A new R-package for statistical modelling and forecasting in non-life insurance

Maria Dolores Martinez-Miranda
Jens Perch Nielsen
Richard Verrall
Our aim: a package implementing recent research developments

2010 Including Count Data in Claims Reserving
2011 Cash flow simulation for a model of outstanding liabilities based on claim amounts and claim numbers
2012 Double Chain Ladder

2012 Statistical modelling and forecasting in Non-life insurance
2013 Double Chain Ladder and Bornhuetter-Ferguson
2013 Double Chain Ladder, Claims Development Inflation and Zero Claims
2013 Continuous Chain Ladder
The problem: the claims reserving exercise

- Claims are first notified and later settled - **reporting** and **settlement delays** exist.
- **Outstanding liability** for claims events that have already happened and for claims that have not yet been fully settled.
- The objectives:
  - How large **future claims payments** are likely to be.
  - The **timing** of future claim payments.
  - The **distribution** of possible outcomes: future **cash-flows**.
Framework: Double Chain Ladder

What is Double Chain Ladder?

A firm statistical model which breaks down the chain ladder estimates into individual components.

Why?

✓ Connection with classical reserving (tacit knowledge)
✓ Intrinsic tail estimation
✓ RBNS and IBNR claims
✓ The distribution: full cash-flow

What is required? It works on run-off triangles (adding expert knowledge if available).
The modelled data: two run-off triangles

We model annual/quarterly run-off triangles:

- **Incremental aggregated payments** (paid triangle).

- **Incremental aggregated counts data**, which is assumed to have fully run off.
The Double Chain Ladder Model

**Parameters** involved in the model:

- **Ultimate claim numbers:** $\alpha_i$
- **Reporting delay:** $\beta_j$
- **Settlement delay:** $\pi_l$
- **Development delay:** $\tilde{\beta}_j$
- **Ultimate payment numbers:** $\tilde{\alpha}_i$
- **Severity:**
  - ✓ **underwriting inflation:** $\gamma_i$
  - ✓ **delay mean dependencies:** $\tilde{\mu}_{jl}$

![Diagram showing parameters and dependencies in the Double Chain Ladder Model]
Implementing Double Chain Ladder

The kernel: calibrating the model

Data

Expert knowledge

Full cash-flow (RBNS/IBNR)

Best estimate (RBNS/IBNR)
### Visualizing the data: the histogram

#### Payment data

<table>
<thead>
<tr>
<th>Accident</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Counts data

<table>
<thead>
<tr>
<th>Accident</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The kernel: calibrating the model

- The available information could make a model infeasible in practice.

- From two run-off triangles, the **Double Chain Ladder Method** estimate a model such as:
  
  severity mean: \( \mu_{\gamma_i} \equiv \mu_i \)
  
  severity variance: \( \sigma_i^2 \gamma_i^2 \equiv \sigma_i^2 \)

- **Classical chain ladder technique** is applied twice to give everything needed to estimate.
The kernel: parameter estimation using DCL

- The function `dcl.estimation()`

```r
dcl.estimation [DCL]

Parameter estimation - Double Chain Ladder model

Description
Compute the estimated parameters in the model (delay parameters, severity underwriting inflation, severity mean and variance) using the Double Chain Ladder method.

Usage
`dcl.estimation(Xtriangle, Ntriangle, adj = 1, Tables = TRUE, num.dec = 4)`

Arguments
- `Xtriangle` The paid run-off triangle: incremental aggregated payments. It should be a matrix with incremental aggregated payments located in the upper triangle and the lower triangle consisting in missing or zero values.
- `Ntriangle` The counts data triangle: incremental number of reported claims. It should be a matrix with the observed counts located in the upper triangle and the lower triangle consisting in missing or zero values. It should have the same dimension as `Xtriangle` (both in the same aggregation level (quarters, years, etc.)).
- `adj` Method to adjust the estimated delay parameters for the distributional model. It should be 1 (default value) or 2. See more in details below.
- `Tables` Logical. If `TRUE` (default) it is showed a table with the estimated parameters.
- `num.dec` Number of decimal places used to report numbers in the tables (if `Tables=TRUE`).

Settlement delay

Severity mean: $\gamma_i \times \mu$

variance: $\gamma_i^2 \times \sigma^2$

Counts

Paid

$\alpha_i$

$\beta_{ji}$

$\beta_{ij}$

$\pi_i$

$\tilde{\alpha}_i$

$\gamma_i$

The kernel: parameter estimation using DCL

- The function `plot.dcl.par()` to visualize the break down of the classical chain ladder parameters

