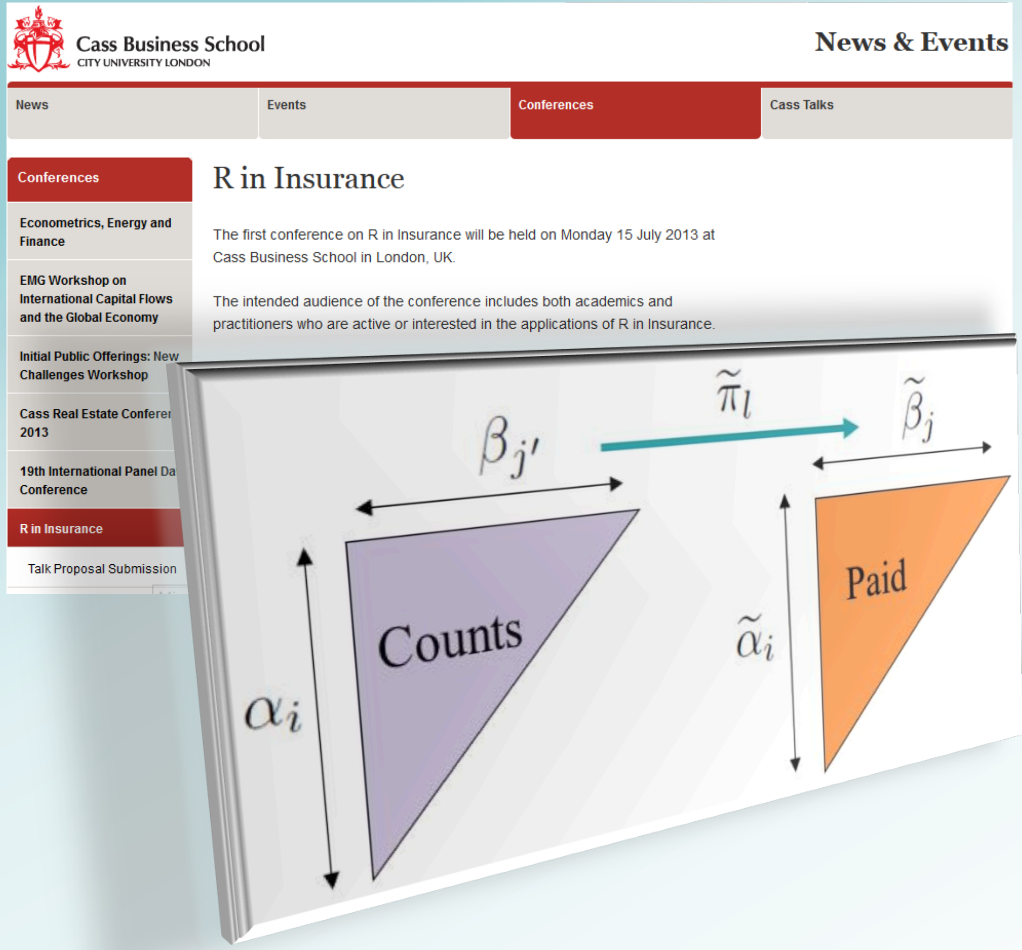


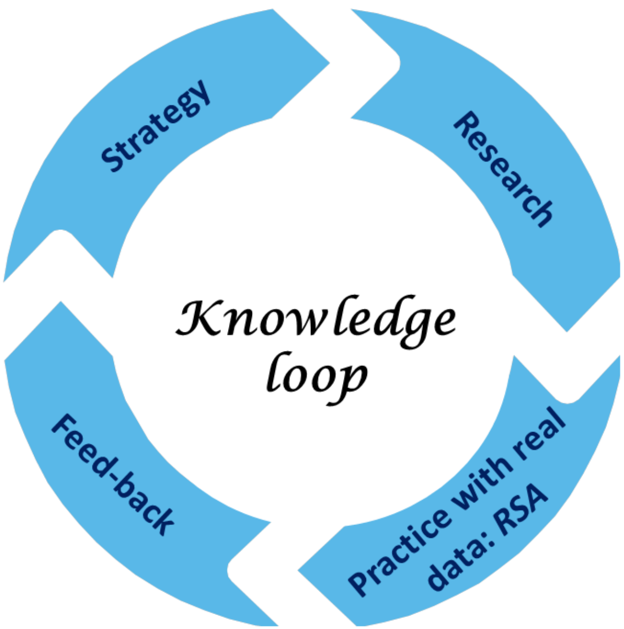
A new R-package for statistical modelling and forecasting in non-life insurance

María Dolores Martínez-Miranda
Jens Perch Nielsen
Richard Verrall

Cass Business School
London, 15 July 2013

Cass Business School
CITY UNIVERSITY LONDON

Background



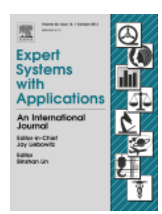
2010 Including Count Data in Claims Reserving

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

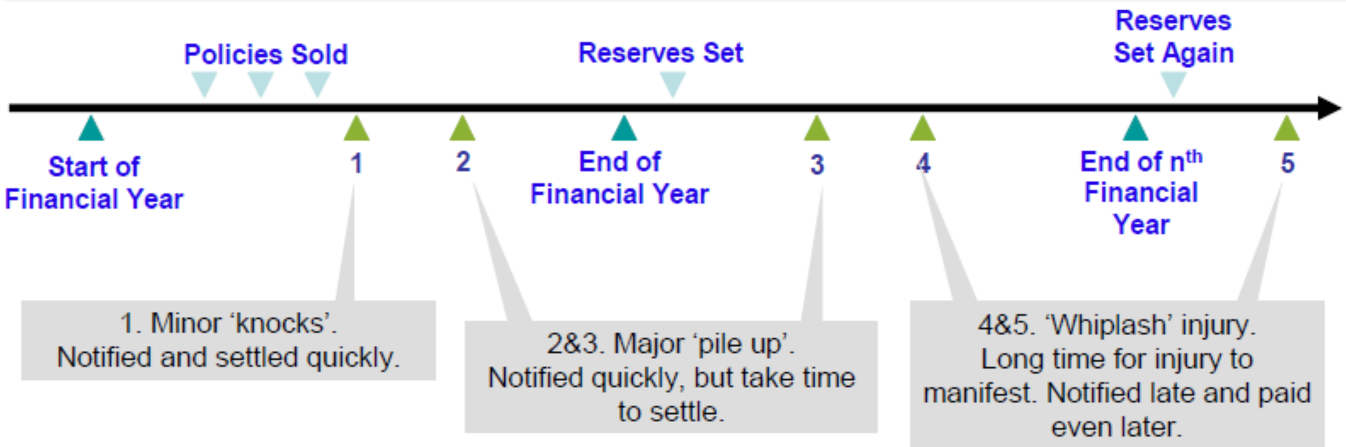


2013 Double Chain Ladder, Claims Development Inflation and Zero Claims

2013 Continuous Chain Ladder

Our aim: a package implementing recent research developments

The problem: the claims reserving exercise



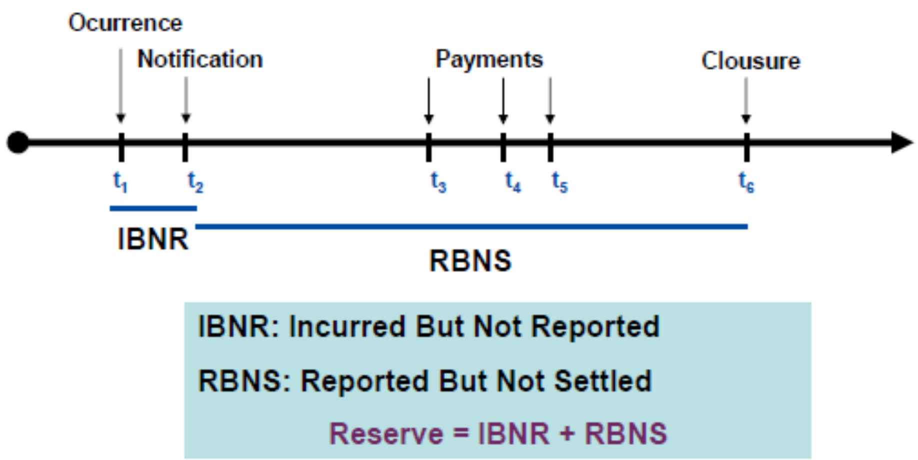
- Claims are first notified and later settled - **reporting** and **settlement delays** exist.
- **Outstanding liability** for claims events that have already happened and for claims that have not yet been fully settled.
- The objectives:
  - ✓ How large **future claims payments** are likely to be.
  - ✓ 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?



