Loss modelling and reserving analytics in R

A hands-on workshop

Katrien Antonio & Jonas Crevecoeur

IA|BE workshop | June 3, 10 & 17, 2021

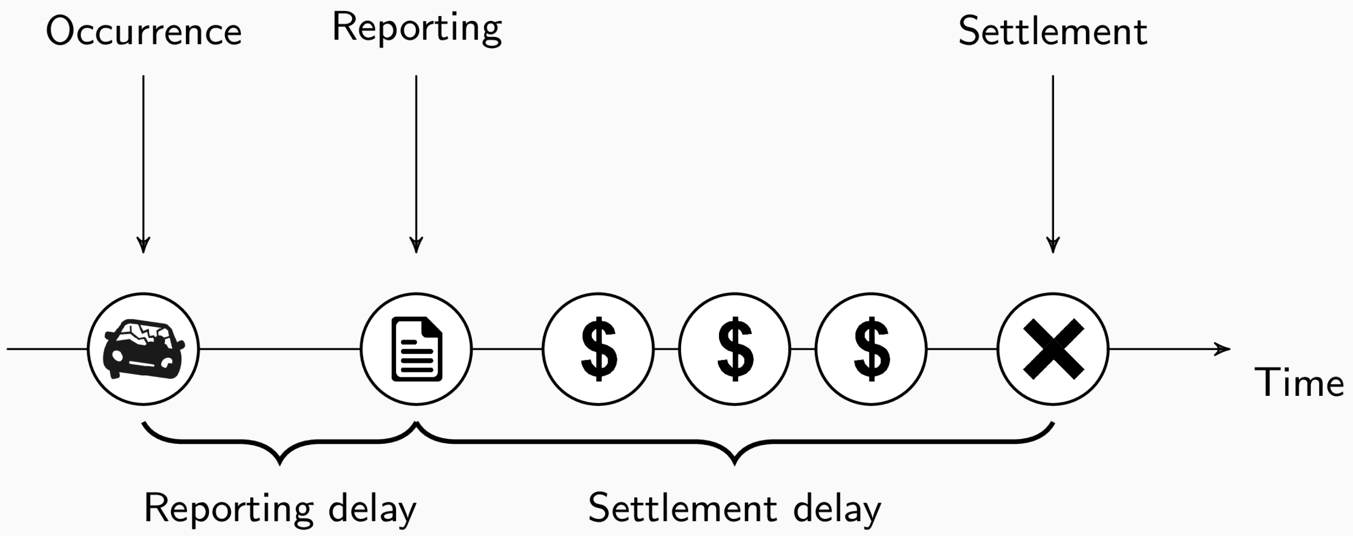
Today's Outline

- Motivation and strategies
- Reserving data structures (part 1)
  - IBNR and RBNS reserve
  - daily and yearly reserving data
- Reserving data structures (part 2)
  - individual and aggregated reserving data
  - runoff triangles
- Claims reserving with triangle
  - chainladder model
  - {ChainLadder} package
  - chainLadder GLM implementation
- When the chain ladder method fails
  - detection tools for triangle stability
  - creating homogeneous triangles
- Research outlook
  - granular reserving methods for predicting the IBNR reserve
  - individual reserving methods for predicting the RBNS reserve

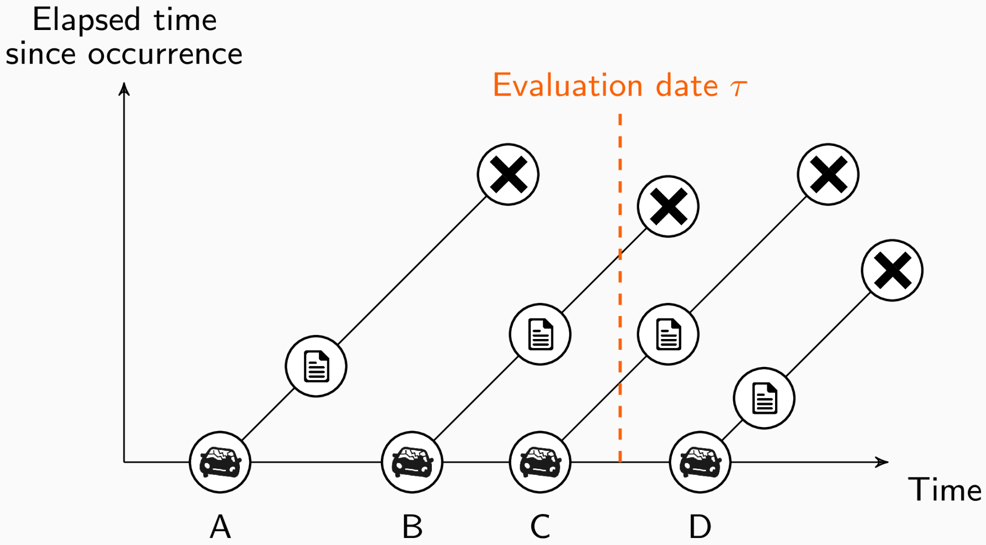
Motivation and strategies

Motivation

Insurers observe the **detailed evolution** of **individual claims**.