```r
plot.dcl.par(DCL)
```

**Plotting the estimated parameters in the DCL model**

**Description**
Show a two by two plot with the estimated parameters in the Double Chain Ladder model

**Usage**

```r
plot.dcl.par(dcl.par, type.inflat = 'DCL')
```

**Arguments**

- `dcl.par` A list object with the estimated parameters: the value returned by the functions `dcl.estimation`, `bdcl.estimation` or `idcl.estimation`
- `type.inflat` Method used to estimate the inflation. Possible values are: 'DCL' (default) if it was used `dcl.estimation`, 'BDCL' if `bdcl.estimation`, and 'IDCL' if `idcl.estimation`

**Documentation**

Severity mean: $\gamma_i \times \mu$

Variance: $\gamma_i^2 \times \sigma^2$
The functions in action: an example

Parameter estimates in two cases: the basic DCL model (only mean specifications) and the distributional model.
The best estimate: RBNS/IBNR split
The best estimate: RBNS/IBNR split using DCL

The function **dcl.predict()**

dcl.predict (DCL)

Pointwise predictions (RBNS/IBNR split)

**Description**

Pointwise predictions by calendar years and rows of the outstanding liabilities. The predictions are split between RBNS and IBNR claims.

**Usage**

dcl.predict( dcl.par, Ntriangle, Model = 2, Tail = TRUE, Tables = TRUE, summ.by="diag", num.dec = 2 )

**Arguments**

dcl.par A list object with the estimated parameters. the value returned by the functions dcl.estimation, bocl.estimation or idcl.estimation.
Ntriangle Optional. The counts data triangle: incremental number of reported claims. It should be a matrix with the observed counts located in the upper triangle and the lower triangle consisting of missing or zero values. It should have the same dimension as the Xtriangle (both in the same aggregation level (quarters, years, etc.) used to derive dcl.par.
Model Possible values are 0, 1 or 2 (default). See more details below.
Tail Logical. If TRUE (default) the tail is provided.
Tables Logical. If TRUE (default) it is shown a table with the predicted outstanding liabilities in the future calendar periods (summar.by="diag") or by underwriting period (summar.by="row").
summ.by A character value such as "diag", "row" or "cell".
num.dec Number of decimal places used to report numbers in the tables. Used only if Tables=TRUE.

**Details**

If Model=0 or Model=1 then the predictions are calculated using the DCL model parameters in assumptions M1-M3 (general delay parameters, see Martinez-Miranda, Nielsen and Verrall 2012). If Model=2 the adjusted delay probabilities (distributional model D1-D4) are considered. By
The function in action: an example

<table>
<thead>
<tr>
<th>Future.years</th>
<th>rbns</th>
<th>ibnr</th>
<th>total</th>
<th>clm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>59845052.96</td>
<td>1386631.90</td>
<td>61231684.86</td>
<td>61090912.9</td>
</tr>
<tr>
<td>2</td>
<td>41447058.01</td>
<td>7405875.89</td>
<td>48852933.90</td>
<td>48061354.9</td>
</tr>
<tr>
<td>3</td>
<td>31016097.53</td>
<td>5610771.34</td>
<td>36626868.87</td>
<td>36266481.8</td>
</tr>
<tr>
<td>4</td>
<td>17542089.42</td>
<td>5501517.13</td>
<td>23043606.55</td>
<td>22989797.0</td>
</tr>
<tr>
<td>5</td>
<td>6443018.76</td>
<td>4069044.13</td>
<td>10512062.89</td>
<td>10439464.1</td>
</tr>
<tr>
<td>6</td>
<td>3192176.74</td>
<td>1719910.74</td>
<td>4912087.48</td>
<td>4913941.1</td>
</tr>
<tr>
<td>7</td>
<td>1445598.60</td>
<td>944953.87</td>
<td>2390552.47</td>
<td>2380120.6</td>
</tr>
<tr>
<td>8</td>
<td>675017.48</td>
<td>486952.87</td>
<td>1161970.35</td>
<td>1174086.8</td>
</tr>
<tr>
<td>9</td>
<td>642274.45</td>
<td>210295.79</td>
<td>852570.24</td>
<td>848055.6</td>
</tr>
<tr>
<td>10</td>
<td>423522.65</td>
<td>168593.53</td>
<td>592116.19</td>
<td>599855.7</td>
</tr>
<tr>
<td>11</td>
<td>535548.94</td>
<td>72125.43</td>
<td>607674.37</td>
<td>593718.3</td>
</tr>
<tr>
<td>12</td>
<td>404459.01</td>
<td>99337.90</td>
<td>503796.92</td>
<td>495023.4</td>
</tr>
<tr>
<td>13</td>
<td>334964.95</td>
<td>74405.59</td>
<td>409370.54</td>
<td>397094.7</td>
</tr>
<tr>
<td>14</td>
<td>60022.99</td>
<td>96886.33</td>
<td>156909.31</td>
<td>135553.4</td>
</tr>
<tr>
<td>15</td>
<td>0.00</td>
<td>37035.26</td>
<td>37035.26</td>
<td>109484.7</td>
</tr>
<tr>
<td>16</td>
<td>0.00</td>
<td>12228.15</td>
<td>12228.15</td>
<td>0.0</td>
</tr>
<tr>
<td>17</td>
<td>0.00</td>
<td>6545.30</td>
<td>6545.30</td>
<td>0.0</td>
</tr>
<tr>
<td>18</td>
<td>0.00</td>
<td>3691.79</td>
<td>3691.79</td>
<td>0.0</td>
</tr>
<tr>
<td>19</td>
<td>0.00</td>
<td>1831.78</td>
<td>1831.78</td>
<td>NA</td>
</tr>
<tr>
<td>20</td>
<td>0.00</td>
<td>1013.15</td>
<td>1013.15</td>
<td>NA</td>
</tr>
<tr>
<td>21</td>
<td>0.00</td>
<td>518.55</td>
<td>518.55</td>
<td>NA</td>
</tr>
</tbody>
</table>

Summary by diagonals (future calendar years), rows (underwriting) and the individual cell predictions.
The **simplest DCL distributional model** assumes that the mean and the variance of the individual payments (severity) only depends on the underwriting period.

The following statistical distributions are assumed for each of the components in the model:

<table>
<thead>
<tr>
<th>Component</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count data</td>
<td><strong>Poisson</strong></td>
</tr>
<tr>
<td>RBNS delay</td>
<td><strong>Multinomial</strong></td>
</tr>
<tr>
<td>Severity</td>
<td><strong>Gamma</strong></td>
</tr>
</tbody>
</table>
The full cash-flow: Bootstrapping using DCL

- The function `dcl.boot()`