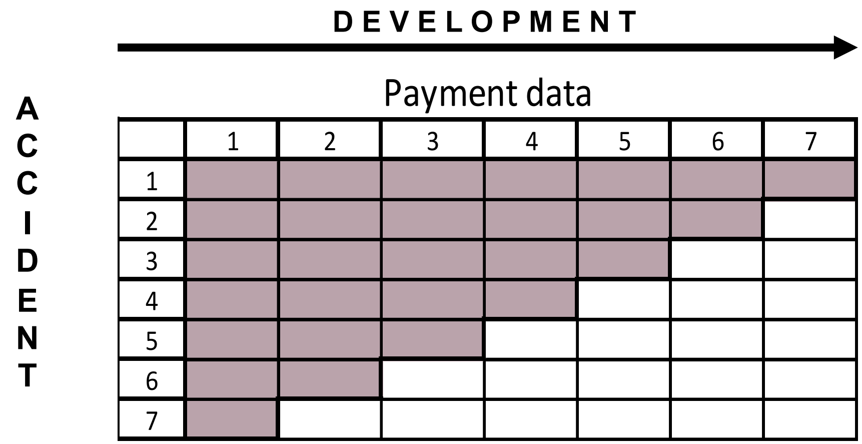
- ✓**Connection with classical reserving (tacit knowledge)**
- ✓Intrinsic **tail** estimation
- ✓**RBNS** and **IBNR** claims
- ✓The distribution: **full cash-flow**

What is required? It works on **run-off triangles (adding expert knowledge** if available).

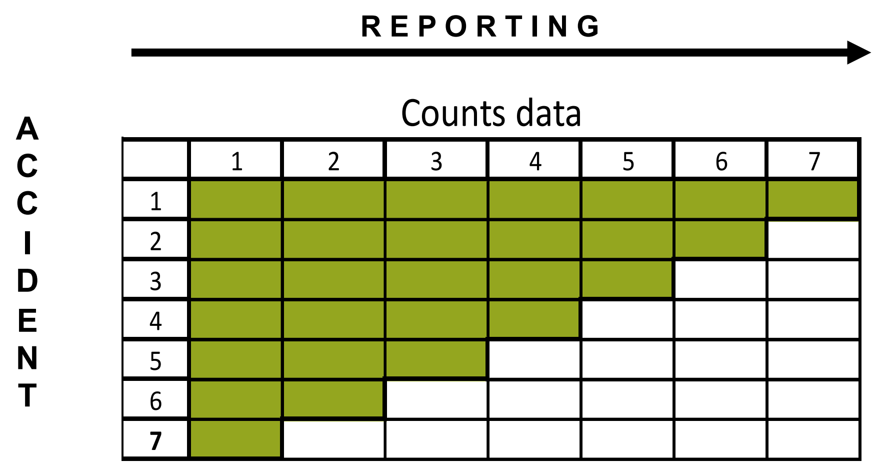
The modelled data: two run-off triangles

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: α_i

Reporting delay: $\beta_{j'}$

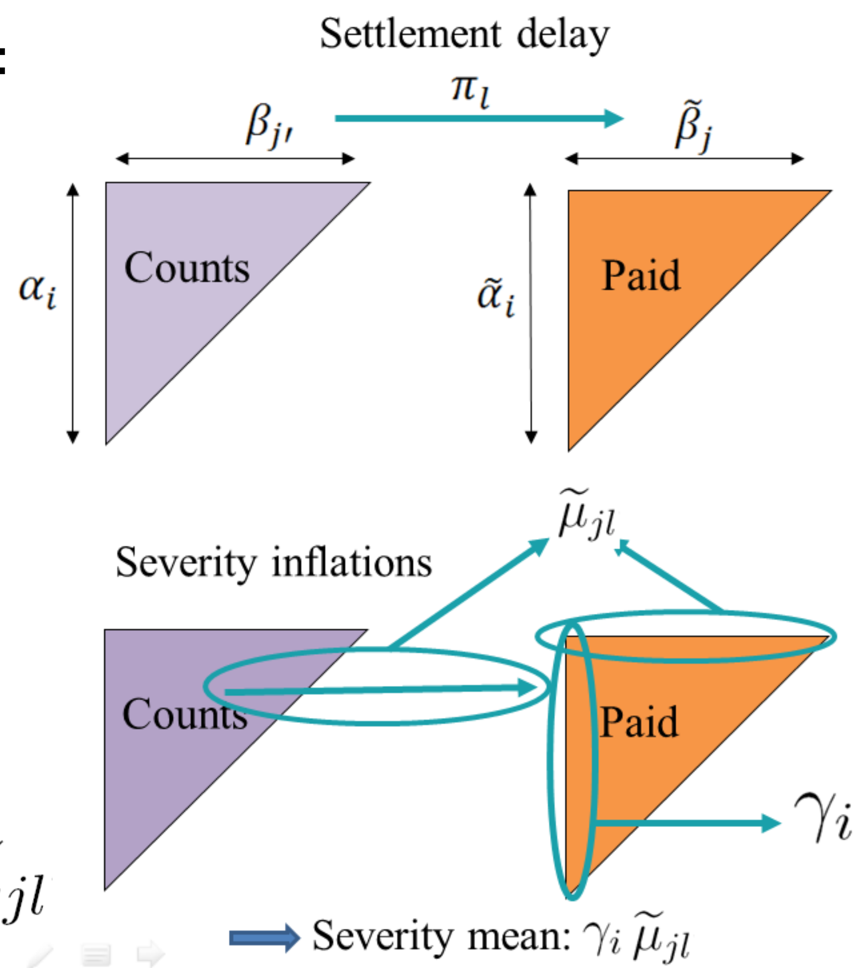
Settlement delay: π_l

Development delay: $\tilde{\beta}_j$

Ultimate payment numbers: $\tilde{\alpha}_i$

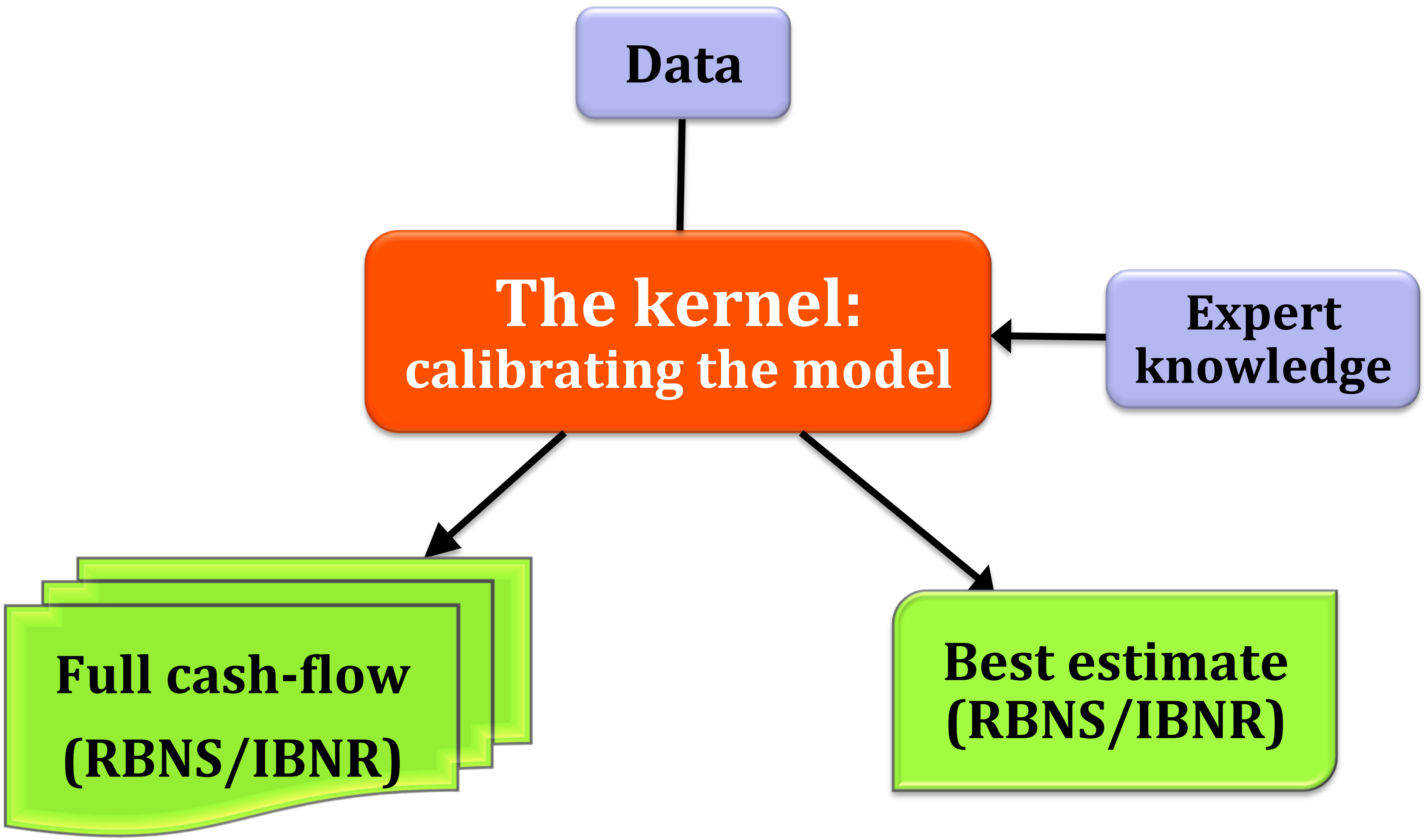
Severity:

