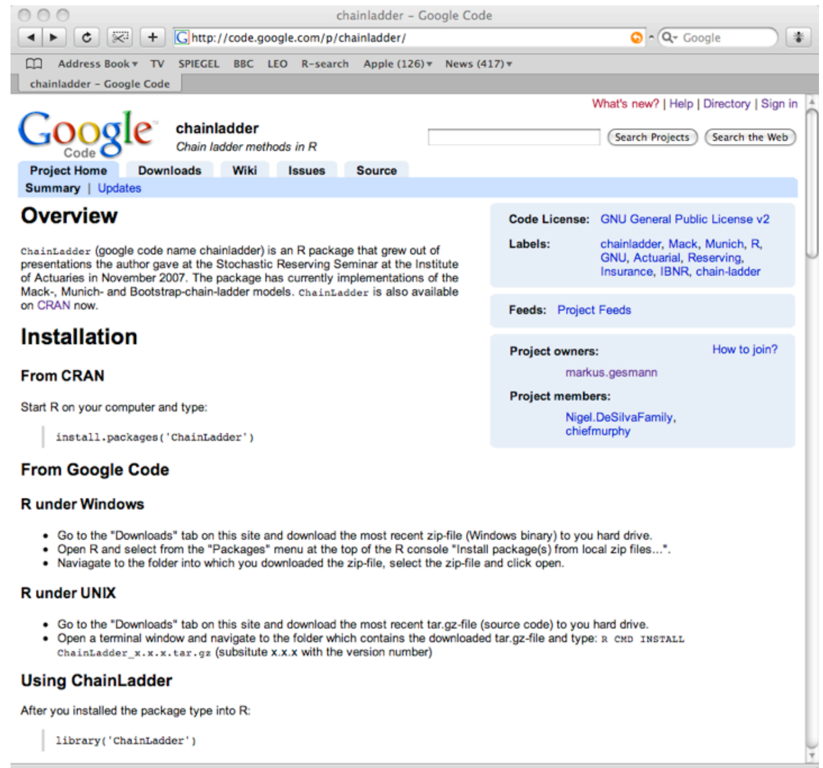
The *ChainLadder* package

- 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*



Project web page: http://code.google.com/p/chainladder/
Current version: 0.1.2-11

Getting started

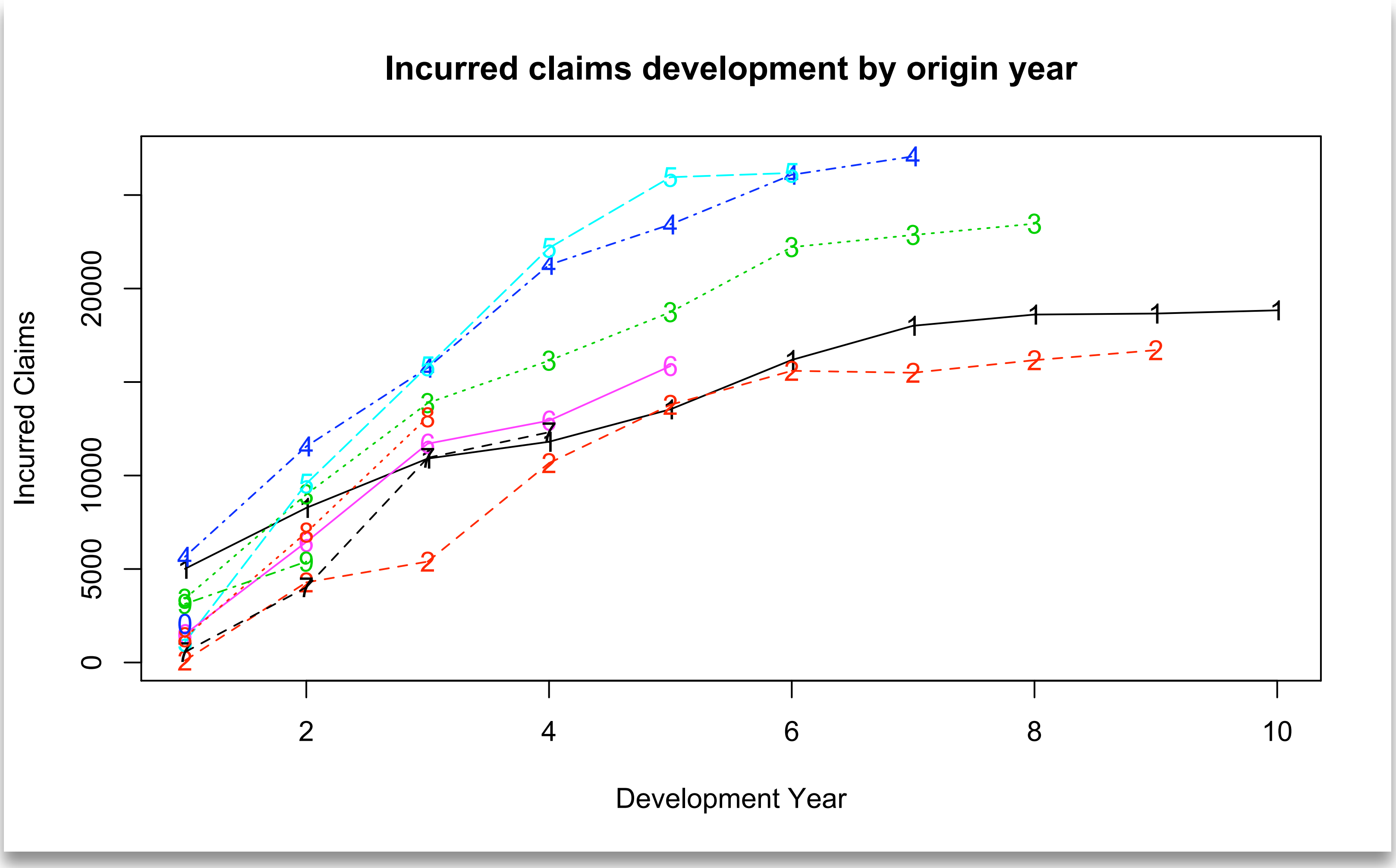
- Start R and type for
  - Installation:
    `install.packages("ChainLadder")`
  - Loading the package:
    `library(ChainLadder)`
  - Help:
    `?ChainLadder`
  - Examples:
    `example(ChainLadder)`

Example data sets

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

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

Triangle plot




```
> 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 (see later for *as.ArrayTriangle* function, works much better with *ChainLadder*)

```
reshape(lRAA, 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 ~ x + 0, 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 <- RAA[,1] # dev. period 1
y <- RAA[,2] # dev. period 2
model <- lm(y ~ x + 0, weights=1/x)
Call:
lm(formula = y ~ x + 0, weights = 1/x)
Coefficients:
    x
2.999
```

Force it through the origin

Volume weighted

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 ' '
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
```