```r
dcl.boot {DCL}

R Documentation

Bootstrap distribution: the full cashflow

Description
Provide the distribution of the IBNR, RBNS and total (RBNS+IBNR) reserves by calendar years and rows using bootstrapping.

Usage
`dcl.boot( dcl.par, sigma2, Ntriangle, boot.type = 2, B = 999, Tail = TRUE, summ.by = "diag", Tables = TRUE, num.dec = 2 )`

Arguments
- `dcl.par` A list object with the estimated parameters: the value returned by the functions `dcl.estimation, bdl.estimation` or `idcl.estimation`
- `sigma2` Optional. The variance of the individual payments in the first underwriting period.
- `Ntriangle` The counts data triangle. Incremental number of reported claims. It should be a matrix with the observed counts located in the upper triangle and the lower triangle consisting in missing or zero values. It should be the same triangle used to get the value passed by the argument `dcl.par`.
- `boot.type` Choose between values 1, to provide only the variance process, or 2 (default), to take into account the uncertainty of the parameters.
- `B` The number of simulations in the bootstrap algorithm. The default value is 999.
- `Tail` Logical. If TRUE (default) the tail is provided.
- `summ.by` A character value such as "diag", "row" or "cell".
- `Tables` Logical. If TRUE (default) it is showed a table with the summary (mean, standard deviation, 1%, 5%, 95%, 99%) of the distribution of the outstanding liabilities in the future calendar periods (if `summ.by="diag"`) or by underwriting period (if `summ.by="row"`).
- `num.dec` Number of decimal places used to report numbers in the tables. Used only if `Tables=TRUE`