Motivation (continued)



Observed claims are **censored** due to **delays** (reporting, settlement) in the claim development process

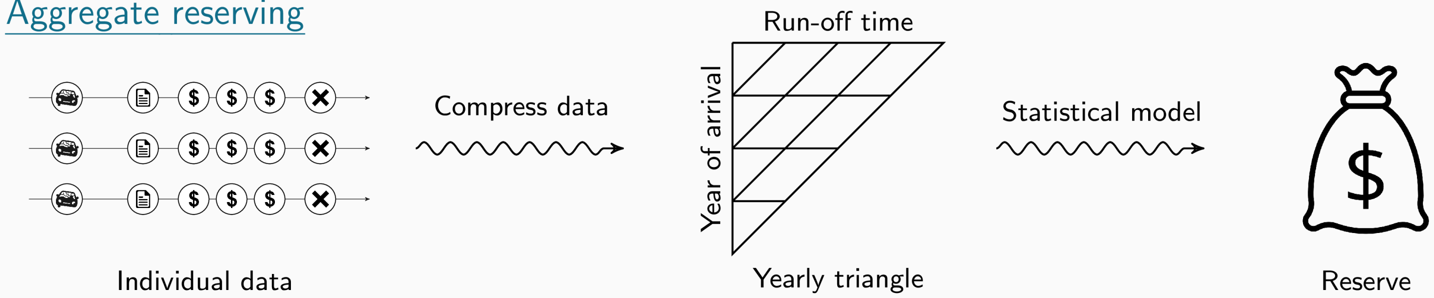
Reserve: future costs for claims that occurred in **past exposure periods** (claim B and C)

- RBNS reserve: Reserve for Reported, But Not yet Settled claims (claim B)
- IBNR reserve: Reserve for Incurred, But Not yet Reported claims (claim C)

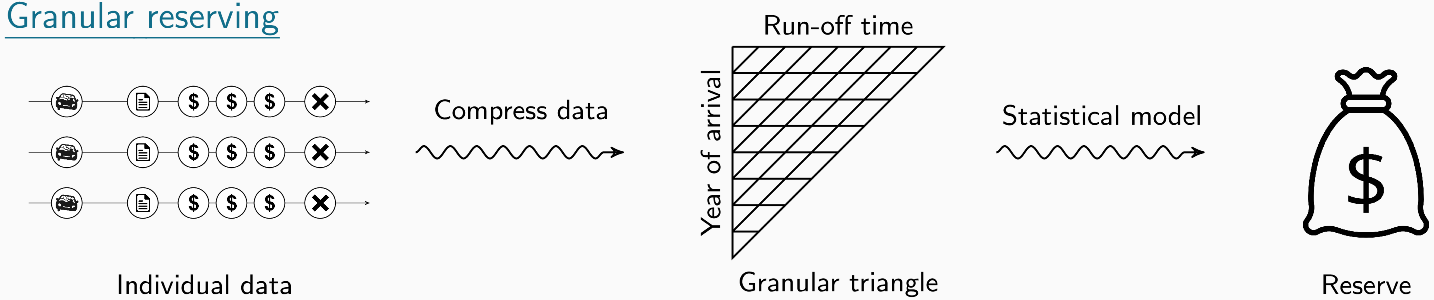
Pricing: all costs for claims that will occur in **future** insured **exposure periods** (claim D)

Three strategies for non-life reserving

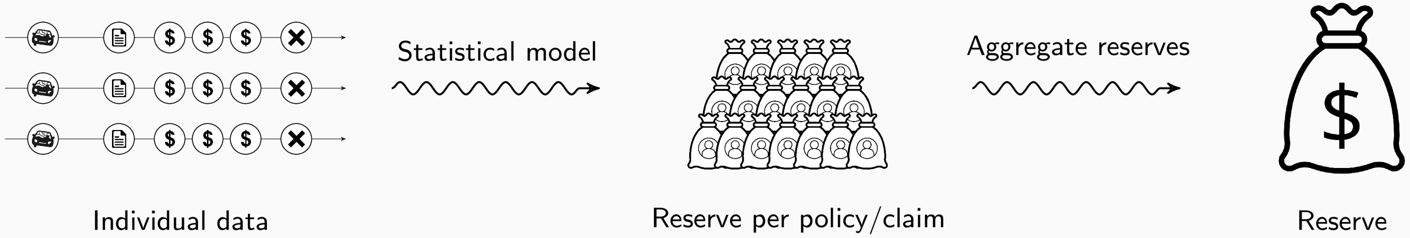
Aggregate reserving



Granular reserving



Individual reserving



Three strategies for non-life reserving (continued)

Aggregate reserving

First aggregate the past claim history into a **small number of summary statistics**,
then predict the total reserve based on these summary statistics.

Granular reserving

First aggregate the past claim history into a **large number of summary statistics**,
then predict the total reserve based on these summary statistics.

Individual reserving

First predict the future costs for **indivudal claims**,
then aggregate these individual predictions to predict the total reserve.

Data set used in this session

We illustrate reserving analytics based on a **simulated dataset** registering the **detailed development** of 30,000 claims between January 1, 2010 and December 31, 2019.

- Information is available at daily level
- Records correspond to payments made for these claims

The data is stored in a .RData file in the `data` directory in the course material:

```
# install.packages("rstudioapi")
dir <- dirname(rstudioapi::getActiveDocumentContext()$path)
setwd(dir)

load("data/reserving_data.RData")
```

```
##   accident_number accident_date reporting_date settlement_date reporting_delay settlement_delay payment_date payment_size
## 1               1    2012-08-27     2012-09-01      2012-12-03               5               98   2012-09-01       0.0000
## 2               1    2012-08-27     2012-09-01      2012-12-03               5               98   2012-09-07     446.2251
## 3               1    2012-08-27     2012-09-01      2012-12-03               5               98   2012-09-18     490.1792
## 4               1    2012-08-27     2012-09-01      2012-12-03               5               98   2012-10-01     587.7409
## 5               1    2012-08-27     2012-09-01      2012-12-03               5               98   2012-10-19    1726.0902
## 6               2    2013-09-20     2013-09-26      2014-04-06               6              198   2013-09-26       0.0000
```

Data set used in this session (continued)