- ✓ underwriting inflation: γ_i
- ✓ delay mean dependencies: $\widetilde{\mu}_{jl}$



Cass Business School
CITY UNIVERSITY LONDON

Implementing Double Chain Ladder



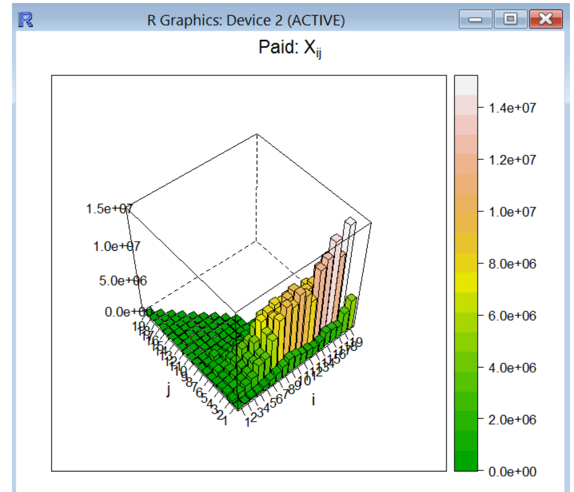
Visualizing the data: the histogram

DEVELOPMENT

ACCIDENT

Payment data

	1	2	3	4	5	6	7
1							
2							
3							
4							
5							
6							
7							

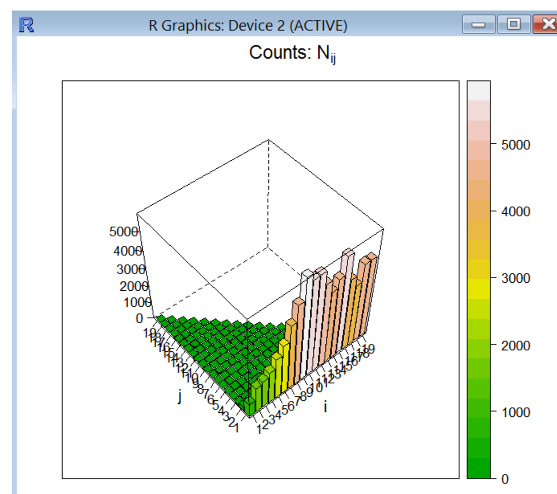


REPORTING

ACCIDENT

Counts data

	1	2	3	4	5	6	7
1							
2							
3							
4							
5							
6							
7							



Cass Business School
CITY UNIVERSITY LONDON

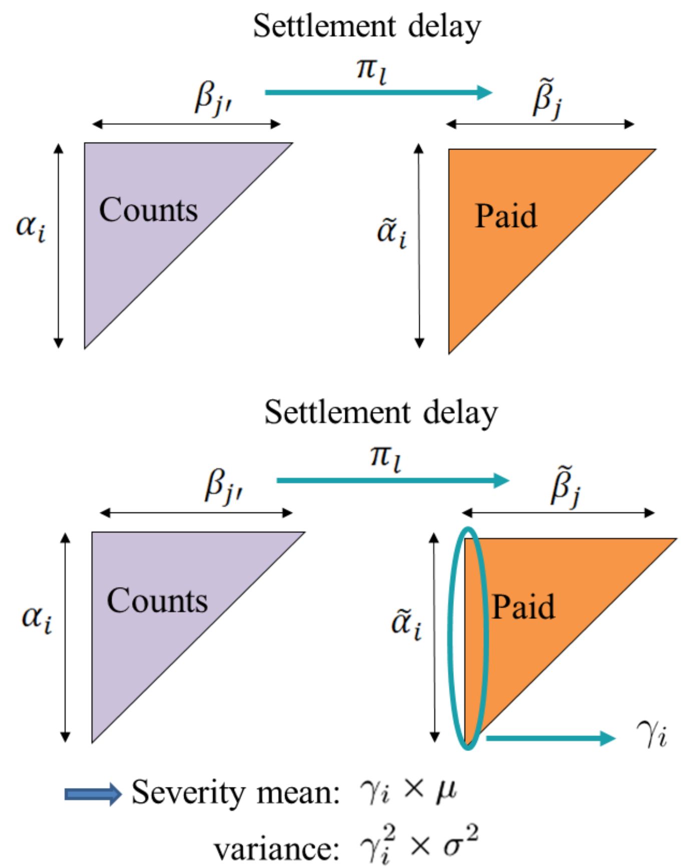
The kernel: calibrating the model

- 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: $\sigma^2\gamma_i^2 \equiv \sigma_i^2$

- **Classical chain ladder technique is applied twice** to give everything needed to estimate.



Cass Business School
CITY UNIVERSITY LONDON

The kernel: parameter estimation using DCL

- The function **`dcl.estimation()`**

dcl.estimation {DCL} — R Documentation

Parameter estimation - Double Chain Ladder model

Description

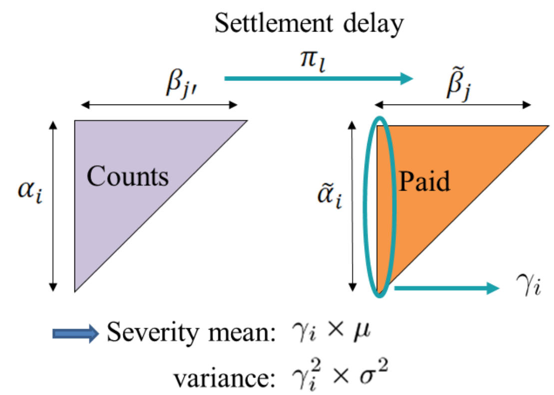
Compute the estimated parameters in the model (delay parameters, severity underwriting inflation, severity mean and variance) using the Double Chain Ladder method.

Usage

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

Arguments

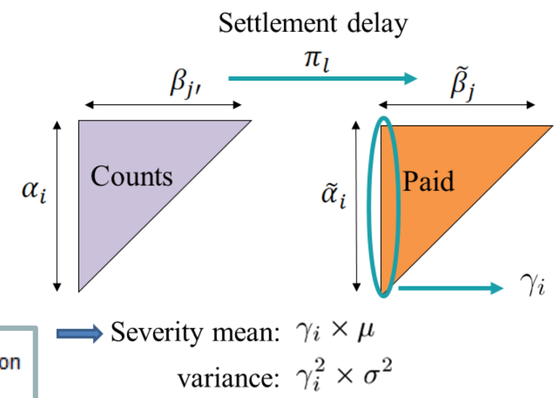
`Xtriangle`	The paid run-off triangle: incremental aggregated payments. It should be a matrix with incremental aggregated payments located in the upper triangle and the lower triangle consisting in missing or zero values.
`Ntriangle`	The counts data triangle: incremental number of reported claims. It 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 has 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`).