Details
```

- The function `plot.cashflow()`
The functions in action: an example

- A table showing a summary of the distribution: mean, std. deviation, quantiles.
- Arrays and matrices with the full simulated distributions
The functions in action: an example

R Console

> names(boot2)
[1] "array.rbns.boot" "array.ibnr.boot"
[3] "Mat.rbns" "Mat.ibnr"
[5] "Mat.total"
> plot.cashflow(boot2)
Prior knowledge, when it is available, can be incorporated to:

- Provide more realistic and stable predictions: Bornhuetter-Ferguson technique and the incurred data

- Consider in practice more general models: development severity inflation, zero-claims etc.
Using incurred data through BDCL and IDCL

- The BDCL method takes a more realistic estimation of the inflation parameter from the incurred triangle

- The IDCL method makes a correction in the underwriting inflation to reproduce the incurred chain ladder reserve

**Summary:** BDCL and IDCL operate on 3 triangles and give a different reserve than the paid chain ladder. Both provide the full cash-flow (RBNS/IBNR)
BDCL and IDCL in the package

- Functions `bdcl.estimation()` `idcl.estimation()`

- Validation strategy: `validating.incurred()`

R Console

```r
> three.inflations<-validating.incurred(nout=0,XtriangleBDCL
> three.inflations
$Inflat.DCL
[7] 2.249936 2.125084 1.902800 2.015875 2.070356 2.266601
[19] 6.750140

$Inflat.BDCL
[1] 1.000000 1.117293 1.495487 1.744521 2.107822 2.091391
[7] 2.239623 2.115821 1.867769 2.006702 2.050375 2.231534
[13] 2.306779 2.442709 2.310905 2.837465 2.494362 2.749805
[19] 2.859387