Discretize daily dates to yearly indices:

```
date_to_year <- function(date, base_year) {
  year <- as.numeric(format(date, '%Y')) - base_year
}

reserving_data <- reserving_data %>%
  mutate(accident_year = date_to_year(accident_date, 2010),
         reporting_year = date_to_year(reporting_date, 2010),
         payment_year = date_to_year(payment_date, 2010),
         development_year = payment_year - accident_year + 1,
         settlement_year = date_to_year(settlement_date, 2010))
```

```
##   accident_number accident_date reporting_date settlement_date reporting_delay settlement_delay payment_date payment_size accident_year reporting_year payment_year
## 1               1    2012-08-27     2012-09-01      2012-12-03               5               98   2012-09-01       0.0000             2              2            2
## 2               1    2012-08-27     2012-09-01      2012-12-03               5               98   2012-09-07     446.2251             2              2            2
## 3               1    2012-08-27     2012-09-01      2012-12-03               5               98   2012-09-18     490.1792             2              2            2
## 4               1    2012-08-27     2012-09-01      2012-12-03               5               98   2012-10-01     587.7409             2              2            2
## 5               1    2012-08-27     2012-09-01      2012-12-03               5               98   2012-10-19    1726.0902             2              2            2
## 6               2    2013-09-20     2013-09-26      2014-04-06               6              198   2013-09-26       0.0000             3              3            3
##   development_year settlement_year
## 1                1               2
## 2                1               2
## 3                1               2
## 4                1               2
## 5                1               2
## 6                1               4
```

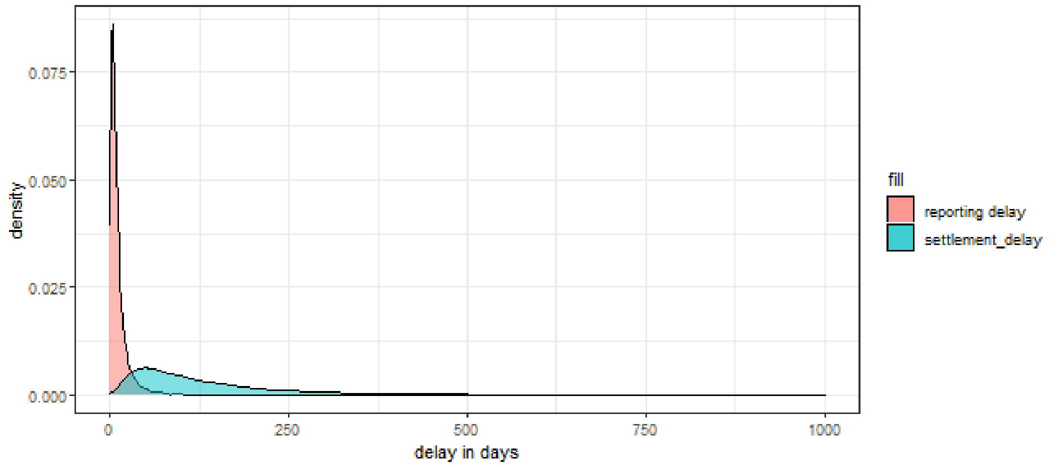
Your turn

In this warm up exercise you load the `reserving data` data set and get some feel for the data.

1. Visualize the reporting and settlement delay of claims with a density plot.
2. When was the last payment registered in the data set?
   What do you conclude?
3. What is the average number of payments per claim?
4. Calculate the number of claims per accident year.

For **Q.1** we plot the density of reporting and settlement delay:

```
ggplot(data = reserving_data) +
  theme_bw() +
  geom_density(aes(reporting_delay, fill = 'reporting delay'),
               alpha = .5) +
  geom_density(aes(settlement_delay, fill = 'settlement_delay'),
               alpha = .5) +
  xlab('delay in days') +
  xlim(c(0, 1000))
```



For **Q.2** the last payment was in 2024, i.e. the simulated data is not yet censored!

```
max(reserving_data$payment_date)
```

```
## [1] "2024-05-05"
```

For **Q.3** the data set records one row per payment.

```
num_claim <- length(unique(reserving_data$accident_number))
num_payment <- nrow(reserving_data)

num_payment/num_claim
```

```
## [1] 3.425433
```

At reporting, the data set registers a zero payment for each claim. The average number of non-zero payments per claim is

```
num_payment <- sum(reserving_data$payment_size > 0)
num_payment/num_claim
```

```
## [1] 2.376633
```

For **Q.4** we group the data by accident year and summarise the number of claims.

```
library(tidyverse)
reserving_data %>%
  group_by(accident_year) %>%
  summarise(num_claims = n())
```

```
## # A tibble: 10 x 2
##    accident_year num_claims
##  *         <dbl>      <int>
##  1             0      10275
##  2             1      10108
##  3             2      10633
##  4             3      10295
##  5             4      10213
##  6             5      10130
##  7             6      10222
##  8             7      10069
##  9             8      10261
## 10             9      10557
```

The portfolio is stable over time.

Reserving data structures: daily and yearly data

From daily to yearly data

Reserves are often calculated based on yearly, **aggregated data**.
Data aggregation removes the daily noise from the data.

In a first step, we construct a data set that consists of **one record per claim and per development year**.
We construct this data set by merging three partial data sets:

- `records`: **empty data set** with one record per claim per development year
- `claim_data`: data set with one record per claim containing the **static claim covariates** (e.g. accident year)
- `payment_data`: data set containing the total **amount paid per claim and per development year**.
  This data set contains only records for development years with at least one payment.