Cass Business School
CITY UNIVERSITY LONDON

The kernel: parameter estimation using DCL

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



plot.dcl.par {DCL} R Documentation

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

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

Arguments

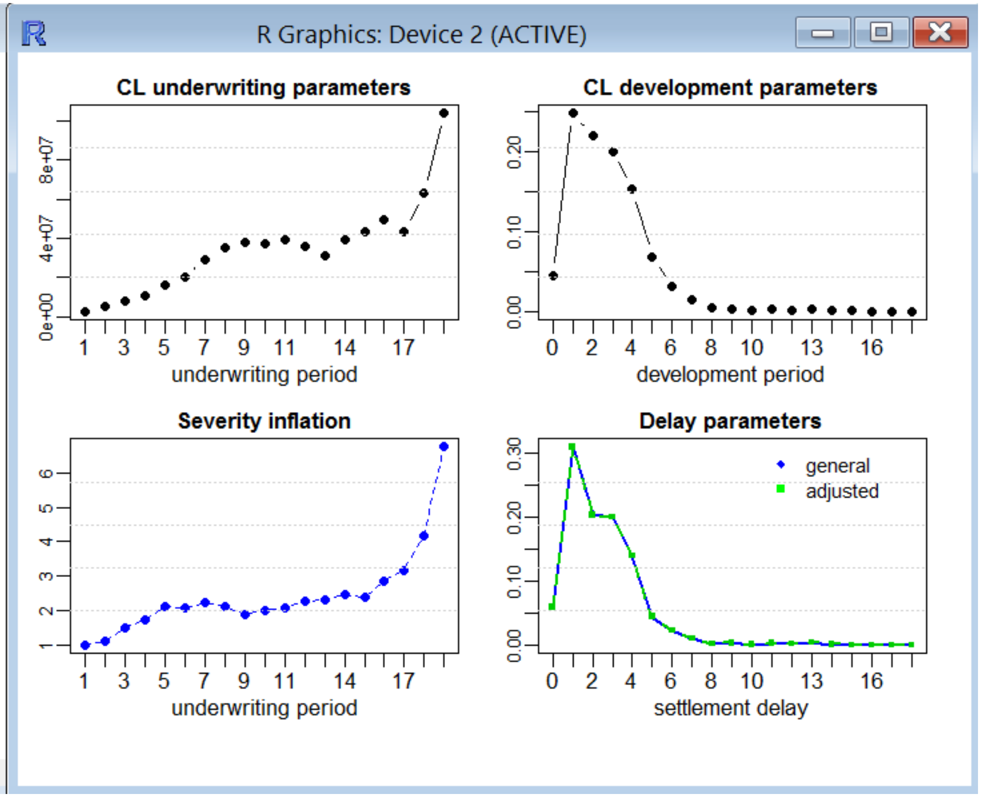
`dcl.par`	A list object with the estimated parameters: the value returned by the functions `dcl.estimation`, `bdcl.estimation` or `idcl.estimation`.
`type.inflat`	Method used to estimate the inflation . Possible values are: 'DCL' (default) if it was used `dcl.estimation`, 'BDCL' if `bdcl.estimation`,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

Cass Business School
CITY UNIVERSITY LONDON

The functions in action: an example

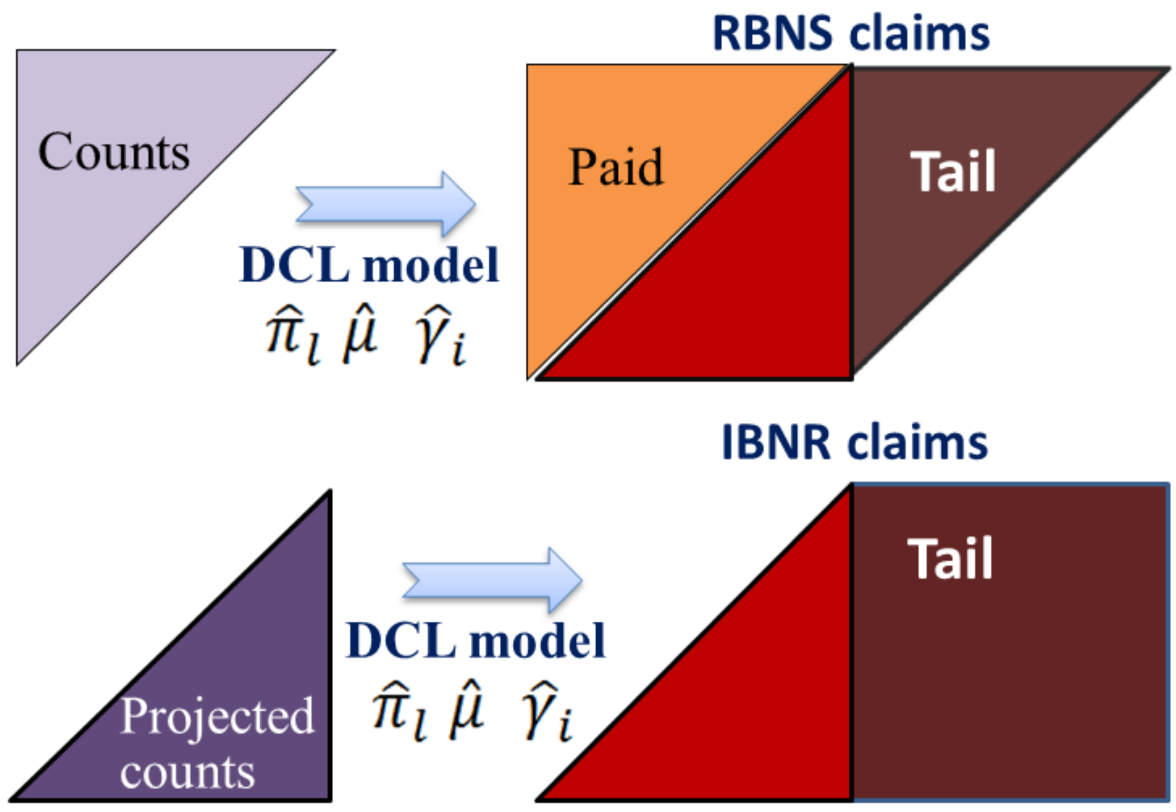
R Console

```
> library(DCL)
> data(NtriangleBDCL)
> data(XtriangleBDCL)
> # The DCL parameters
> my.dcl.par<-dcl.estimation(XtriangleBDCL,NtriangleBDCL)
   delay.par delay.prob inflation severity.mean severity.var
1     0.0592     0.0592    1.0000      2579.064    286808926
2     0.3098     0.3098    1.1173      2881.570    358036053
3     0.2032     0.2032    1.4947      3855.014    640796811
4     0.1996     0.1996    1.7461      4503.280    874432486
5     0.1388     0.1388    2.1075      5435.263   1273824141
6     0.0440     0.0440    2.0936      5399.464   1257099346
7     0.0227     0.0227    2.2495      5801.697   1451371123
8     0.0095     0.0095    2.1250      5480.521   1295126156
9     0.0018     0.0018    1.9028      4907.442   1038433681
10    0.0029     0.0029    2.0197      5208.871   1169918179
11    0.0002     0.0002    2.0704      5339.587   1229373075
12    0.0026     0.0026    2.2666      5845.709   1473474978
13    0.0019     0.0019    2.3157      5972.242   1537953134
14    0.0032     0.0032    2.4747      6382.359   1756429648
15   -0.0002     0.0006    2.3829      6145.592   1628530112
16    0.0013     0.0000    2.8391      7322.296   2311867264
17   -0.0004     0.0000    3.1815      8205.383   2903127034
18    0.0000     0.0000    4.1747     10766.824   4998544792
19    0.0000     0.0000    6.7501     17409.045  13068274219
  mean.factor mean.factor.adj variance.factor
1    2579.002        2579.064       286808926
> plot.dcl.par(my.dcl.par)
> 
```



Parameter estimates in two cases: the basic DCL model (only mean specifications) and the distributional model.