The output shows:

- model formula
- chain-ladder link ratio
- std. error of the link ratio
- P-value
- Residual std. error

Chain-ladder using the "lm" function

Idea: Create linear model for each development period

```
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)
}
```

Accessing regression statistics

```
CL <- ChainLadder(RAA)
```

Get chain-ladder link-ratios

```
sapply(CL, coef)
# 2.999 1.62 1.27 1.17 1.11 1.04 1.03 1.016 1.0
```

Get residual standard errors

```
sapply(lapply(CL, summary), "[[", "sigma")
# 166.98 33.29 26.295 7.82 10.9 6.389 1.159 2.8 NaN
```

Get R squared values

```
sapply(lapply(ChainLadder(RAA), summary), "[[",
"r.squared")
# 0.468 0.95 0.97 0.997 0.995 0.998 0.999 0.999 1.00
```

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:

- $$E\left[\frac{C_{i,k+1}}{C_{ik}}|C_{i1}, C_{i2}, \ldots, C_{ik}\right] = f_k$$
- $$\mathrm{Var}\left(\frac{C_{i,k+1}}{C_{ik}}|C_{i1}, C_{i2}, \ldots, C_{ik}\right) = \frac{\sigma_k^2}{C_{ik}}$$
- All accident years are independent

MackChainLadder

Usage:

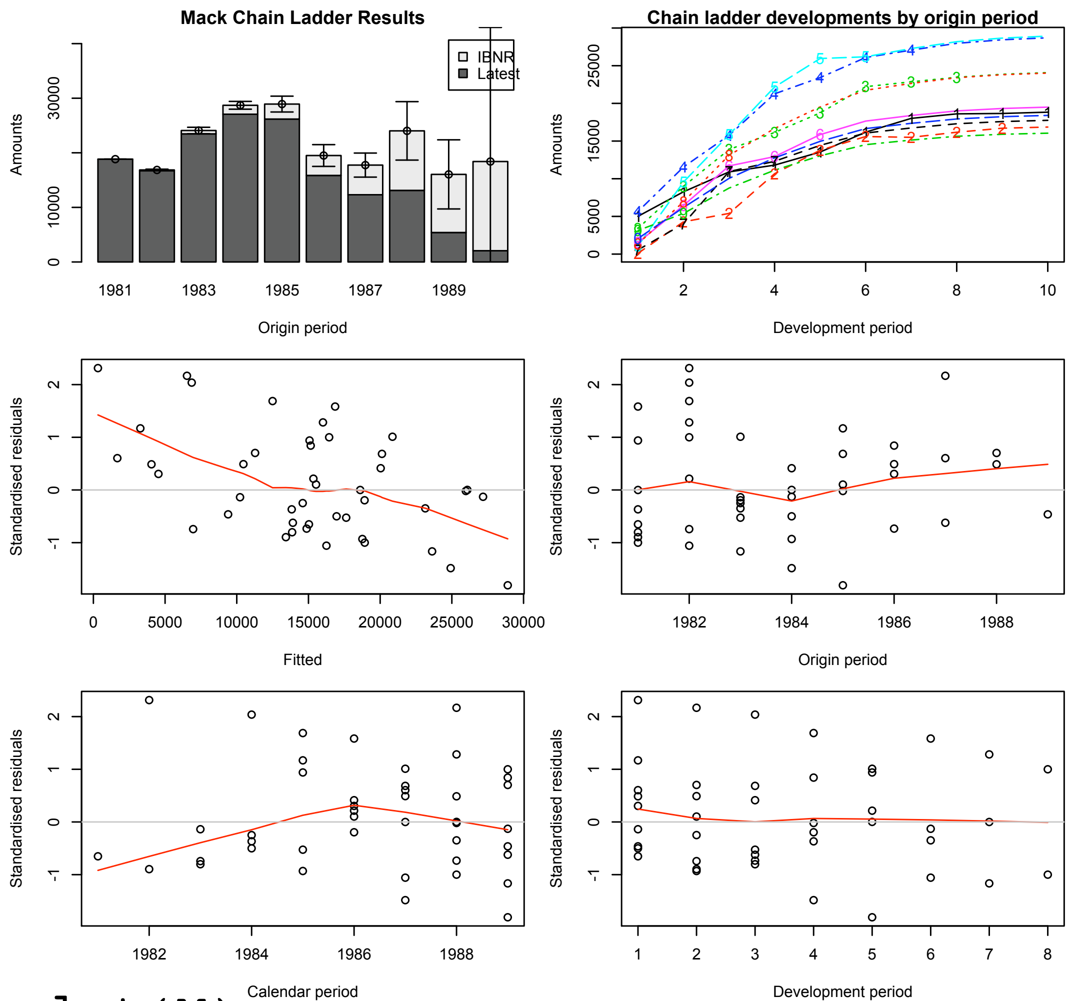
```
MackChainLadder(Triangle,
                weights = 1/Triangle,
                est.sigma="log-linear",
                tail=FALSE, tail.se=NULL,
                tail.sigma=NULL)
```

- Triangle: cumulative claims triangle
- weights: default (1/Triangle) volume weighted CL
- est.sigma: Estimator for sigma_{n-1}
- tail, tail.se, tail.sigma: estimators for the tail

MackChainLadder example

```
library(ChainLadder)
M <- MackChainLadder(Triangle = RAA, est.sigma = "Mack")
M
      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
Ultimate:   213,122.23
IBNR:        52,135.23
Mack S.E.:   26,909.01
CV(IBNR):         0.52
```

plot.MackChainLadder




```
plot(M)
```

The residual plots show the standardised residuals against fitted values, origin period, calendar period and development period.

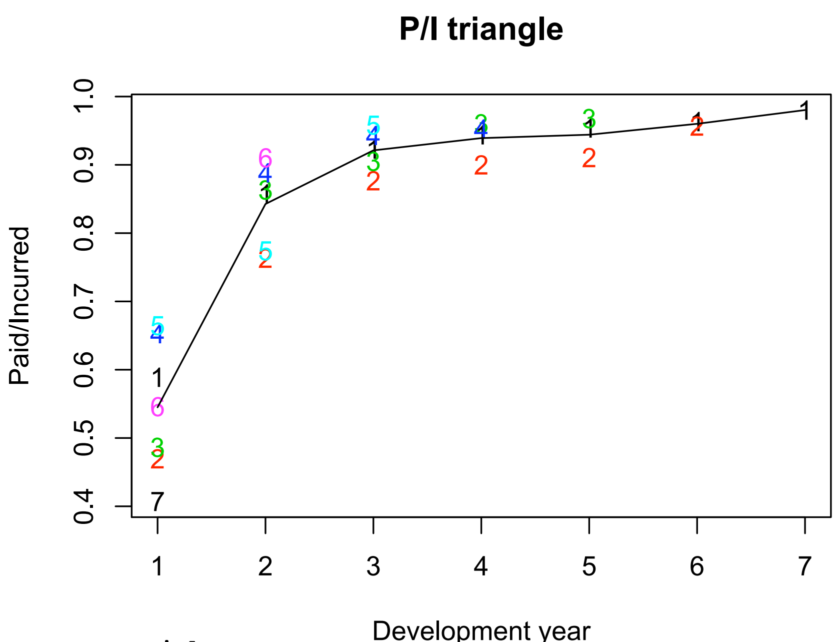
All residual plots should show no pattern or direction for Mack's method to be applicable.