From daily to yearly data (continued)

For `records`, `expand.grid` creates a data.frame with all combinations of `accident_number` and `development year`.

```
max_dev_year <- 10
accidents <- unique(reserving_data$accident_number)

records <- expand.grid(accident_number = accidents,
                       development_year = 1:max_dev_year)
```

`expand.grid` creates a data.frame with all possible combinations of the input vectors.

```
expand.grid(A = c(1, 5, 7), B = c(4, 6))
```

```
##   A B
## 1 1 4
## 2 5 4
## 3 7 4
## 4 1 6
## 5 5 6
## 6 7 6
```

```
head(records)
```

```
##   accident_number development_year
## 1               1                1
## 2               2                1
## 3               3                1
## 4               4                1
## 5               5                1
## 6               6                1
```

From daily to yearly data (continued)

For `claim_data`, we select for each `accident_number` the first row in `reserving_data`.

```
claim_data <- reserving_data %>%
  group_by(accident_number) %>%
  slice(1) %>%
  ungroup() %>%
  select(accident_number, accident_year,
         accident_date, reporting_year,
         settlement_year)
```

```
head(claim_data)
```

```
## # A tibble: 6 x 5
##   accident_number accident_year accident_date reporting_year
##             <int>         <dbl> <date>                 <dbl>
## 1               1             2 2012-08-27                 2
## 2               2             3 2013-09-20                 3
## 3               3             5 2015-09-24                 5
## 4               4             9 2019-01-30                 9
## 5               5             2 2012-01-07                 2
## 6               6             8 2018-12-25                 8
```

`slice(k)` selects the k-th record of a data set

```
df <- data.frame(x = c(1, 2, 3, 4, 5, 6), y = c(1, 1, 1, 2, 2,
df %>% slice(4)
```

```
##   x y
## 1 4 2
```

In combination with `group_by` we can select the k-th record within each group

```
df %>% group_by(y) %>% slice(1)
```

```
## # A tibble: 2 x 2
## # Groups:   y [2]
##       x     y
##   <dbl> <dbl>
## 1     1     1
## 2     4     2
```

From daily to yearly data (continued)

For `payment_data`, we aggregate payments per claim by `development_year`.

```
payment_data <- reserving_data %>%
  group_by(accident_number, development_year) %>%
  summarise(size = sum(payment_size),
            payment = size > 0)
```

```
head(payment_data)
```

```
## # A tibble: 6 x 4
## # Groups:   accident_number [5]
##   accident_number development_year  size payment
##             <int>            <dbl> <dbl> <lgl>
## 1               1                1 3250. TRUE
## 2               2                1 2347. TRUE
## 3               2                2 1189. TRUE
## 4               3                1 1975. TRUE
## 5               4                1 3107. TRUE
## 6               5                1 2376. TRUE
```

`summarise` returns one record for each combination of `accident_number` and `development_year` in `reserving_data`.

The data set contains no records development years in which no payment was made for a claim!

From daily to yearly data (continued)

Merge `records`, `claim_data` and `payment_data` into a new data set `individual_data`.

```
individual_data <- records %>%
  left_join(claim_data,
            by = 'accident_number') %>%
  left_join(payment_data,
            by = c('accident_number', 'development_year')) %>%
  mutate(size = replace_na(size, 0),
         payment = replace_na(payment, FALSE))
```

```
dim(individual_data)
```

```
## [1] 300000      8
```

`left_join` combines records in the first (left) with records from the second (right) data set with matching values in the `by` column.

If no match is found in the right data set an `NA` value is imputed.

```
left <- data.frame(x = c(1, 2, 3), left = 1)
right <- data.frame(x = c(1, 2, 4), right = 1)
left_join(left, right, by = 'x')
```

```
##   x left right
## 1 1    1     1
## 2 2    1     1
## 3 3    1    NA
```

```
head(individual_data, 3)
```

```
##   accident_number development_year accident_year accident_date reporting_year settlement_year     size payment
## 1               1                1             2    2012-08-27              2               2 3250.235    TRUE
## 2               2                1             3    2013-09-20              3               4 2347.445    TRUE
## 3               3                1             5    2015-09-24              5               5 1975.269    TRUE
```

Censoring the data

The simulated data is not censored, i.e. we observe the full development of all claims that occur before 2019.

In practice the available data set is censored and we only observe:

- reported claims
- development information for past calendar years

We censor the data assuming that we observe data until the end of 2019

```
observed_data <- individual_data %>%
  mutate(calendar_year = accident_year + development_year - 1) %>%
  filter(calendar_year >= reporting_year,
         calendar_year <= 9)

unobserved_data <- individual_data %>%
  mutate(calendar_year = accident_year + development_year - 1) %>%
  filter(calendar_year > 9)
```

IBNR and RBNS reserves

Reserving models predict the total payments in the unobserved data

```
reserve_actual <- sum(unobserved_data$size)
reserve_actual
```

```
## [1] 2468246
```

Some reserving methods split this reserve in an

- IBNR reserve: a reserve for Incurred, But Not (yet) Reported claims
- RBNS reserve: a reserve for Reported, But Not (yet) Settled claims

```
unobserved_data %>%
  mutate(reported = (reporting_year <= 9)) %>%
  group_by(reported) %>%
  summarise(reserve = sum(size))
```

```
## # A tibble: 2 x 2
##   reported  reserve
## * <lgl>       <dbl>
## 1 FALSE     158665.
## 2 TRUE     2309581.
```

Reserving data structures: individual and aggregated data

From individual to aggregated data

Individual triangle

The data set `individual_data` describes the development of **individual claims discretised by calendar year**.

We can represent the claim characteristics in this data set (settlement, payment, payment size) with **two-dimensional tables** in which the rows represent **individual claims** and columns represent **development years**.

For payment size, this table consists of cells $U_{k,j}^{\texttt{size}}$ with $U_{k,j}^{\texttt{size}}$ denoting the total amount paid for claim k in development year j.