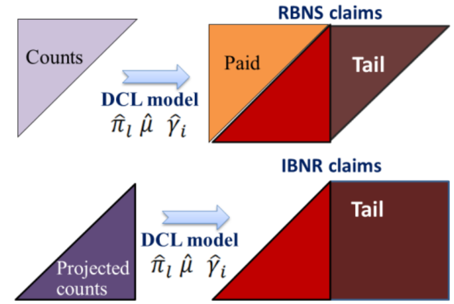
The best estimate: RBNS/IBNR split



Cass Business School
CITY UNIVERSITY LONDON

The best estimate: RBNS/IBNR split using DCL

- The function **`dcl.predict()`**



dcl.predict {DCL} R Documentation

Pointwise predictions (RBNS/IBNR split)

Description

Pointwise predictions by calendar years and rows of the outstanding liabilities. The predictions are splitted 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`, `bdcl.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 in missing or zero values. It should has 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 (`summ.by="diag"`) or by underwriting period (`summ.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`

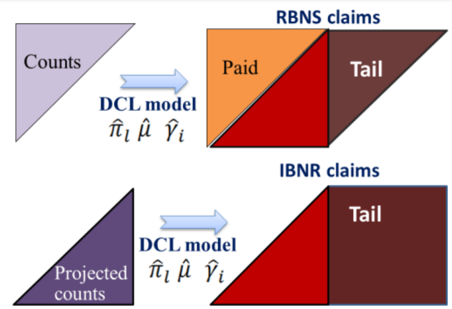
Details

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

R Console

```
> pred.by.diag<-dcl.predict(my.dcl.par,NtriangleBDCL)
   Future.years        rbns       ibnr       total        clm
1             1 59845052.96 1386631.90 61231684.86 61090912.9
2             2 41447058.01 7405875.89 48852933.90 48061354.9
3             3 31016097.53 5610771.34 36626868.87 36266481.8
4             4 17542089.42 5501517.13 23043606.55 22989797.0
5             5  6443018.76 4069044.13 10512062.89 10439464.1
6             6  3192176.74 1719910.74  4912087.48  4913941.1
7             7  1445598.60  944953.87  2390552.47  2380120.6
8             8   675017.48  486952.87  1161970.35  1174086.8
9             9   642274.45  210295.79   852570.24   848055.6
10           10   423522.65  168593.53   592116.19   599855.7
11           11   535548.94   72125.43   607674.37   593718.3
12           12   404459.01   99337.90   503796.92   495823.4
13           13   334964.95   74405.59   409370.54   397094.7
14           14    60022.99   96886.33   156909.31   135553.4
15           15        0.00   37035.26    37035.26   109484.7
16           16        0.00   12228.15    12228.15        0.0
17           17        0.00    6545.30     6545.30        0.0
18           18        0.00    3691.79     3691.79        0.0
19           19        0.00    1831.78     1831.78         NA
20           20        0.00    1013.15     1013.15         NA
21           21        0.00     518.55      518.55         NA
22
23
24
25
26
27
28           28        0.00       6.47        6.47         NA
```



Summary by diagonals (future calendar years), rows (underwriting) and the individual cell predictions

The full cash-flow: Bootstrapping RBNS/IBNR

- 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:

Component	Distribution
Count data	**Poisson**
RBNS delay	**Multinomial**
Severity	**Gamma**

The full cash-flow: Bootstrapping using DCL

- The function **`dcl.boot()`**

dcl.boot {DCL} R Documentation

Bootstrap distribution: the full cashflow

Description

Provide the distribution of the IBNR, RBNS and total (RBNS+IBRN) 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`, `bdcl.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 defaul 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%, 50%, 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

R Console

```
> boot2<-dcl.boot(my.dcl.par,Ntriangle=NtriangleBDCL)
[1] "Please wait, simulating the distribution..."
[1] "Done!"
   period        rbns   mean.rbns     sd.rbns     Q1.rbns     Q5.rbns
1       1 59845052.96 60104639.86 15073076.18 33597995.93 40915229.26
2       2 41447058.01 41679263.34 11015109.71 22154274.56 27730524.16
3       3 31016097.53 31146079.93  9826146.21 14928552.44 18278582.96
4       4 17542089.42 17251007.40  6956163.95  4958809.18  8595612.19
5       5  6443018.76  6403003.15  3843801.59  1016332.33  2071798.34
6       6  3192176.74  3510417.33  2623285.96   177136.11   762954.18
7       7  1445598.60  1597909.78  1873972.33     2517.46    84721.67
8       8   675017.48   852208.53  1174759.22        0.00      412.15
9       9   642274.45   536551.31  1424260.00        0.00        0.00
10     10   423522.65   376713.73   827293.75        0.00        0.00
11     11   535548.94   164627.71   503755.34        0.00        0.00
12     12   404459.01    96801.19   355413.76        0.00        0.00
13     13   334964.95    56962.35   324013.16        0.00        0.00
14     14    60022.99    13651.87   137771.89        0.00        0.00
15     15        0.00       95.42     2144.22        0.00        0.00
16     16        0.00        0.00        0.00        0.00        0.00
17     17        0.00        0.00        0.00        0.00        0.00
18     18        0.00        0.00        0.00        0.00        0.00
19     19        0.00        0.00        0.00        0.00        0.00
20     20        0.00        0.00        0.00        0.00        0.00
21
22
23
24
25
26
27
28
```

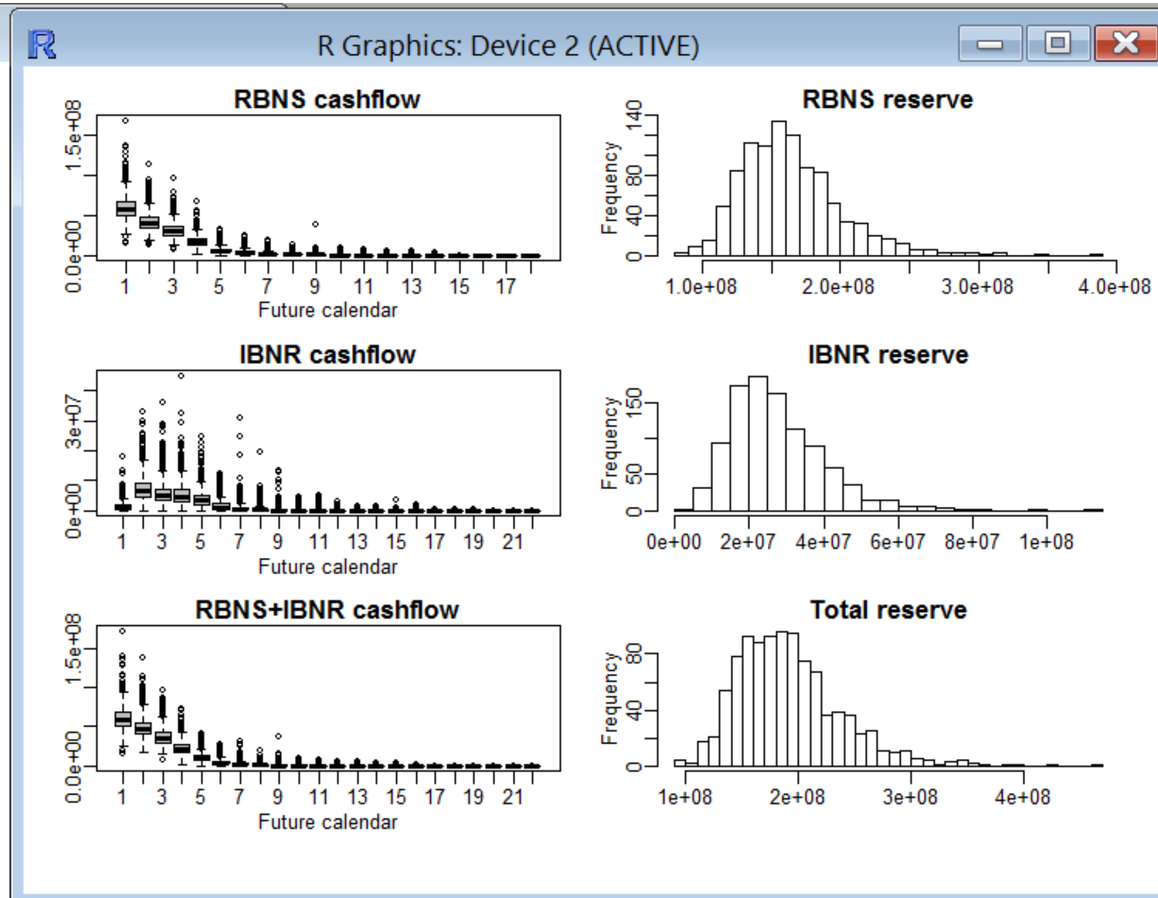
- A table showing a summary of the distribution: mean, std. deviation, quantiles.
- Arrays and matrices with the full simulated distributions