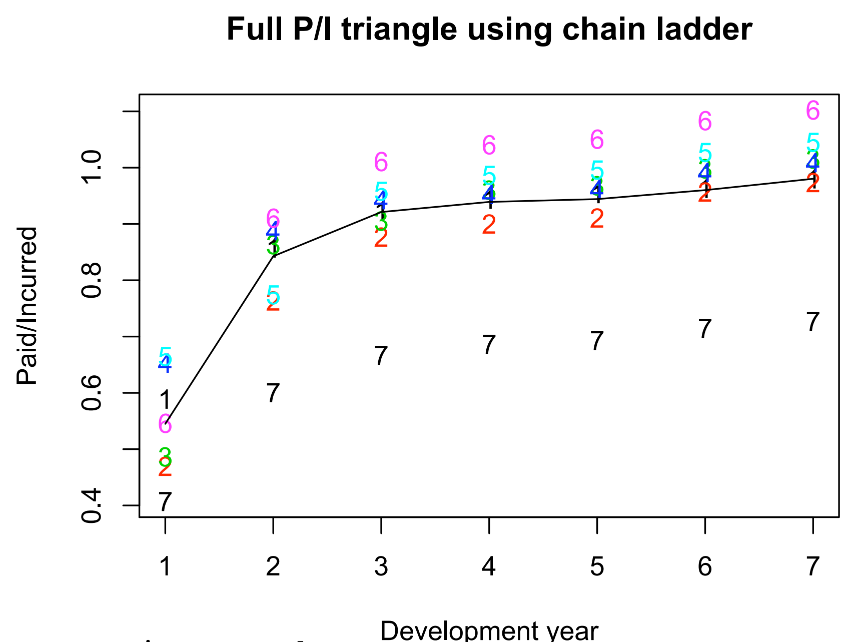
Pattern in any direction can be the result of trends and require further investigations.

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






```
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   N
```

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

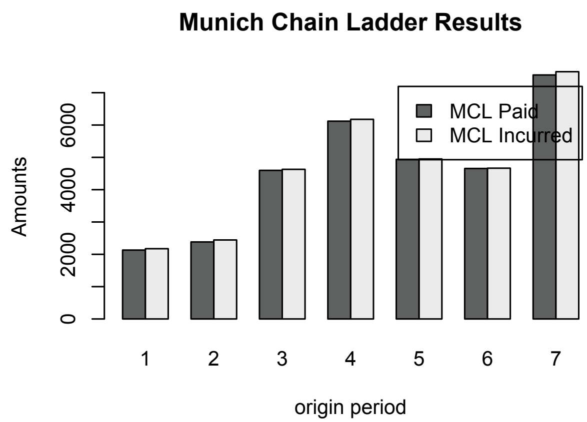
MunichChainLadder example

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

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

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

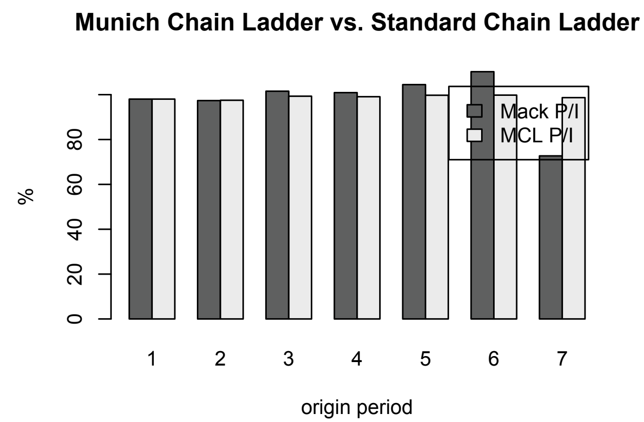
Munich-chain-ladder forecasts based on paid and incurred losses

plot.MunichChainLadder

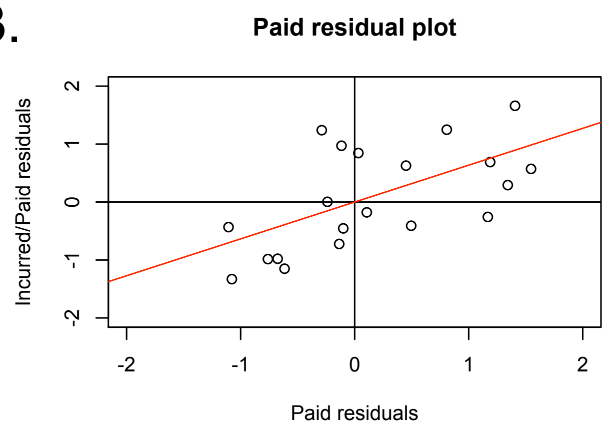
1.



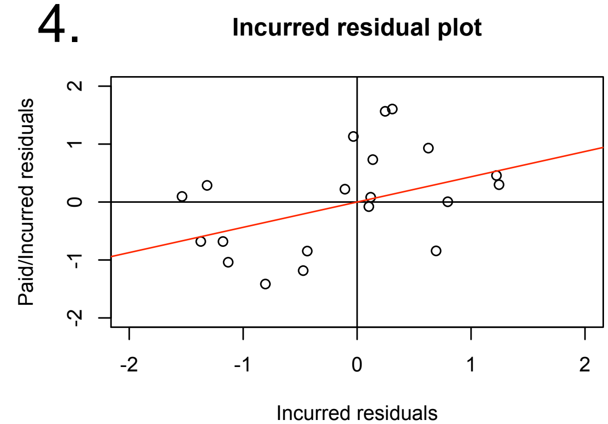
2.



3.