Runoff triangle

Many reserving models go one step further and remove individual claim characteristics by aggregating claims by occurrence year. The data is then represented in a (small) two-dimensional table, the so-called **incremental runoff triangle**.

$$X_{i,j}^{\texttt{size}} = \sum_{\texttt{occ.year(k)}=i} U_{k,j}^{\texttt{size}}.$$

As a result of the Law of Large Numbers (LLN), aggregating data from many claims into runoff triangles **reduces the variance** when all claims are independent realizations from the same distribution.

From individual to aggregated data (continued)

We construct a runoff triangle by aggregating individual claims by `accident_year`.

```
observed_data %>%
  group_by(accident_year, development_year) %>%
  summarise(value = sum(size)) %>%
  pivot_wider(values_from = value,
              names_from = development_year,
              names_prefix = 'DY.')
```

```
## # A tibble: 10 x 11
## # Groups:   accident_year [10]
##    accident_year     DY.1     DY.2    DY.3   DY.4   DY.5  DY.6  DY.7  DY.8  DY.9 DY.1
##            <dbl>    <dbl>    <dbl>   <dbl>  <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl
##  1             0 4532915. 1739699. 121876. 25952.  9305. 6703. 9479. 6217. 3468.
##  2             1 4414954. 1673577. 107415. 19472.     0     0     0     0     0     N
##  3             2 4786833. 1785616. 149240. 59348. 14858. 4404.     0     0    NA     N
##  4             3 4653451. 1614053. 127900. 36214.  9301.     0     0    NA    NA     N
##  5             4 4321019. 1909085. 159247. 49184. 11728.     0    NA    NA    NA     N
##  6             5 4417129. 1722151. 103907. 65592. 15129.    NA    NA    NA    NA     N
##  7             6 4538329. 1718509. 146138. 19248.     NA    NA    NA    NA    NA     N
##  8             7 4578285. 1618201.  77054.     NA     NA    NA    NA    NA    NA     N
##  9             8 4503558. 1756243.      NA     NA     NA    NA    NA    NA    NA     N
## 10             9 4575529.       NA      NA     NA     NA    NA    NA    NA    NA     N
```

`pivot_wider` widens the data set by increasing the number of columns, while decreasing the number of rows.

`names_from`: The resulting data set contains a separate column for each unique outcome of this variable.

`values_from`: Values from this variable are stored in the newly created columns.

```
df <- data.frame(x = c(1, 1, 1, 2, 2, 2, 3, 3, 3),
                 y = c(1, 2, 3, 1, 2, 3, 1, 2, 3),
                 value = c(1, 2, 3, 4, 5, 6, 7, 8, 9))
pivot_wider(df,
            names_from = y,
            values_from = value)
```

```
## # A tibble: 3 x 4
##       x   `1`   `2`   `3`
##   <dbl> <dbl> <dbl> <dbl>
## 1     1     1     2     3
## 2     2     4     5     6
## 3     3     7     8     9
```

From individual to aggregated data (continued)

A more sophisticated function for computing incremental runoff triangles is available in the R code.

```r
incremental_triangle(observed_data,
                     variable = 'payment',
                     lower_na = TRUE)
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,] 2264  550   20    5    2    1    1    1    1     0
##  [2,] 2269  533   27    5    0    0    0    0    0    NA
##  [3,] 2307  557   26    8    3    1    0    0   NA    NA
##  [4,] 2289  537   23    8    1    0    0   NA   NA    NA
##  [5,] 2239  581   33    8    1    0   NA   NA   NA    NA
##  [6,] 2211  532   24    6    3   NA   NA   NA   NA    NA
##  [7,] 2197  537   30    5   NA   NA   NA   NA   NA    NA
##  [8,] 2250  510   18   NA   NA   NA   NA   NA   NA    NA
##  [9,] 2271  544   NA   NA   NA   NA   NA   NA   NA    NA
## [10,] 2319   NA   NA   NA   NA   NA   NA   NA   NA    NA
```

Using this function you can easily inspect the reserving data using multiple triangles.

```r
incremental_triangle(observed_data,
                     variable = 'size',
                     lower_na = TRUE)
```

```
##          [,1]    [,2]      [,3]     [,4]      [,5]     [,6]    [,7]     [,8]     [,9]
##  [1,] 4532915 1739699 121876.31 25951.66  9304.915 6703.063 9478.77 6217.266 3468.364
##  [2,] 4414954 1673577 107415.07 19472.27     0.000    0.000    0.00    0.000    0.000
##  [3,] 4786833 1785616 149239.81 59348.18 14857.743 4404.145    0.00    0.000       NA
##  [4,] 4653451 1614053 127900.21 36213.57  9301.109    0.000    0.00       NA       NA
##  [5,] 4321019 1909085 159246.74 49184.25 11727.691    0.000      NA       NA       NA
##  [6,] 4417129 1722151 103906.89 65592.12 15129.218       NA      NA       NA       NA
##  [7,] 4538329 1718509 146138.00 19248.11        NA       NA      NA       NA       NA
##  [8,] 4578285 1618201  77053.93       NA        NA       NA      NA       NA       NA
##  [9,] 4503558 1756243        NA       NA        NA       NA      NA       NA       NA
## [10,] 4575529      NA        NA       NA        NA       NA      NA       NA       NA
```

Cumulative runoff triangles

Cumulative runoff triangles are constructed by computing the cumulative row sum of an **incremental runoff triangle**, i.e.