Cass Business School
CITY UNIVERSITY LONDON

The functions in action: an example

R Console

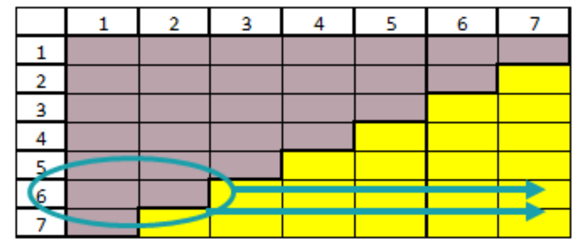
```
> names(boot2)
[1] "array.rbns.boot" "array.ibnr.boot"
[3] "Mat.rbns"        "Mat.ibnr"
[5] "Mat.total"
> plot.cashflow(boot2)
> |
```



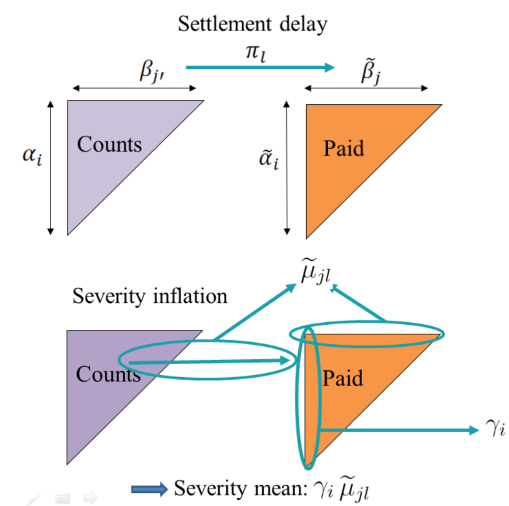
Moving from the (paid) chain ladder mean

Prior knowledge, when it is available, can be incorporated to:

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

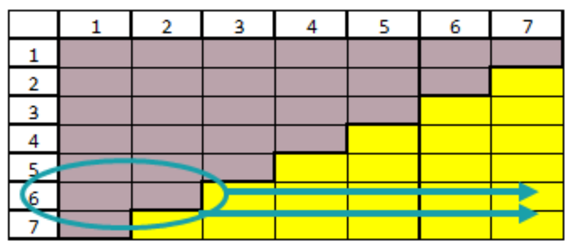


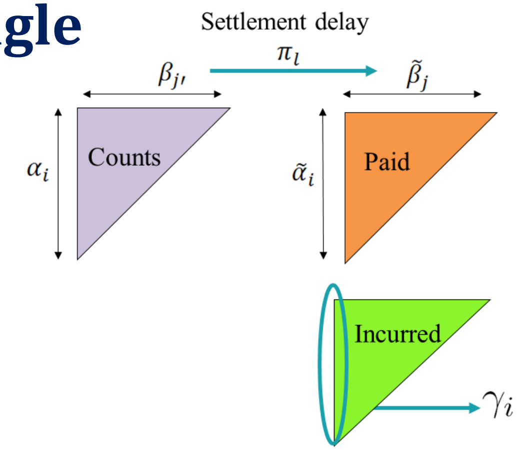
- Consider in practice more **general models**: development severity inflation, zero-claims etc.



Using incurred data through BDCL and IDCL

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





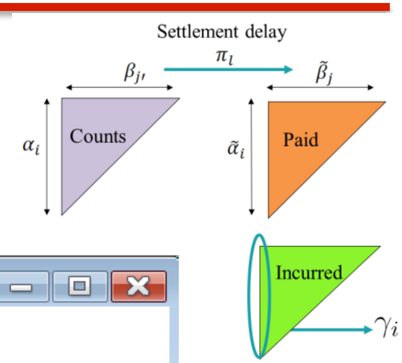
- The IDCL method makes a correction in the underwriting inflation to reproduce the incurred chain ladder reserve

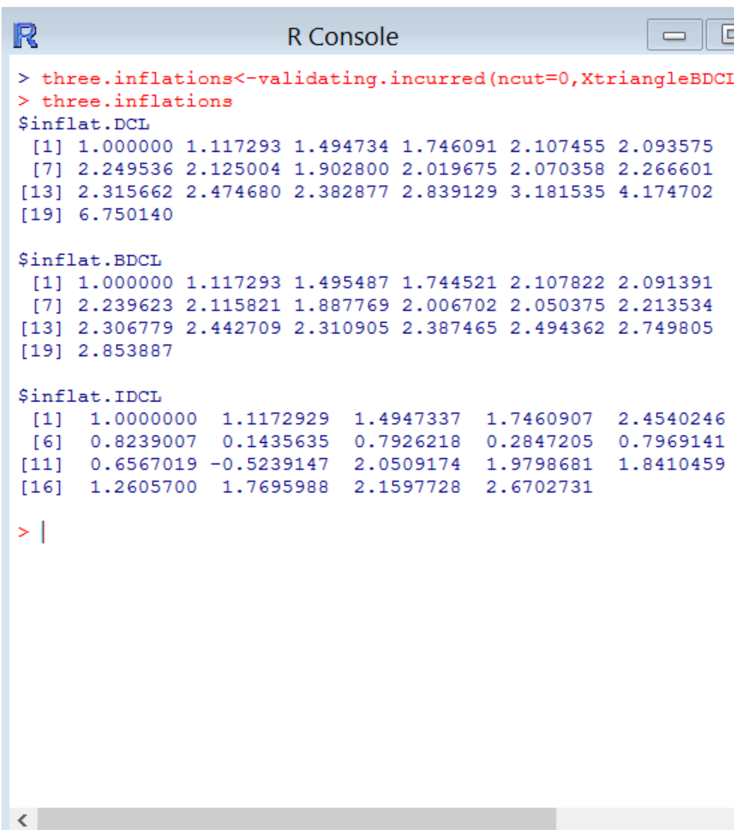
Summary: BDCL and IDCL operate on **3 triangles** and give a different reserve than the paid chain ladder. Both provide the **full cash-flow (RBNS/IBNR)**

Cass Business School
CITY UNIVERSITY LONDON

BDCL and IDCL in the package

- Functions `bdcl.estimation()` `idcl.estimation()`
- Validation strategy: `validating.incurred()`



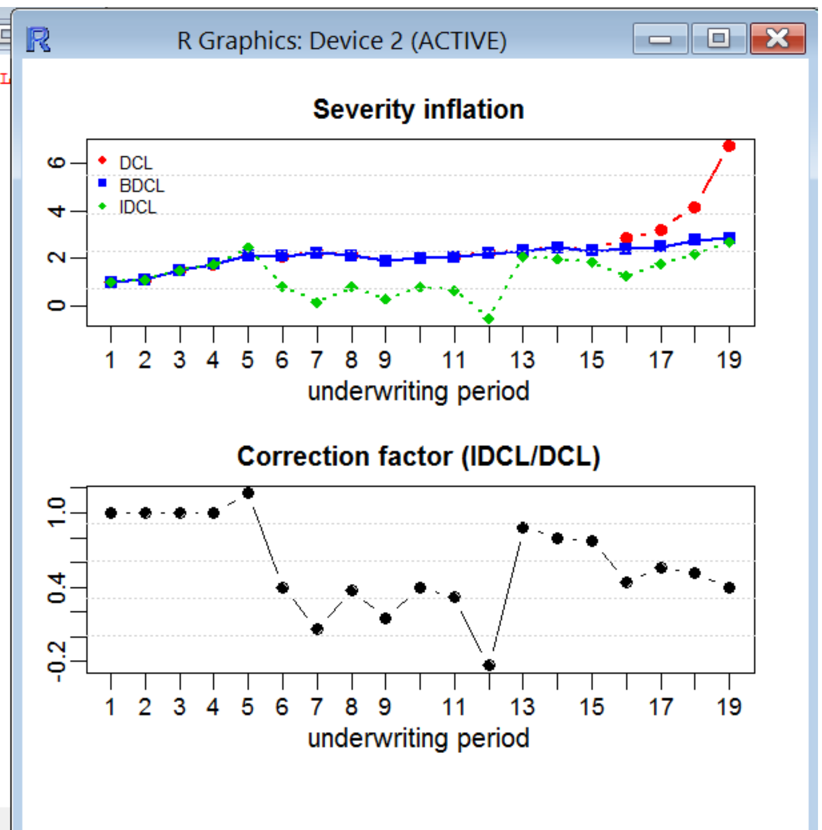


```
> three.inflations<-validating.incurred(ncut=0,XtriangleBDCL
> three.inflations
$inflat.DCL
 [1] 1.000000 1.117293 1.494734 1.746091 2.107455 2.093575
 [7] 2.249536 2.125004 1.902800 2.019675 2.070358 2.266601
[13] 2.315662 2.474680 2.382877 2.839129 3.181535 4.174702
[19] 6.750140

$inflat.BDCL
 [1] 1.000000 1.117293 1.495487 1.744521 2.107822 2.091391
 [7] 2.239623 2.115821 1.887769 2.006702 2.050375 2.213534
[13] 2.306779 2.442709 2.310905 2.387465 2.494362 2.749805
[19] 2.853887

$inflat.IDCL
 [1]  1.0000000  1.1172929  1.4947337  1.7460907  2.4540246
 [6]  0.8239007  0.1435635  0.7926218  0.2847205  0.7969141
[11]  0.6567019 -0.5239147  2.0509174  1.9798681  1.8410459
[16]  1.2605700  1.7695988  2.1597728  2.6702731

> |
```