4.




```
plot(MCL)
```

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-ratios residuals

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"))
```

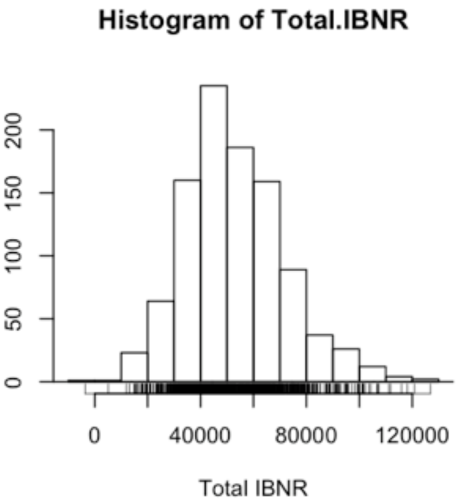
- Triangle: cumulative claims triangle
- R: Number of resampled bootstraps
- process.distr: Assumed process distribution

BootChainLadder example

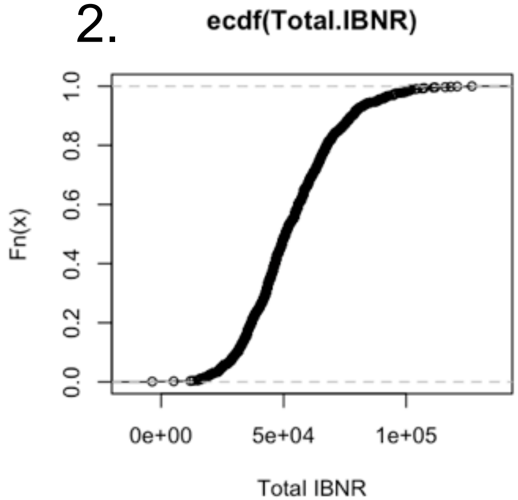
```
set.seed(1)
BootChainLadder(Triangle = RAA, R = 999, process.distr = "od.pois")

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

                    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
```

plot.BootChainLadder

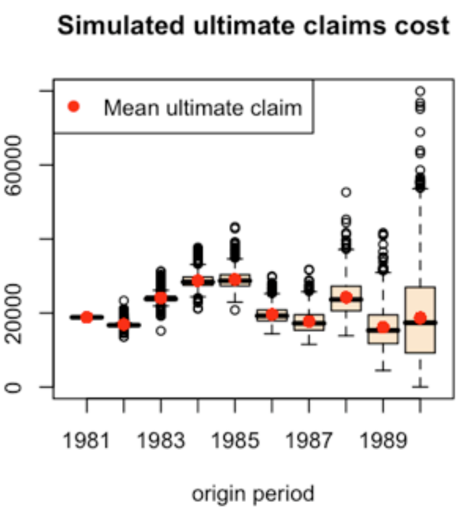
1.



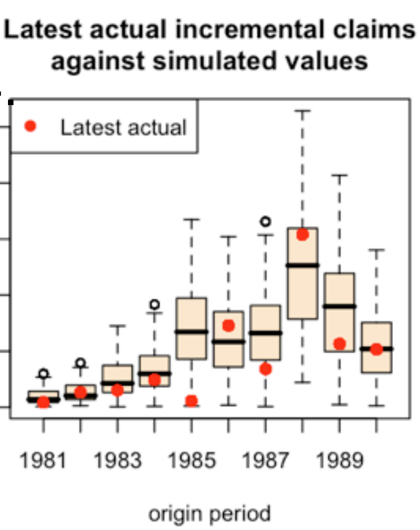
2.



3.



4.



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

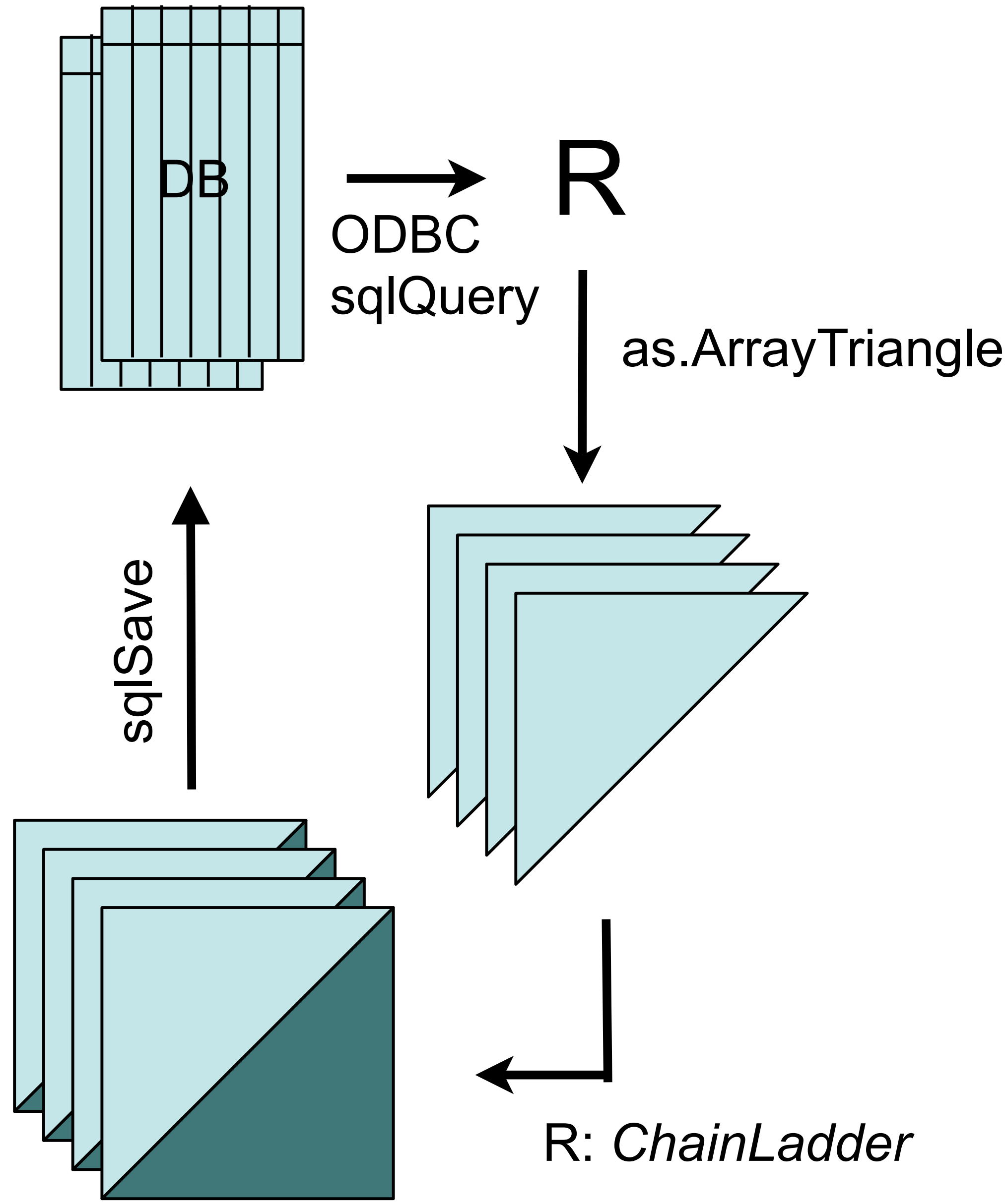
Generic Methods

- *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

R and databases

Agenda:

- Create test data
- ODBC - SQL
- Query database
- Tables to triangles
- Apply functions
- Write to database



Working with databases

- Triangles are usually stored in databases
  - Triangles are stored in long tables
- Use ODBC to connect to databases
- Use SQL to interact with databases
- Use R to transform tables into triangles
- Apply *ChainLadder* function across many triangles in **one statement**
- Write results back into database

Create sample data in a table format

Use example data sets to create a sample data table