$$C_{i,j}^{\texttt{size}} = \sum_{l=1}^{j} X_{i,l}^{\texttt{size}}.$$

When using a new reserving method, check whether the incremental or cumulative runoff triangle is required!

```r
incremental_triangle(observed_data,
                     variable = 'size',
                     lower_na = TRUE)
```

```
##          [,1]    [,2]      [,3]     [,4]      [,5]     [,6]    [,7]     [,8]     [,9]
##  [1,] 4532915 1739699 121876.31 25951.66  9304.915 6703.063 9478.77 6217.266 3468.364
##  [2,] 4414954 1673577 107415.07 19472.27     0.000    0.000    0.00    0.000    0.000
##  [3,] 4786833 1785616 149239.81 59348.18 14857.743 4404.145    0.00    0.000       NA
##  [4,] 4653451 1614053 127900.21 36213.57  9301.109    0.000    0.00       NA       NA
##  [5,] 4321019 1909085 159246.74 49184.25 11727.691    0.000      NA       NA       NA
##  [6,] 4417129 1722151 103906.89 65592.12 15129.218       NA      NA       NA       NA
##  [7,] 4538329 1718509 146138.00 19248.11        NA       NA      NA       NA       NA
##  [8,] 4578285 1618201  77053.93       NA        NA       NA      NA       NA       NA
##  [9,] 4503558 1756243        NA       NA        NA       NA      NA       NA       NA
## [10,] 4575529      NA        NA       NA        NA       NA      NA       NA       NA
```

```r
cumulative_triangle(observed_data,
                    variable = 'size',
                    lower_na = TRUE)
```

```
##          [,1]    [,2]    [,3]    [,4]    [,5]    [,6]    [,7]    [,8]    [,9]   [,10]
##  [1,] 4532915 6272614 6394490 6420442 6429747 6436450 6445929 6452146 6455614 6455614
##  [2,] 4414954 6088531 6195946 6215419 6215419 6215419 6215419 6215419 6215419      NA
##  [3,] 4786833 6572449 6721689 6781037 6795895 6800299 6800299 6800299      NA      NA
##  [4,] 4653451 6267504 6395404 6431618 6440919 6440919 6440919      NA      NA      NA
##  [5,] 4321019 6230104 6389351 6438535 6450263 6450263      NA      NA      NA      NA
##  [6,] 4417129 6139280 6243187 6308779 6323908      NA      NA      NA      NA      NA
##  [7,] 4538329 6256837 6402975 6422224      NA      NA      NA      NA      NA      NA
##  [8,] 4578285 6196486 6273540      NA      NA      NA      NA      NA      NA      NA
##  [9,] 4503558 6259801      NA      NA      NA      NA      NA      NA      NA      NA
## [10,] 4575529      NA      NA      NA      NA      NA      NA      NA      NA      NA
```

Claims reserving with triangles

Mack chain ladder method

Assumptions Mack (1993) **chain ladder method**:

1. There exist development factors f_j such that

$$E(C_{i,j+1} \mid C_{i,1}, \ldots, C_{i,j}) = C_{i,j} \cdot f_j \quad \text{for} \quad 1 \leq i \leq l, 1 \leq j \leq l-1,$$

   with l the dimension of the runoff triangle.

2. There exist parameters σ_j such that

$$Var(C_{i,j+1} \mid C_{i,1}, \ldots, C_{i,j}) = C_{i,j} \cdot \sigma_j^2 \quad \text{for} \quad 1 \leq i \leq l, 1 \leq j \leq l-1.$$

3. Occurrence years are independent.

Remarks:

- Method based on the **cumulative triangle**!
- Assumption 1. and 2. add an **Markov property**. Future evolutions of the reserve depend only on the last known situation.

Mack chain ladder method (continued)

Estimating the **development factors** $\hat{f}_j$:

$$\hat{f}_j = \frac{\sum_{i=1}^{l-j} C_{i,j+1}}{\sum_{i=1}^{l-j} C_{i,j}}.$$

```
triangle <- cumulative_triangle(observed_data, variable = 'size')
l <- nrow(triangle)
f <- rep(0, l-1)

for(j in 1:(l-1)) {
  f[j] <- sum(triangle[1:(l-j), j+1]) / sum(triangle[1:(l-j), j])
}

f
```

```
## [1] 1.381312 1.019846 1.006146 1.001563 1.000344 1.000366 1.000319 1.000274 1.000000
```

Development factors should converge to 1 for high development years.

Mack chain ladder method (continued)

Use the development factors to estimate cells in the lower triangle ($i + j > l + 1$):

$$\hat{C}_{i,j} = \hat{C}_{i,j-1} \cdot \hat{f}_{j-1} \quad \text{for} \quad i + j > l + 1.$$

```r
triangle_completed <- triangle
for(j in 2:l) {
  triangle_completed[l:(l-j+2), j] <- triangle_completed[l:(l-j+2), j-1] * f[j-1]
}
```

Completed cumulative triangle:

```r
triangle_completed
```

```
##          [,1]    [,2]    [,3]    [,4]    [,5]    [,6]    [,7]    [,8]    [,9]   [,10]
##  [1,] 4532915 6272614 6394490 6420442 6429747 6436450 6445929 6452146 6455614 6455614
##  [2,] 4414954 6088531 6195946 6215419 6215419 6215419 6215419 6215419 6215419 6215419
##  [3,] 4786833 6572449 6721689 6781037 6795895 6800299 6800299 6800299 6802161 6802161
##  [4,] 4653451 6267504 6395404 6431618 6440919 6440919 6440919 6442977 6444741 6444741
##  [5,] 4321019 6230104 6389351 6438535 6450263 6450263 6452624 6454685 6456452 6456452
##  [6,] 4417129 6139280 6243187 6308779 6323908 6326081 6328397 6330418 6332152 6332152
##  [7,] 4538329 6256837 6402975 6422224 6432261 6434470 6436826 6438882 6440645 6440645
##  [8,] 4578285 6196486 6273540 6312100 6321965 6324137 6326452 6328473 6330206 6330206
##  [9,] 4503558 6259801 6384034 6423273 6433312 6435522 6437878 6439934 6441698 6441698
## [10,] 4575529 6320235 6445667 6485285 6495420 6497652 6500030 6502107 6503887 6503887
```

