Reserving In R

A Practical Approach (Non Life)

Cyril Adoh (adocyrilazuka@gmail.com)

Contents

Actuarial Automation

Actuarial Automation

…We are stuck in using Excel similar to how actuaries in the old days were stuck using logbooks. We need to make peace with the fact that we cannot visualize all the calculations and every part of the process like with using Excel… —Adriaan Rowan

Backstory

Backstory

- Motivation
- How do we do it today?
- Other use cases. (RI optimisation, Large Loss Modelling, EC Modelling, ALM, IFRS17 cashflows, Portfolio Analytics)
- What skills are required?
  - Strong actuarial knowledge (!important)
  - Programming skills
  - Patience
-

Package ‘*ChainLadder*’

Package '*ChainLadder*'

- Standard definition: The `ChainLadder` package provides various statistical methods which are typically used for the estimation of outstanding claims reserves in general insurance.
- Methods available are: `MackChainLadder`, `MunichChainLadder`, `Bootstrap`, GLM etc.
- Also contains useful functions `incr2cum`, `cum2incr`, `ata` etc.

`ChainLadder` package example

- In this example, we will use `RAA` data that come with the `ChainLadder` package to illustrate its features.

```
1 library(ChainLadder) #load the chainladder package
2 RAA #Run-off triangle of Automatic Factultative business in General Liability
```

```
      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
```

`ChainLadder` package example

- Use `cum2incr` to convert from cumulative to incremental

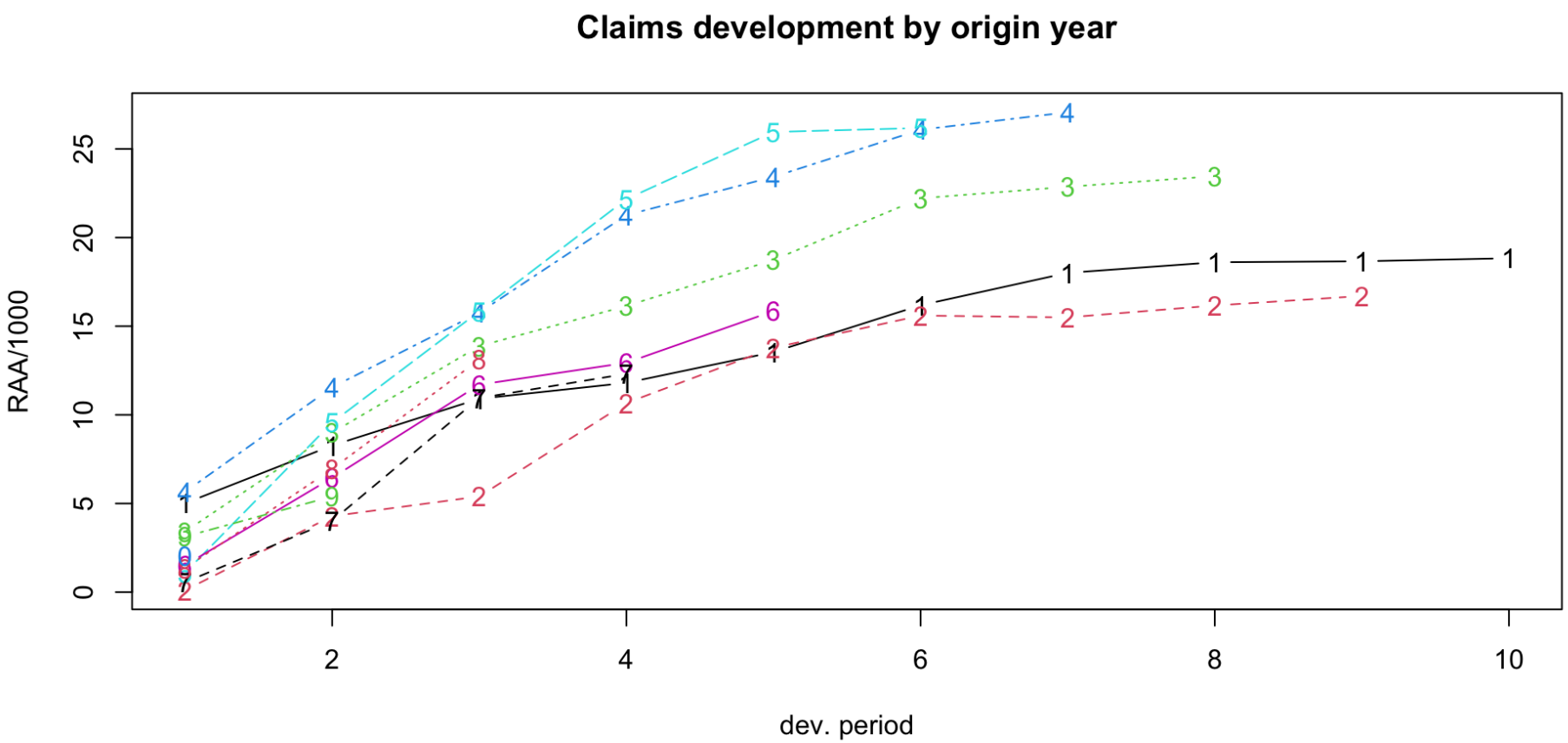
```
1  cum2incr(RAA)
```

```
      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
```