```
tri=list(RAA=RAA, Mortgage=Mortgage, GenIns=GenIns, ABC=ABC)
# create function to transform triangle into long format
longTriangle <- function(triangle){
    long <- expand.grid(origin=as.numeric(dimnames(triangle)$origin),
                        dev=as.numeric(dimnames(triangle)$dev))
      long$value <- as.vector(triangle)
      return(na.omit(long))
}
# apply the new function to our list
ltri <- lapply(tri, longTriangle)
# add the names of the triangles to the list
ltri <- lapply(names(ltri), function(x) data.frame(LOB=x, ltri[[x]]))
# transform list into data.frame
triangleTable <- do.call("rbind", ltri)
```

Write test data into database

Example with MS Access 2003
See also documentation for RODBC

```
library(RODBC)
# Create a test database in c:/Temp (here MS Access 2003)
channel <- odbcConnectAccess(
            "C:/Temp/ChainLadderTestData.mdb")
sqlSave(channel, triangleTable, "tblTestTriangles",
rownames=FALSE)
odbcClose(channel)
```

Read tables from database

Access data via ODBC and SQL-statements

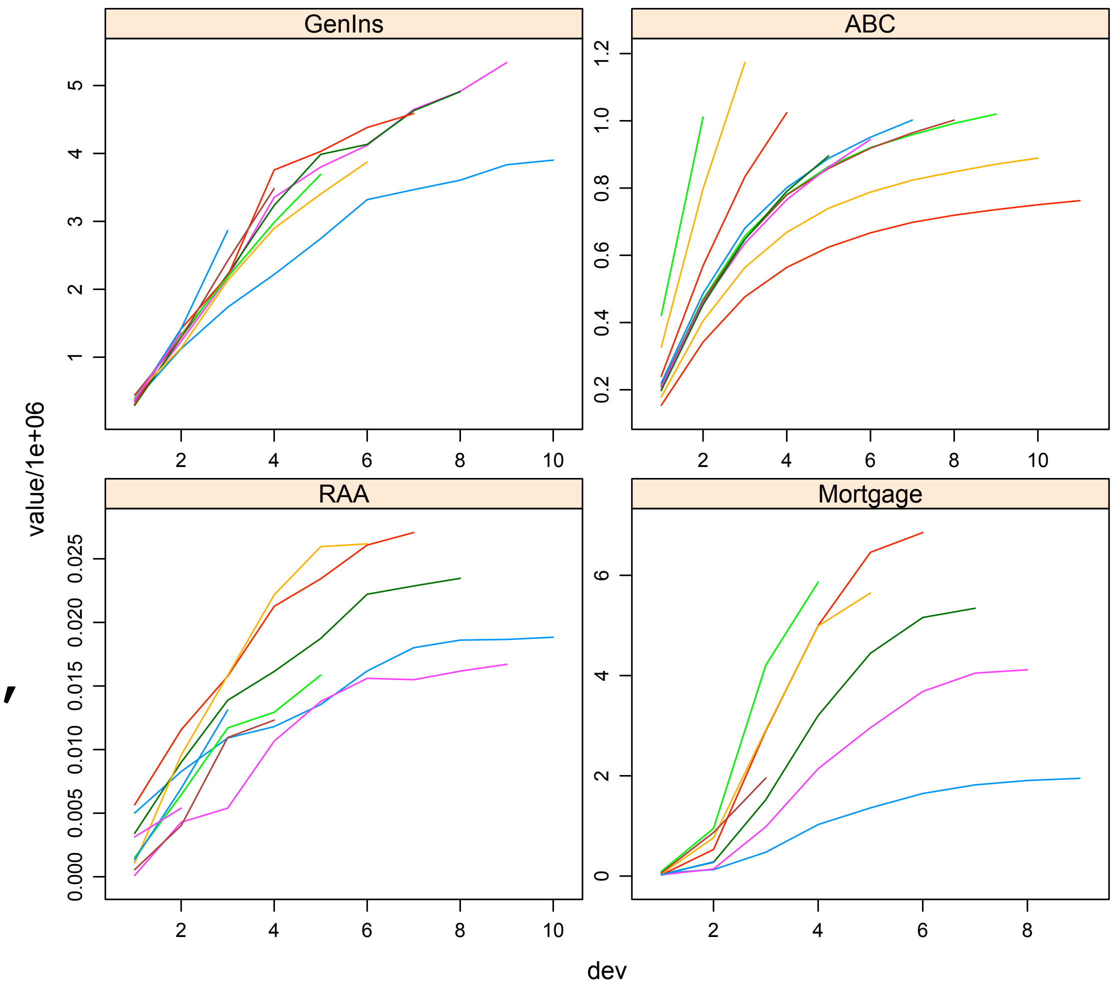
ChainLadderTestData.mdb attached at the bottom of this page (double click on paper clip) and save in C:/Temp

```
# From database
channel <- odbcConnectAccess(
                  "C:/Temp/ChainLadderTestData.mdb")
myData <- sqlQuery(channel,
                   "SELECT * FROM tblTestTriangles;")
odbcClose(channel)
```

As an aside: Plot tables with *lattice*

Triangles stored in long tables are much easier to plot than triangles in cross-tab formats

```
# Plot long triangles
library(lattice)
xyplot(
 value/1e6 ~ dev | LOB,
 groups=origin, t="l",
 data=myData,
 scales="free"
)
```



Transform tables into triangles

We use the *array* function rather than *reshape*, as its output is ready to be used by *ChainLadder*

```
as.ArrayTriangle <- function(x){
 # x has to be a data.frame with columns: origin, dev and value
 .names <- apply(x[,c("origin", "dev", "value")], 2, unique)
 .namesOD <- .names[c("origin", "dev")]
 # Expand to include entire array, in case don't have complete data
 .id <- paste(x$origin, x$dev,  sep='.')
 .grid <- expand.grid(.namesOD)
 .grid$id <- paste(.grid$origin, .grid$dev, sep='.')
 .grid$data <- x$value[match(.grid$id, .id)]
 # Create data array
 .data <- array(.grid$data, dim=unlist(lapply(.namesOD, length)),
                dimnames=.namesOD)
 return(.data)
}
```

Use *by to* apply *ChainLadder* functions

- *by* function applies functions on sub sets of data
  - convert table for each LOB into a triangle
  - apply *MackChainLadder* for each triangle
- Output is stored in a list