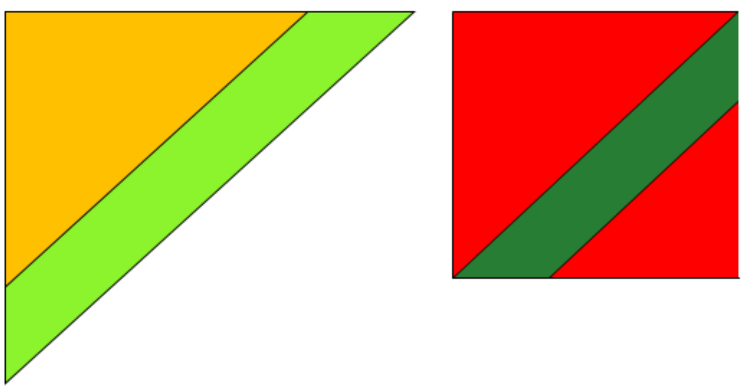


Cass Business School
CITY UNIVERSITY LONDON

Validation

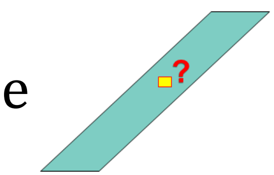
Testing results against experience:

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

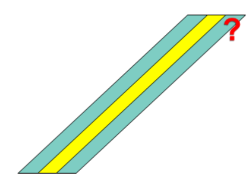


Three objectives:

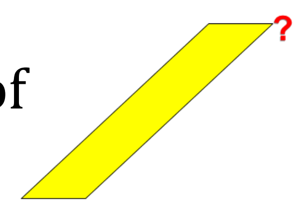
- Predictions of the individual cells



- Predictions by calendar years



- The prediction of the overall total



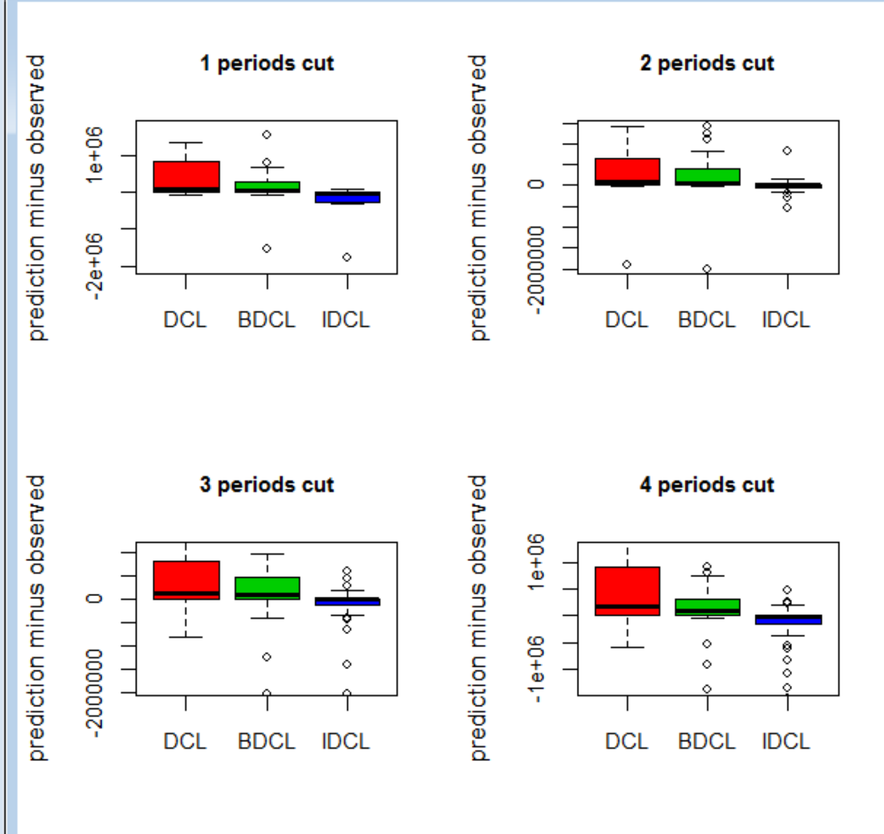
Cass Business School
CITY UNIVERSITY LONDON

Validation strategy: `validating.incurred()`

R Console

```
> par(mfrow=c(2,2),cex.axis=0.9,cex.main=1)
> for (i in 1:4)
+ {
+   res<-validating.incurred(ncut=i,XtriangleBDCL,Ntr$
+   test.res[i,]<-as.numeric(res$pe.vector)
+ }
> test.res<-as.data.frame(test.res)
> names(test.res)<-c("num.cut","pe.point.DCL","pe.poi$
+ "pe.calendar.DCL","pe.calendar.BDCL","pe.calendar.I$
+ "pe.total.DCL","pe.total.BDCL","pe.total.IDCL")
> print(test.res)
  num.cut pe.point.DCL pe.point.BDCL pe.point.IDCL
1       1    0.2798278     0.2801055     0.3135739
2       2    0.3164563     0.2814852     0.2764642
3       3    0.2946255     0.3471936     0.3629964
4       4    0.3265120     0.3012860     0.3480651
  pe.calendar.DCL pe.calendar.BDCL pe.calendar.IDCL
1       0.3193591       0.05319649        0.2642224
2       0.2762849       0.08022043        0.1991053
3       0.2617929       0.14101643        0.3015099
4       0.3574390       0.14166052        0.3516227
  pe.total.DCL pe.total.BDCL pe.total.IDCL
1    0.3193591    0.05319649     0.2642224
2    0.2764211    0.02251542     0.1651506
3    0.2602013    0.04770648     0.2399416
4    0.3946862    0.01887476     0.2710949
>
```

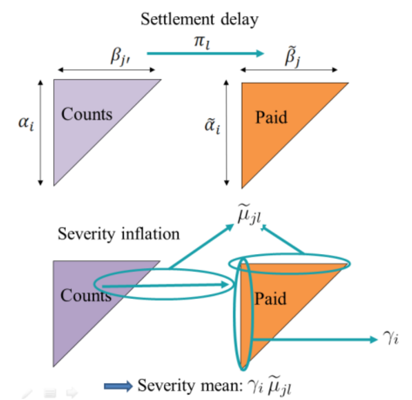
R Graphics: Device 2 (ACTIVE)



Cass Business School
CITY UNIVERSITY LONDON

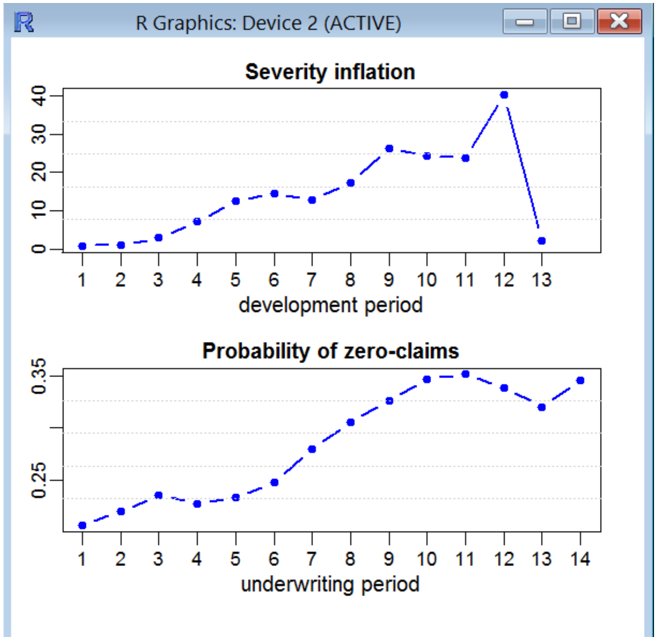
Working in practice with a more general model