`ChainLadder` package example

- use `plot` to plot the triangle. Specify `lattice = T` to plot in lattice.

```
1  plot(RAA/1000,  main = "Claims development by origin year")
```



ChainLadder package example

- See the age-to-age factors

```
1  ata(RAA)
```

```
      dev
origin    1-2   2-3   3-4   4-5   5-6   6-7   7-8   8-9  9-10
  1981  1.650 1.319 1.082 1.147 1.195 1.113 1.033 1.003 1.009
  1982 40.425 1.259 1.977 1.292 1.132 0.993 1.043 1.033    NA
  1983  2.637 1.543 1.163 1.161 1.186 1.029 1.026    NA    NA
  1984  2.043 1.364 1.349 1.102 1.113 1.038    NA    NA    NA
  1985  8.759 1.656 1.400 1.171 1.009    NA    NA    NA    NA
  1986  4.260 1.816 1.105 1.226    NA    NA    NA    NA    NA
  1987  7.217 2.723 1.125    NA    NA    NA    NA    NA    NA
  1988  5.142 1.887    NA    NA    NA    NA    NA    NA    NA
  1989  1.722    NA    NA    NA    NA    NA    NA    NA    NA
  smpl  8.206 1.696 1.315 1.183 1.127 1.043 1.034 1.018 1.009
  vwtd  2.999 1.624 1.271 1.172 1.113 1.042 1.033 1.017 1.009
```

- Compute the weighted average factor use the code below.