```
myResults <- by(myData, list(LOB=myData$LOB),
    function(x){
       triangle <- as.ArrayTriangle(x)
       M <- MackChainLadder(triangle, est.sigma="Mack")
       return(M)
 })
myResults
```

Combine results in tables

- Use *lapply* to access *MackChainLadder* output
  - Access origin year and total results separately

```
OriginResults <- lapply(lapply(myResults, summary), "[[", "ByOrigin")
# add the names of the triangles to the list
OriginResults <- lapply(names(OriginResults ),
                   function(x) data.frame(LOB=x, OriginResults[[x]]))
# transform list into data.frame
OriginResultsTable <- do.call("rbind", OriginResults)

TotalResults <- lapply(lapply(lapply(myResults, summary),
                   "[[", "Totals"),t)
# add the names of the triangles to the list
TotalResults <- lapply(names(TotalResults ),
                   function(x) data.frame(LOB=x, TotalResults[[x]]))
# transform list into data.frame
TotalResultsTable <- do.call("rbind", TotalResults)
```

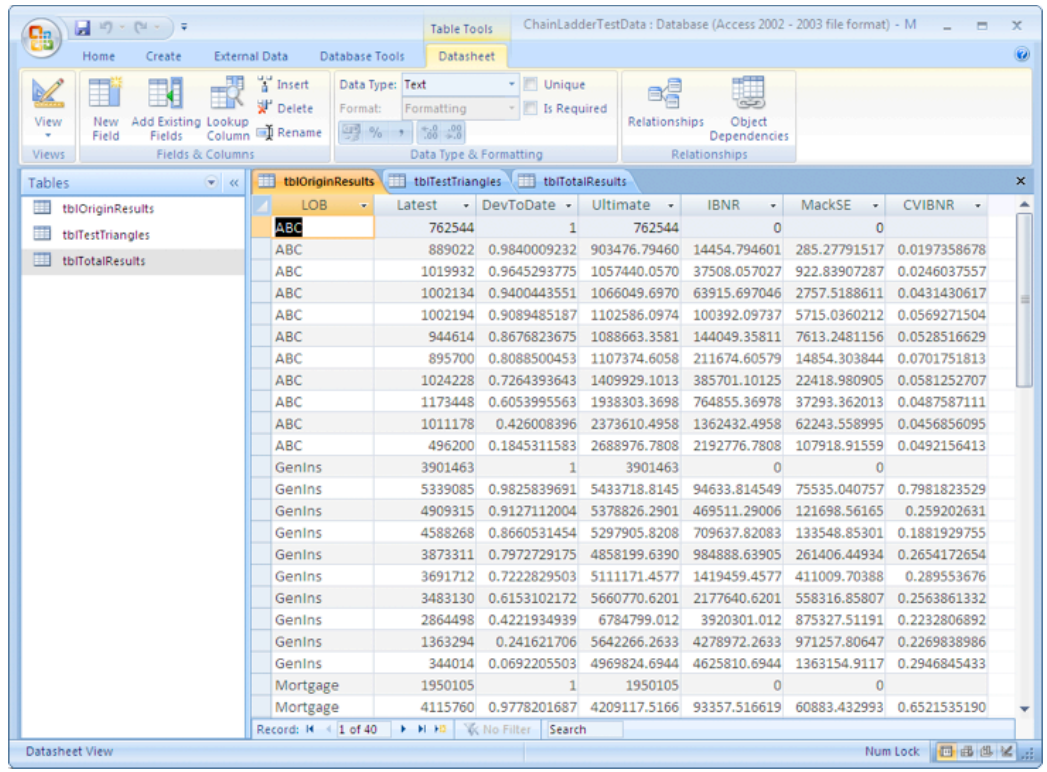
Write results into database

- Write results back into new tables of the database via QDBC and *sqlSave*

```
channel <- odbcConnectAccess("C:/Temp/
ChainLadderTestData.mdb")
sqlSave(channel, OriginResultsTable, "myOriginResults",
        rownames=FALSE)
sqlSave(channel, TotalResultsTable, "myTotalResults",
        rownames=FALSE)
odbcClose(channel)
```

Database summary

- Use R to query DB
- Transform table to triangles
- Apply *ChainLadder* function across all triangles
- Summaries results
- Save results in DB

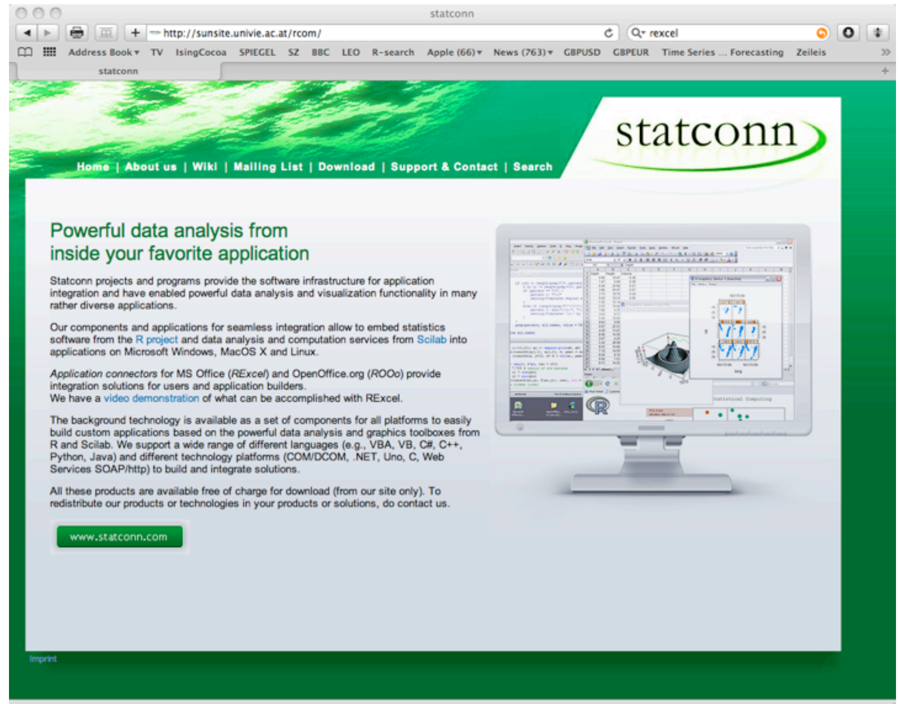


LOB	Latest	DevToDate	Ultimate	IBNR	MackSE	CVIBNR
ABC	762544	1	762544	0	0	
ABC	889022	0.9840009232	903476.79460	14454.794601	285.27791517	0.0197358678
ABC	1019932	0.9645293775	1057440.0570	37508.057027	922.83907287	0.0246037557
ABC	1002134	0.9400443551	1066049.6970	63915.697046	2757.5188611	0.0431430617
ABC	1002194	0.9089485187	1102586.0974	100392.09737	5715.0360212	0.0569271504
ABC	944614	0.8676823675	1088663.3581	144049.35811	7613.2481156	0.0528516629
ABC	895700	0.8088500453	1107374.6058	211674.60579	14854.303844	0.0701751813
ABC	1024228	0.7264393643	1409929.1013	385701.10125	22418.980905	0.0581252707
ABC	1173448	0.6053995563	1938303.3698	764855.36978	37293.362013	0.0487587111
ABC	1011178	0.426008396	2373610.4958	1362432.4958	62243.558995	0.0456856095
ABC	496200	0.1845311583	2688976.7808	2192776.7808	107918.91559	0.0492156413
GenIns	3901463	1	3901463	0	0	
GenIns	5339085	0.9825839691	5433718.8145	94633.814549	75535.040757	0.7981823529
GenIns	4909315	0.9127112004	5378826.2901	469511.29006	121698.56165	0.259202631
GenIns	4588268	0.8660531454	5297905.8208	709637.82083	133548.85301	0.1881929755
GenIns	3873311	0.7972729175	4858199.6390	984888.63905	261406.44934	0.2654172654
GenIns	3691712	0.7222829503	5111171.4577	1419459.4577	411009.70388	0.289553676
GenIns	3483130	0.6153102172	5660770.6201	2177640.6201	558316.85807	0.2563861332
GenIns	2864498	0.4221934939	6784799.012	3920301.012	875327.51191	0.2232806892
GenIns	1363294	0.241621706	5642266.2633	4278972.2633	971257.80647	0.2269838986
GenIns	344014	0.0692205503	4969824.6944	4625810.6944	1363154.9117	0.2946845433
Mortgage	1950105	1	1950105	0	0	
Mortgage	4115760	0.9778201687	4209117.5166	93357.516619	60883.432993	0.6521535190

R and MS Office interfaces

Agenda:

- win.metafile
- Clipboard
- RExcel
- COM-server
- *rcom*



Windows meta-file

- Windows meta-file (WMF, or EMF (Enhanced meta-file) is a vector graphic format
- High quality, but editable format for MS Office
- Create WMF-files in R with *win.metafile()*