Completed incremental triangle:

```r
require(ChainLadder)
cum2incr(triangle_completed)
```

```
##          [,1]    [,2]      [,3]     [,4]      [,5]     [,6]     [,7]     [,8]     [,9
##  [1,] 4532915 1739699 121876.31 25951.66  9304.915 6703.063 9478.770 6217.266 3468.36
##  [2,] 4414954 1673577 107415.07 19472.27     0.000    0.000    0.000    0.000    0.00
##  [3,] 4786833 1785616 149239.81 59348.18 14857.743 4404.145    0.000    0.000 1861.91
##  [4,] 4653451 1614053 127900.21 36213.57  9301.109    0.000    0.000 2057.632 1764.07
##  [5,] 4321019 1909085 159246.74 49184.25 11727.691    0.000 2361.270 2061.371 1767.28
##  [6,] 4417129 1722151 103906.89 65592.12 15129.218 2172.474 2315.810 2021.685 1733.26
##  [7,] 4538329 1718509 146138.00 19248.11 10037.169 2209.697 2355.488 2056.324 1762.95
##  [8,] 4578285 1618201  77053.93 38559.90  9865.059 2171.807 2315.098 2021.064 1732.72
##  [9,] 4503558 1756243 124232.58 39239.04 10038.809 2210.058 2355.873 2056.660 1763.24
## [10,] 4575529 1744706 125431.95 39617.87 10135.726 2231.395 2378.618 2076.516 1780.26
```

Mack chain ladder method (continued)

The estimated reserve is the sum of the estimates in the **lower half** of the **incremental runoff triangle**.

```
triangle_completed_incr <- cum2incr(triangle_completed)
lower_triangle <- row(triangle_completed_incr) + col(triangle_completed_incr) > l+1

reserve_cl <- sum(triangle_completed_incr[lower_triangle])

data.frame(reserve_cl = reserve_cl,
           reserve_actual = reserve_actual,
           difference = reserve_cl - reserve_actual,
           relative_difference_pct = (reserve_cl - reserve_actual) / reserve_actual * 100)
```

```
##   reserve_cl reserve_actual difference relative_difference_pct
## 1    2205459        2468246  -262787.3               -10.64672
```

Mack chain ladder method (continued)

These calculations are already implemented in the {ChainLadder} package.

```r
require(ChainLadder)
triangle <- cumulative_triangle(observed_data, variable = 'size')
MackChainLadder(triangle)
```

```
## MackChainLadder(Triangle = triangle)
##
##       Latest Dev.To.Date  Ultimate      IBNR Mack.S.E CV(IBNR)
## 1  6,455,614       1.000 6,455,614         0        0      NaN
## 2  6,215,419       1.000 6,215,419         0    1,243      Inf
## 3  6,800,299       1.000 6,802,161     1,862    3,366   1.8079
## 4  6,440,919       0.999 6,444,741     3,822    5,265   1.3777
## 5  6,450,263       0.999 6,456,452     6,190    7,483   1.2089
## 6  6,323,908       0.999 6,332,152     8,243    8,128   0.9860
## 7  6,422,224       0.997 6,440,645    18,422   10,090   0.5477
## 8  6,273,540       0.991 6,330,206    56,666   22,280   0.3932
## 9  6,259,801       0.972 6,441,698   181,896   35,842   0.1970
## 10 4,575,529       0.704 6,503,887 1,928,358  137,089   0.0711
##
##                   Totals
## Latest:   62,217,515.46
## Dev:               0.97
## Ultimate: 64,422,974.02
## IBNR:      2,205,458.56
## Mack.S.E     148,272.46
## CV(IBNR):          0.07
```

`Latest`: Amount already paid

`Ultimate`: Estimated total amount (= amount paid + reserve)

`IBNR`: Estimated total reserve, IBNR + RBNS!

`Mack.S.E`: Estimated standard deviation of the reserve

GLM chain ladder method

The chain ladder reserve estimate can also be obtained by assuming a Poisson distribution for the incremental runoff triangle with multiplicative mean, i.e.

$$X_{i,j} \sim \texttt{POI}(\alpha_i \cdot \beta_j), \quad 1 \leq i,j \leq l.$$

```r
triangle <- incremental_triangle(observed_data,
                                 variable = 'size')

triangle_long <- data.frame(
  occ.year = as.numeric(row(triangle)),
  dev.year = as.numeric(col(triangle)),
  size = as.numeric(triangle))
```

```r
head(triangle_long)
```

```
##   occ.year dev.year    size
## 1        1        1 4532915
## 2        2        1 4414954
## 3        3        1 4786833
## 4        4        1 4653451
## 5        5        1 4321019
## 6        6        1 4417129
```

GLM chain ladder method (continued)

The chain ladder reserve estimate can also be obtained by assuming a Poisson distribution for the incremental runoff triangle with multiplicative mean, i.e.

$$X_{i,j} \sim \mathtt{POI}(\alpha_i \cdot \beta_j), \quad 1 \leq i, j \leq l.$$

This Poisson model can be estimated using the `glm` (Generalized Linear Model) routine in R:

$$\log(E(X_{i,j})) = \log(\alpha_i) + \log(\beta_j).$$

```
fit <- glm(size ~ factor(occ.year) + factor(dev.year),
           data = triangle_long,
           family = poisson(link = log))
```