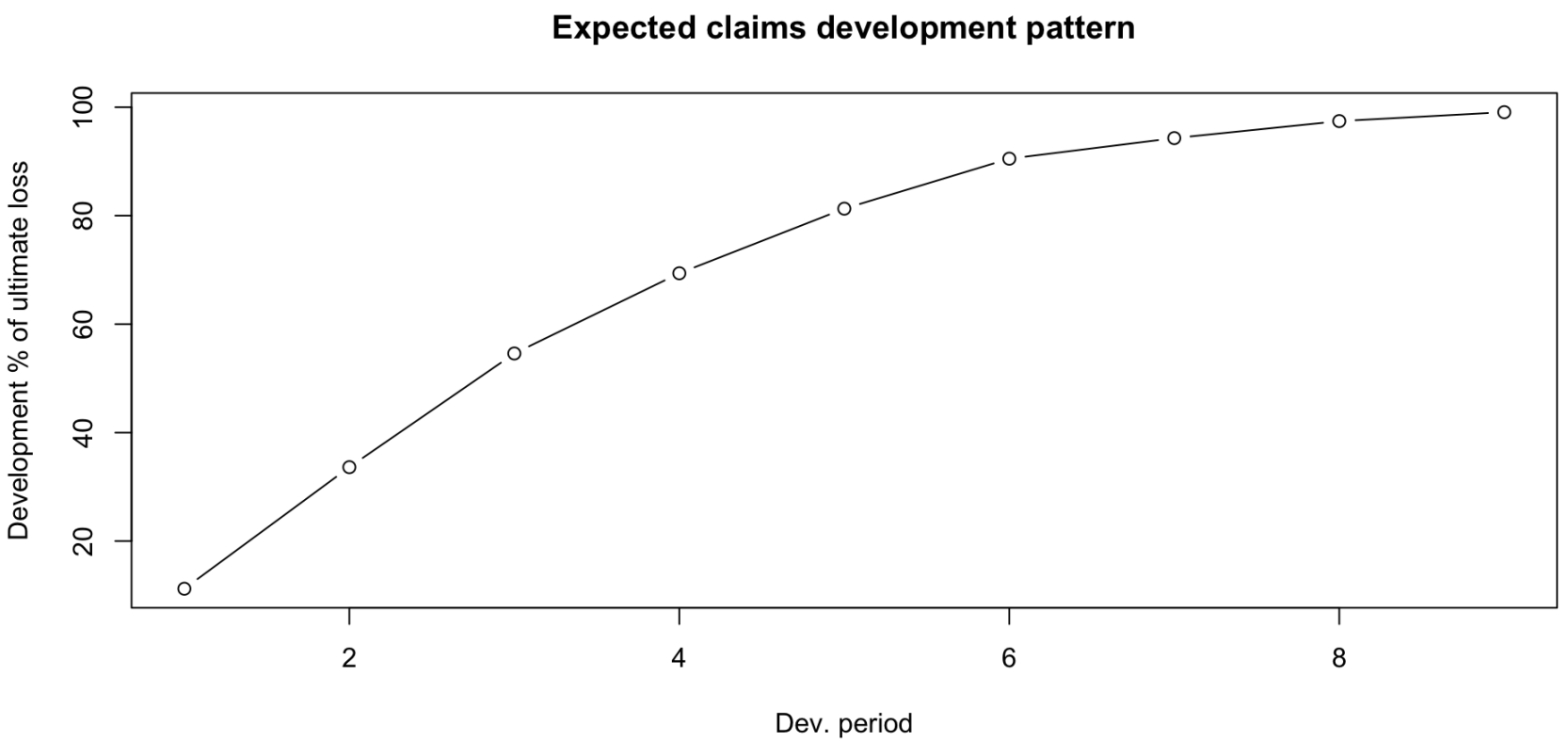
```
1  n <- ncol(RAA)
2  f <- sapply(1:(n-1),
3              function(i){
4                sum(RAA[c(1:(n-i)),i+1])/sum(RAA[c(1:(n-i)),i])
5              }
6  )
7  f
```

```
[1] 2.999359 1.623523 1.270888 1.171675 1.113385 1.041935 1.033264 1.016936
[9] 1.009217
```

`ChainLadder` package example

- Lets `plot f` to see the rate of development for this claims process.

```
plot(100*(rev(1/cumprod(rev(f)))), t="b",
     main="Expected claims development pattern",
     xlab="Dev. period", ylab="Development % of ultimate loss")
```



ChainLadder package example

- IBNR for RAA?

```
library(tidyverse)
library(scales)
currentEval <- getLatestCumulative(RAA)
LDF <- cumprod(rev(c(f,1)))
EstdUlt <- currentEval * LDF #
# Start with the body of the exhibit
Exhibit <- data.frame(currentEval, LDF = round(LDF, 3), EstdUlt) %>% mutate(IBNR =  EstdUlt - currentEval)
# Tack on a Total row
Exhibit <- rbind(Exhibit,
                 data.frame(currentEval=sum(currentEval), LDF=NA, EstdUlt=sum(EstdUlt), IBNR = sum(Exhibit$IB
                            row.names = "Total"))

Exhibit <- Exhibit %>% mutate(currentEval = formatC(currentEval, big.mark = ","), EstdUlt = format(round(Estd

Exhibit
```

```
      currentEval   LDF EstdUlt   IBNR
1981       18,834 1.000  18,834      0
1982       16,704 1.009  16,858    154
1983       23,466 1.026  24,083    617
1984       27,067 1.060  28,703  1,636
1985       26,180 1.105  28,927  2,747
1986       15,852 1.230  19,501  3,649
1987       12,314 1.441  17,749  5,435
1988       13,112 1.832  24,019 10,907
1989        5,395 2.974  16,045 10,650
1990        2,063 8.920  18,402 16,339
Total     160,987    NA 213,122 52,135
```