```
win.metafile(file="C:/Temp/Testplot.wmf")
plot(sin(seq(0,round(2*pi,2),0.01)))
dev.off()
```

Clipboard to exchange data

Copy and paste from R to and from Excel

- R -> Excel

```
mydf=data.frame(x=1:10, y=letters[1:10])
write.table(mydf, file="clipboard",
sep="\t", row.names=FALSE)
```

- Excel -> R

```
read.table(file= "clipboard", sep="\t")
```

RExcel - Using R from within Excel

RExcel Add-in allows to use R functions from Excel, see:

http://sunsite.univie.ac.at/rcom/

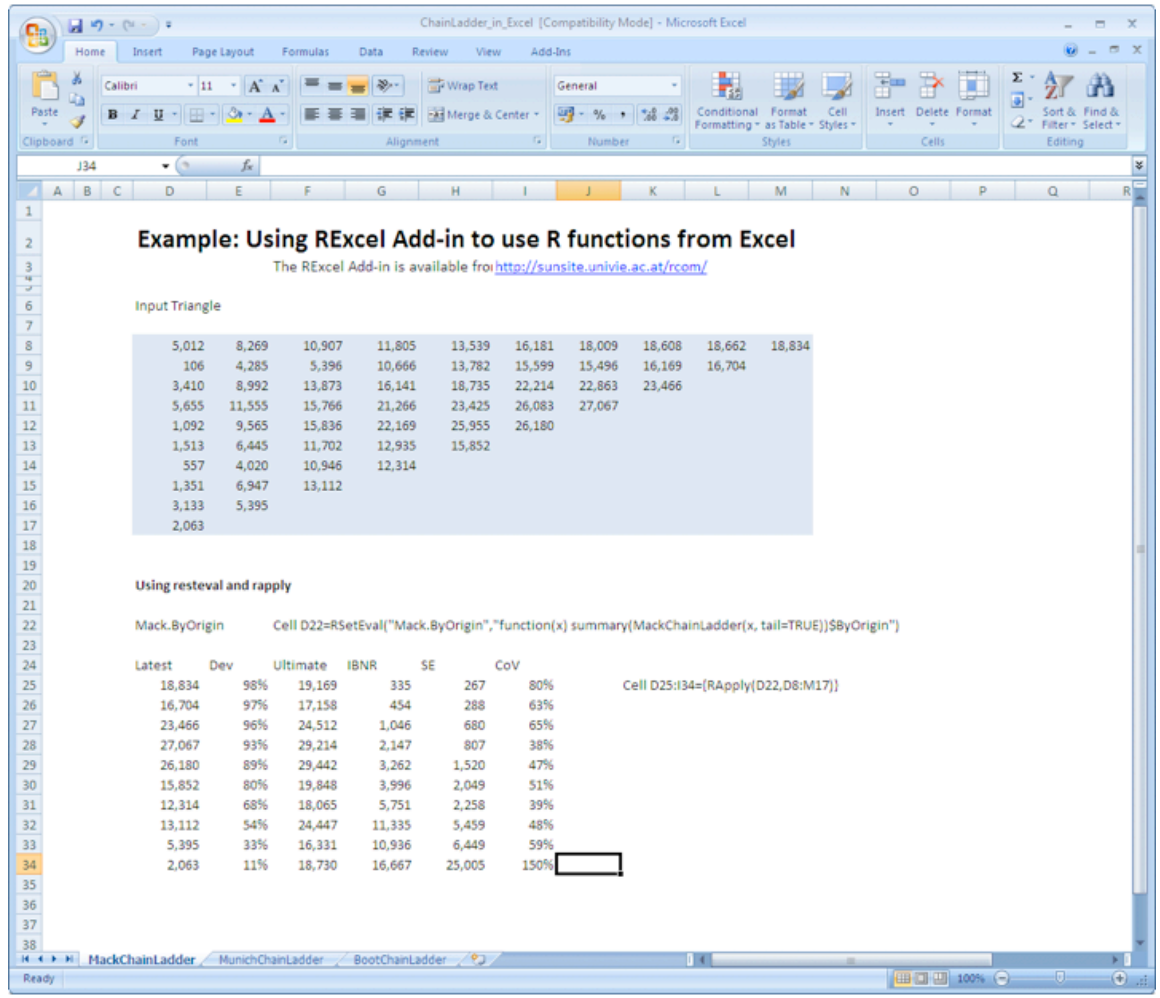
There are at least three different ways of using R from within Excel

- Scratchpad mode
  - Writing R Code directly in an Excel worksheet and transferring scalar, vector, and matrix variables between R and Excel
- Macro mode
  - Writing macros using VBA and the macros supplied by RExcel, attaching the macros to menu items or toolbar items
- Worksheet functions
  - R can be called directly in functions in worksheet cells

Source: http://sunsite.univie.ac.at/rcom/server/doc/RExcel.html

ChainLadder_in_Excel.xls

- RExcel allows to use R functions within Excel
- Package comes with example file
- R function can be embedded and are interactive
- Use R graphics



Example: Using RExcel Add-in to use R functions from Excel

The RExcel Add-in is available fro[m] http://sunsite.univie.ac.at/rcom/

Input Triangle

5,012	8,269	10,907	11,805	13,539	16,181	18,009	18,608	18,662	18,834
106	4,285	5,396	10,666	13,782	15,599	15,496	16,169	16,704	
3,410	8,992	13,873	16,141	18,735	22,214	22,863	23,466		
5,655	11,555	15,766	21,266	23,425	26,083	27,067			
1,092	9,565	15,836	22,169	25,955	26,180				
1,513	6,445	11,702	12,935	15,852					
557	4,020	10,946	12,314						
1,351	6,947	13,112							
3,133	5,395								
2,063									

Using resteval and rapply

Mack.ByOrigin Cell D22=RSetEval("Mack.ByOrigin","function(x) summary(MackChainLadder(x, tail=TRUE))$ByOrigin")

Latest	Dev	Ultimate	IBNR	SE	CoV
18,834	98%	19,169	335	267	80%
16,704	97%	17,158	454	288	63%
23,466	96%	24,512	1,046	680	65%
27,067	93%	29,214	2,147	807	38%
26,180	89%	29,442	3,262	1,520	47%
15,852	80%	19,848	3,996	2,049	51%
12,314	68%	18,065	5,751	2,258	39%
13,112	54%	24,447	11,335	5,459	48%
5,395	33%	16,331	10,936	6,449	59%
2,063	11%	18,730	16,667	25,005	150%

Cell D25:I34={RApply(D22,D8:M17)}

Using the COM server (VBA Example)

StatConnector allows to use R within MS Office VBA

Add reference to **StatConnectorSrv** 1.1 Type Library