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



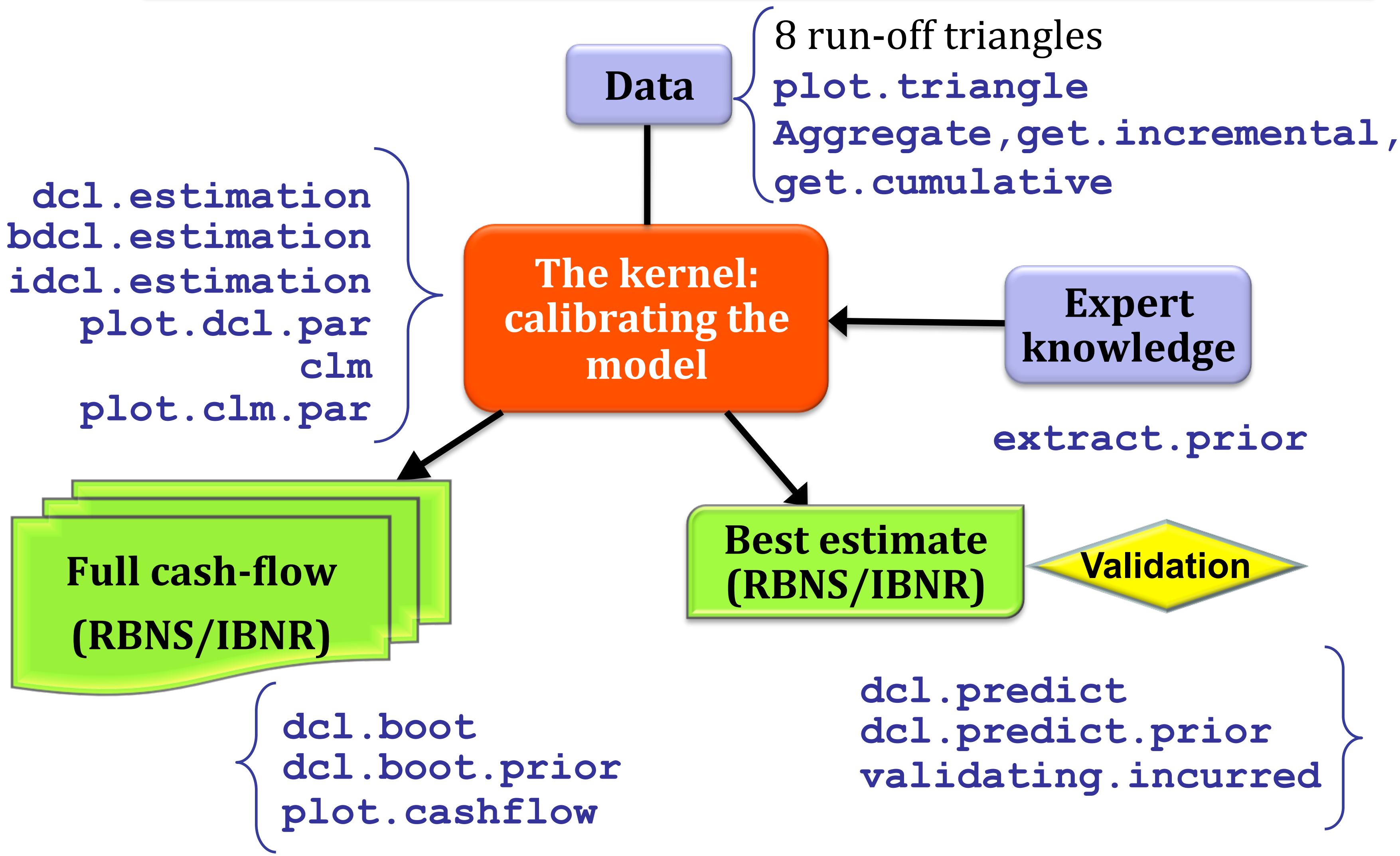
- The package provides the functions:

```
dcl.predict.prior()
dcl.boot.prior()
extract.prior()
```



Cass Business School
CITY UNIVERSITY LONDON

Summary: the content of the package



Cass Business School
CITY UNIVERSITY LONDON

What next?

- We look for a **wide audience** (academics, practitioners, students).
- Your feedback is very valuable…
- Submitting to the CRAN…
- Variations and extensions are expected to come soon from the knowledge loop…
- It is just the first version…

Cass Business School
CITY UNIVERSITY LONDON

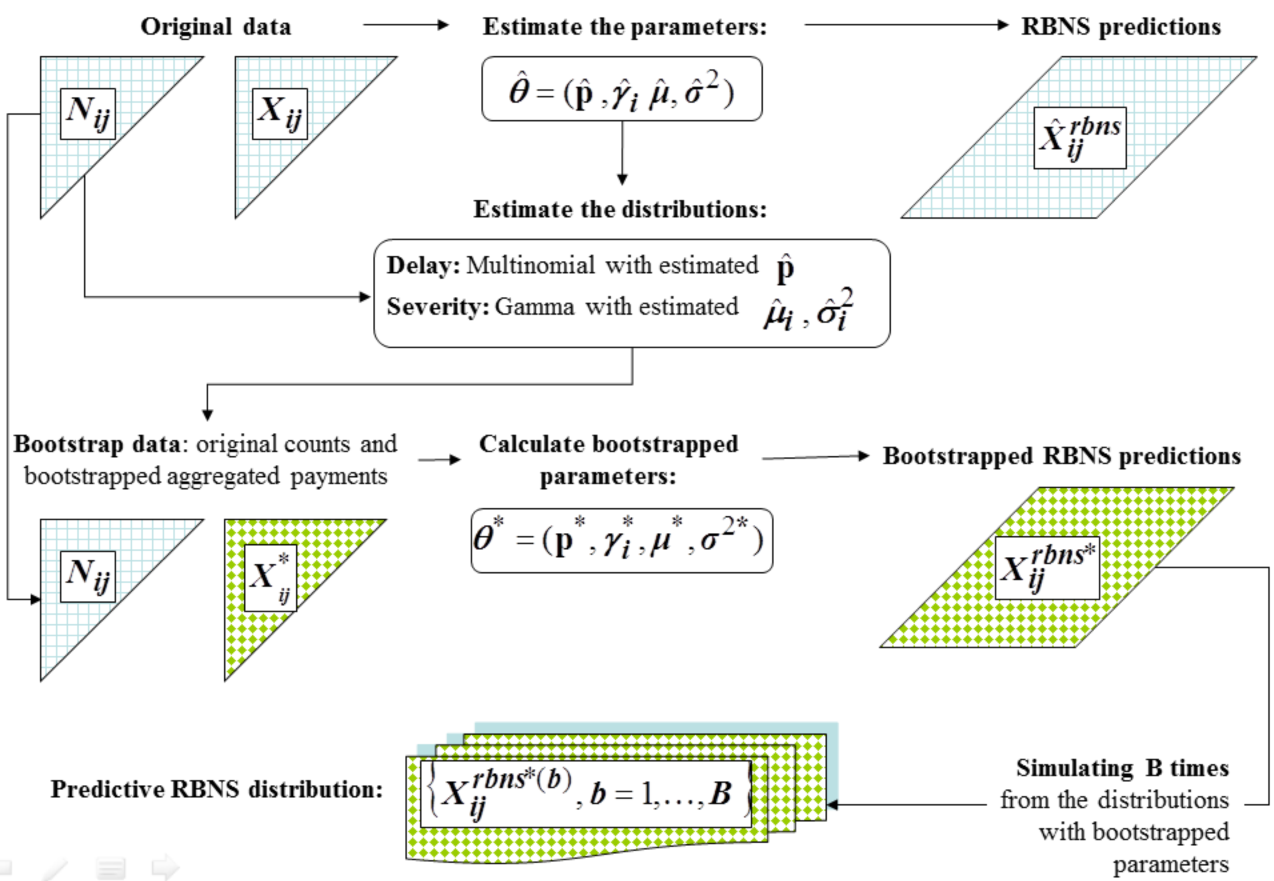
Appendix A: code -examples in this presentation

```
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)
```

Cass Business School
CITY UNIVERSITY LONDON

Appendix B: Bootstrap methods

Algorithm RBNS – Bootstrapping taking into account parameters uncertainty



Appendix B: Bootstrap methods

Algorithm IBNR – Bootstrapping taking into account parameters uncertainty