`ChainLadder` package example

- Complete triangle

```
f <- c(f, 1)
fullRAA <- cbind(RAA, Ult = rep(0, 10))
for(k in 1:n){
  fullRAA[(n-k+1):n, k+1] <- fullRAA[(n-k+1):n,k]*f[k]
}
round(fullRAA) #Run-off triangle of Automatic Factultative business in General Liability
```

```
        1     2     3     4     5     6     7     8     9    10   Ult
1981 5012  8269 10907 11805 13539 16181 18009 18608 18662 18834 18834
1982  106  4285  5396 10666 13782 15599 15496 16169 16704 16858 16858
1983 3410  8992 13873 16141 18735 22214 22863 23466 23863 24083 24083
1984 5655 11555 15766 21266 23425 26083 27067 27967 28441 28703 28703
1985 1092  9565 15836 22169 25955 26180 27278 28185 28663 28927 28927
1986 1513  6445 11702 12935 15852 17649 18389 19001 19323 19501 19501
1987  557  4020 10946 12314 14428 16064 16738 17294 17587 17749 17749
1988 1351  6947 13112 16664 19525 21738 22650 23403 23800 24019 24019
1989 3133  5395  8759 11132 13043 14521 15130 15634 15898 16045 16045
1990 2063  6188 10046 12767 14959 16655 17353 17931 18234 18402 18402
```

`ChainLadder` package example

- `BootChainLadder` by England and Verrall (England and Verrall 2002)
- Predictive distribution of reserves
- use `quantiles`
- IFRS17 RA (VAR method)

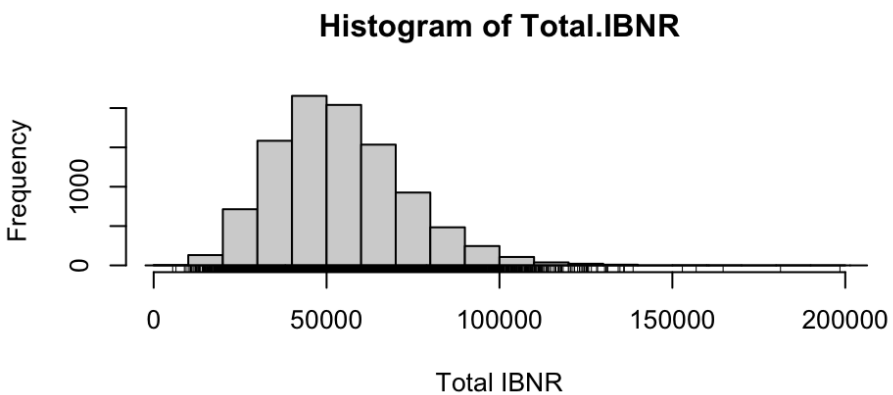
```
raa.boot <- BootChainLadder(RAA, 10000, "od.pois")
quantile(raa.boot, c(0.5,0.6,0.75,0.95))
```

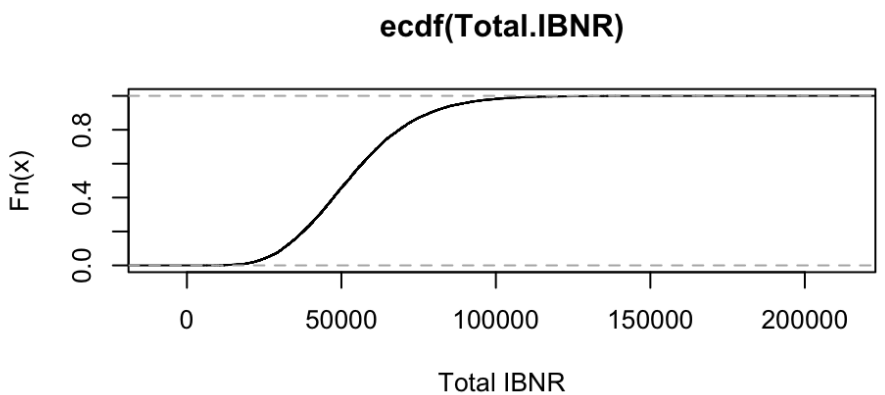
```
$ByOrigin
     IBNR 50% IBNR 60% IBNR 75% IBNR 95%
1981      0.0      0.0     0.00     0.00
1982      1.0     22.0   199.00  1382.20
1983    318.0    567.0  1147.00  3092.10
1984   1306.0   1746.4  2624.00  5255.00
1985   2426.0   2971.4  4091.25  7181.05
1986   3296.5   3938.0  5098.50  8431.05
1987   5048.0   5847.2  7291.75 11363.05
1988  10478.5  11747.4 14122.25 20308.30
1989  10171.5  11745.4 14641.25 21992.20
1990  15363.5  18654.4 24467.50 42708.35

$Totals
           Totals
IBNR 50%: 52101.5
IBNR 60%: 56757.0
IBNR 75%: 64964.5
IBNR 95%: 87915.5
```