```
Sub FirstR()
 Dim nrandom As Integer, x As Double
 nrandom = 100
 Set StaR = New StatConnector
 StaR.Init ("R")
   With StaR
    .SetSymbol "n", nrandom
   .EvaluateNoReturn ("x <- rnorm(n)")
   .EvaluateNoReturn ("pdf(file='c:/Temp/Testplot.pdf')")
   .EvaluateNoReturn ("hist(x)")
   .EvaluateNoReturn ("dev.off()")
   x = .Evaluate("mean(x)")
 End With
 Debug.Print x
End Sub
```

rcom: Control MS Office from R

- Using the *rcom* R-package you can write output from R into MS Office application
  - Example: Create PowerPoint slide with *MackChainLadder* output

```
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)
```

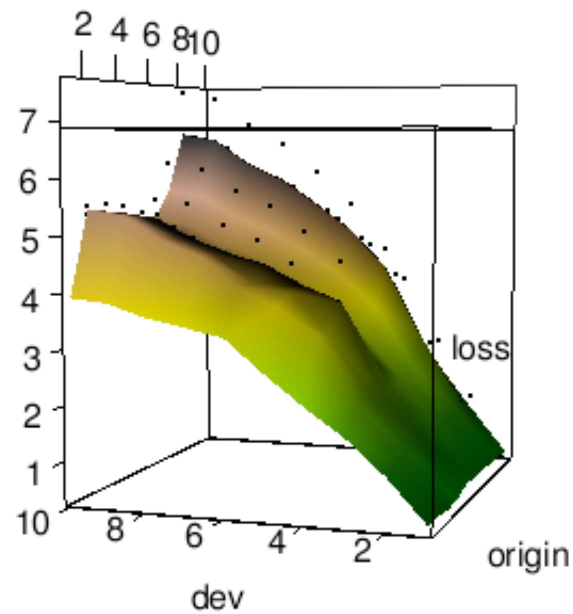
More help ...

- See examples 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
  - Easy to set-up connections to databases (ODBC)
  - RExcel add-in allows to share R functions with colleagues without R knowledge
  - *rcom* allows to control MS Office from R
  - Effective knowledge transfer - plain text files

For a laugh - fancy 3d plot




```
library(rgl) #provides interactive 3d plotting functions
MCL=MackChainLadder(GenIns/1e6)
FT <- MCL$FullTriangle
FTpSE <- FT+MCL$Mack.S.E
FTpSE[which(MCL$Mack.S.E==0, arr.ind=TRUE)] <- NA
FTmSE <- FT-MCL$Mack.S.E
FTmSE[which(MCL$Mack.S.E==0, arr.ind=TRUE)] <- NA
zr <- round(FT/FT[1,10]*100)
zlim <- range(zr, na.rm=TRUE)
zlen <- zlim[2] - zlim[1] + 1
colorlut <- terrain.colors(zlen) # height color lookup table
cols <- colorlut[ zr -zlim[1]+1 ] # assign colors to heights for each point
x <- as.numeric(dimnames(FT)$origin)
y <- as.numeric(dimnames(FT)$dev)
persp3d(x, y=y,
        z=(FT), col=cols, xlab="origin", ylab="dev", zlab="loss",back="lines")
mSE <- data.frame(as.table(FTmSE))
points3d(xyz.coords(x=as.numeric(as.character(mSE$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)
```

References

- Thomas Mack. Distribution-free calculation of the standard error of chain ladder reserve estimates. Astin Bulletin. Vol. 23. No 2. 1993. pp 213-225.
- Thomas Mack. The standard error of chain ladder reserve estimates: Recursive calculation and inclusion of a tail factor. Astin Bulletin. Vol. 29. No 2. 1999. pp 361-366.
- Murphy, Daniel M. Unbiased Loss Development Factors. Proceedings of the Casualty Actuarial Society Casualty Actuarial Society - Arlington, Virginia 1994: LXXXI 154-222.
- Zehnwirth and Barnett. Best estimates for reserves. Proceedings of the CAS, LXXXVI I(167), November 2000.
- P.D.England and R.J.Verrall, Stochastic Claims Reserving in General Insurance, British Actuarial Journal, Vol. 8, pp.443-544, 2002.
- Gerhard Quarg and Thomas Mack. Munich Chain Ladder. Blätter DGVFM 26, Munich, 2004.
- Nigel De Silva. An Introduction to R: Examples for Actuaries. Actuarial Toolkit Working Party, version 0.1 edition, 2006. http://toolkit.pbwiki.com/RToolkit.