$Inflat.IDCL
[1] 1.000000 1.117293 1.494737 1.746090 2.4540246
[6] 0.8239007 0.1435635 0.7926218 0.2847205 0.7969141
[11] 0.6587015 -0.5239147 2.0509174 1.9786681 1.8410459
[16] 1.2695700 1.7693588 2.1597728 2.6702731
>
```
Validation

Testing results against experience:
1. Cut c=1,2,...,5 diagonals (periods) from the observed triangle.
2. Apply the estimation methods.
3. Compare forecasts and actual values.

Three objectives:
- Predictions of the individual cells
- Predictions by calendar years
- The prediction of the overall total
Validation strategy: validating.incurred()
Working in practice with a more general model

- Information about development severity, inflation, zero-claims etc. can be incorporated through DCL in a straightforward and coherent way.

- The package provides the functions:
  
  \[ \text{dcl.predict.prior()} \]
  
  \[ \text{dcl.boot.prior()} \]
  
  \[ \text{extract.prior()} \]
Summary: the content of the package

**Data**
- dcl.estimation
- bdcl.estimation
- idcl.estimation
- plot.dcl.par
- clm
- plot.clm.par
- 8 run-off triangles
  - plot.triangle
  - Aggregate, get.incremental, get.cumulative

**Expert knowledge**
- extract.prior

**The kernel: calibrating the model**
- dcl.predict
- dcl.predict.prior
- validating.incurred

**Full cash-flow (RBNS/IBNR)**
- dcl.boot
- dcl.boot.prior
- plot.cashflow

**Best estimate (RBNS/IBNR)**
- dcl.predict
- dcl.predict.prior
  - validating.incurred
We look for a **wide audience** (academics, practitioners, students).

The package has been published in the CRAN:

http://cran.r-project.org/web/packages/DCL/index.html

Your feedback is very valuable:

María Dolores Martínez-Miranda

-Maintainer of the DCL package-

mmiranda@ugr.es
library(DCL)
data(NtriangleBDCL)
data(XtriangleBDCL)

# Plotting the data
plot.triangle(NtriangleBDCL,Histogram=TRUE,tit=expression(paste('Counts: ',N[ij])))
plot.triangle(XtriangleBDCL,Histogram=TRUE,tit=expression(paste('Paid: ',X[ij])))

# The kernel: parameter estimation
my.dcl.par<-dcl.estimation(XtriangleBDCL,NtriangleBDCL)
plot.dcl.par(my.dcl.par)

# The best estimate (RBNS/IBNR split)
pred.by.diag<-dcl.predict(my.dcl.par,NtriangleBDCL)

# Full cashflow considering the tail (only the variance process)
boot2<-dcl.boot(my.dcl.par,Ntriangle=NtriangleBDCL)
plot.cashflow(boot2)

## Compare the three methods to be validated (three different inflations)
data(ItriangleBDCL)
validating.incurred(ncut=0,XtriangleBDCL,NtriangleBDCL,ItriangleBDCL)
test.res<-matrix(NA,4,10)
par(mfrow=c(2,2),cex.axis=0.9,cex.main=1)
for (i in 1:4)
{
  res<-validating.incurred(ncut=i,XtriangleBDCL,NtriangleBDCL,ItriangleBDCL,Tables=FALSE)
  test.res[i,]<-as.numeric(res$pe.vector)
}
test.res<-as.data.frame(test.res)
names(test.res)<-c("num.cut","pe.point.DCL","pe.point.BDCL","pe.point.IDCL",
"pe.calendar.DCL","pe.calendar.BDCL","pe.calendar.IDCL",
"pe.total.DCL","pe.total.BDCL","pe.total.IDCL")
print(test.res)

# Extracting information about severity inflation and zero claims
data(NtrianglePrior);data(NpaidPrior);data(XtrianglePrior)
extract.prior(XtrianglePrior,NpaidPrior,NtrianglePrior)
Appendix B: Bootstrap methods

**Algorithm RBNS** – Bootstrapping taking into account parameters uncertainty

- **Original data**: \( N_{ij}, X_{ij} \)
- **Bootstrapped data**: original counts and bootstrapped aggregated payments
- **Estimate the parameters**: \( \hat{\Theta} = (\hat{p}, \hat{\gamma}, \hat{\mu}, \hat{\sigma}^2) \)
- **Estimate the distributions**:
  - Delay: Multinomial with estimated \( \hat{p} \)
  - Severity: Gamma with estimated \( \hat{\mu} \) and \( \hat{\sigma}^2 \)
- **Calculate bootstrapped parameters**: \( \Theta^* = (p^*, \gamma^*, \mu^*, \sigma^2*) \)
- **Bootstrapped RBNS predictions**: \( X_{ij}^{rbns*} \)
- **Predictive RBNS distribution**:
  - \( X_{ij}^{rbns*(b)}, b = 1, \ldots, B \)
- **Simulating B times** from the distributions with bootstrapped parameters
Appendix B: Bootstrap methods

Algorithm IBNR – Bootstrapping taking into account parameters uncertainty

Original data

Estimate the parameters:

\[ \hat{\Theta} = (\hat{p}, \hat{y}_i, \hat{\mu}, \hat{\sigma}^2) \]

\[ \hat{\omega} = (R_i, F_j) \]

\[ \text{CLM} \rightarrow \hat{N} = N_{J_1}(\hat{\omega}) \]

IBNR predictions

Estimate the distributions:

- Delay: Multinomial with estimated \( \hat{p} \)
- Payments: Gamma with estimated \( \hat{\mu_i}, \hat{\sigma_i}^2 \)
- Counts: Poisson with means \( N_{ij} \)

Bootstrap data: original and bootstrapped counts and bootstrapped aggregated payments

Calculate bootstrapped parameters:

\[ \Theta^* = (p^*, \gamma_i^*, \mu^*, \sigma_{\gamma_i}^2) \]

\[ \omega^* = (R_i^*, F_j^*) \]

\[ \text{CLM} \rightarrow \hat{N}_{J_1}^*(\omega^*) \]

Bootstrapped IBNR predictions

Predictive IBNR distribution:

\[ \{ X_{ij}^{ibnr*(b)}, b = 1, \ldots, B \} \]

Simulating B times from the distributions with bootstrapped parameters