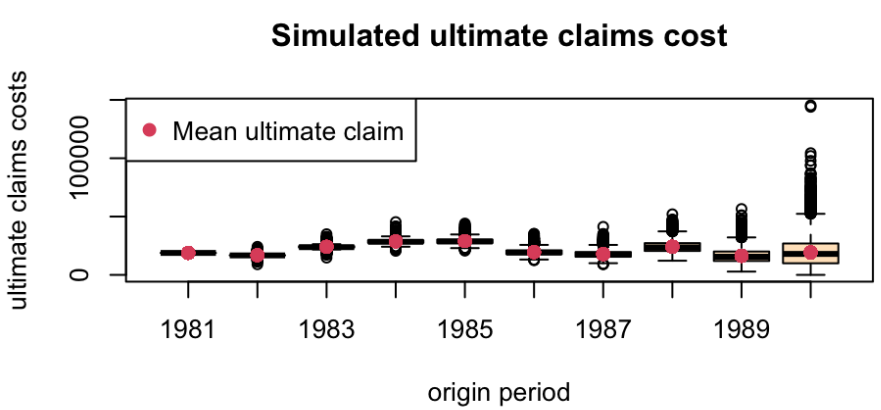
ChainLadder package example

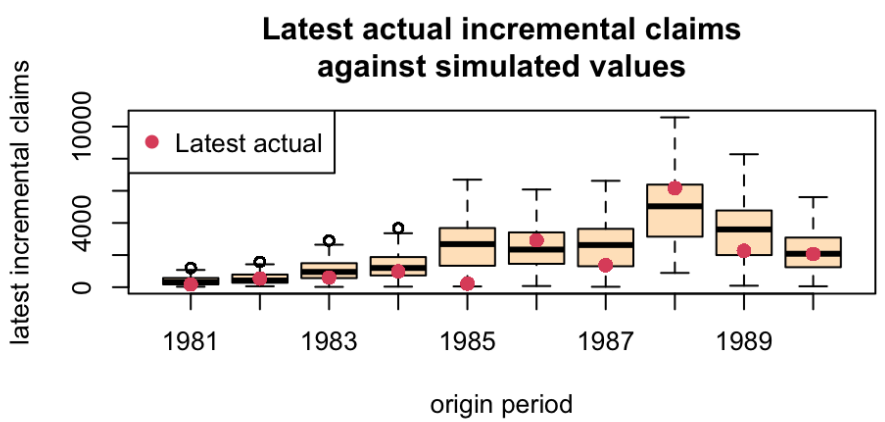
- plot raa.boot

```
1  plot(raa.boot)
```









`ChainLadder` package discussion

- More information on the method underlying the `BootChainLadder`
- `ChainLadder` requires imported triangles (wide/long).
- Use other methods e.g BF, LR, CapeCod etc for specific cohort. Not available in the package yet.
- Apply weights from specific section of the triangle for LDF computation.
- Our goal is to automate the entire process in R. The `ChainLadder` package will help with about 10%.
- You have to take control.

Take Control

Take Control

- Import data into R from Excel or Database or `.RData`. More info here.
- Take advantage of `tidyverse` package to clean your data and put in right format for use.
- Summary statistics.
- For Non-life reserving, we need the **exposure** database and **claims** database.
  - Do a reconciliation to external sources e.g ledger or revenue account.
  - Use exposure data to computes, EP, UPR, DAC etc.
  - Use Claims data to generate incremental triangles from claims data as required.
  - Generate cumulative and compute `ata` factors if necessary. Selected required LDF.
  - Write functions to help generate IBNR as needed.
- Generate IBNR in a few minutes.

More Control with '*Shiny*'

Oh! I love `shiny`

- Yes, you can visualise your work.
- See docs on `shiny` here. Also `shinydashboard` here
- Initial investment is required. Walk in the park afterwards.
- Write your reports with RMarkdown.
- Design your presentation with Quarto.

Wrap Up, Resources and References

- Code documentation, version control and collaboration with GitHub
- ChainLadder
- England and Verrall 2002 paper.
- Claims Reserving in General Insurance (SP7) - David Hindley
- Learn `tidyverse`, SQL (use SQL in R), GitHub, `shiny`, `shinydashboard`.
- Source code: DavidHindley, `ChainLadder`
- ReservingInR_NAS_CPD repository.

Thank you.