`factor`: Treat the input as a categorical instead of a numeric variable.

```
summary(fit)
```

```
##
## Call:
## glm(formula = size ~ factor(occ.year) + factor(dev.year), family = poisson(link = log
##     data = triangle_long)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -139.185   -64.152    -4.062    38.541   132.371
##
## Coefficients:
##                      Estimate Std. Error   z value Pr(>|z|)
## (Intercept)         1.533e+01  4.029e-04 38048.572  < 2e-16 ***
## factor(occ.year)2  -3.792e-02  5.620e-04   -67.473  < 2e-16 ***
## factor(occ.year)3   5.229e-02  5.495e-04    95.155  < 2e-16 ***
## factor(occ.year)4  -1.686e-03  5.570e-04    -3.027 0.002472 **
## factor(occ.year)5   1.298e-04  5.568e-04     0.233 0.815596
## factor(occ.year)6  -1.931e-02  5.596e-04   -34.510  < 2e-16 ***
## factor(occ.year)7  -2.322e-03  5.574e-04    -4.165 3.12e-05 ***
## factor(occ.year)8  -1.962e-02  5.608e-04   -34.979  < 2e-16 ***
## factor(occ.year)9  -2.158e-03  5.615e-04    -3.843 0.000121 ***
## factor(occ.year)10  7.450e-03  6.171e-04    12.072  < 2e-16 ***
## factor(dev.year)2  -9.641e-01  2.982e-04 -3233.538  < 2e-16 ***
## factor(dev.year)3  -3.597e+00  1.017e-03 -3537.016  < 2e-16 ***
## factor(dev.year)4  -4.749e+00  1.915e-03 -2480.500  < 2e-16 ***
## factor(dev.year)5  -6.112e+00  4.076e-03 -1499.723  < 2e-16 ***
```

GLM chain ladder method

Predict the cells in the lower triangle

```
lower_triangle <- triangle_long$occ.year +
  triangle_long$dev.year > l + 1

triangle_long$size[lower_triangle] <-
  predict(fit, newdata = triangle_long[lower_triangle, ], type

triangle_long %>%
  pivot_wider(values_from = size,
              names_from = dev.year,
              names_prefix = 'DY.')
```

```
## # A tibble: 10 x 11
##    occ.year     DY.1     DY.2    DY.3   DY.4   DY.5  DY.6  DY.7  DY.8  DY.9    DY.10
##       <dbl>    <dbl>    <dbl>   <dbl>  <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl>
##  1        1 4532915. 1739699. 121876. 25952.  9305. 6703. 9479. 6217. 3468. 0
##  2        2 4414954. 1673577. 107415. 19472.     0     0     0     0     0  0.000239
##  3        3 4786833. 1785616. 149240. 59348. 14858. 4404.    0     0  1862. 0.000261
##  4        4 4653451. 1614053. 127900. 36214.  9301.    0     0  2058. 1764. 0.000247
##  5        5 4321019. 1909085. 159247. 49184. 11728.    0  2361. 2061. 1767. 0.000248
##  6        6 4417129. 1722151. 103907. 65592. 15129. 2172. 2316. 2022. 1733. 0.000243
##  7        7 4538329. 1718509. 146138. 19248. 10037. 2210. 2355. 2056. 1763. 0.000247
##  8        8 4578285. 1618201.  77054. 38560.  9865. 2172. 2315. 2021. 1733. 0.000243
##  9        9 4503558. 1756243. 124233. 39239. 10039. 2210. 2356. 2057. 1763. 0.000247
## 10       10 4575529. 1744706. 125432. 39618. 10136. 2231. 2379. 2077. 1780. 0.000250
```

Compute the reserve

```
reserve_glm <- sum(triangle_long$size[lower_triangle])
reserve_glm
```

```
## [1] 2205459
```

Your turn

In certain lines of business, the **Corona pandemic** and social distancing measures have **reduced claim frequency**.

The R script contains code to simulate a 50% reduction in claim frequency in the months March, April and May of 2019. The new data sets are named `observed_data_covid` and `unobserved_data_covid`.

In this exercise you will investigate the effect of this reduction in claim frequency on the accuracy of the chain ladder method.

1. Compute the actual reserve from the covid data set.
2. Estimate the reserve using the chain ladder method from the {ChainLadder} package.
3. Compute the difference between the estimated and actual reserve. How many standard deviations is this error?

For **Q.1** we compute the actual reserve from the unobserved data.

```
reserve_covid_actual <- sum(unobserved_data_covid$size)
reserve_covid_actual
```

```
## [1] 2389277
```

The reserve is smaller since less claims have occurred in the past. However, the difference is small since many claims from these months have already settled.

```
c(reserve_actual, reserve_covid_actual)
```

```
## [1] 2468246 2389277
```

The effect on the amount already paid is larger.

```
c(already_paid_no_covid = sum(observed_data$size),
  already_paid_covid = sum(observed_data_covid$size))
```

```
## already_paid_no_covid    already_paid_covid
##              62217515              61504837
```

For **Q.2** we appy the chain ladder method to the cumulative runoff triangle.

```
require(ChainLadder)
triangle <- cumulative_triangle(observed_data_covid,
                                variable = 'size')
cl <- MackChainLadder(triangle)
cl
```

```
## MackChainLadder(Triangle = triangle)
##
##       Latest Dev.To.Date  Ultimate      IBNR Mack.S.E CV(IBNR)
## 1  6,455,614       1.000 6,455,614         0        0      NaN
## 2  6,215,419       1.000 6,215,419         0    1,243      Inf
## 3  6,800,299       1.000 6,802,161     1,862    3,366   1.8079
## 4  6,440,919       0.999 6,444,741     3,822    5,265   1.3777
## 5  6,450,263       0.999 6,456,452     6,190    7,483   1.2089
## 6  6,323,908       0.999 6,332,152     8,243    8,128   0.9860
## 7  6,422,224       0.997 6,440,645    18,422   10,090   0.5477
## 8  6,273,540       0.991 6,330,206    56,666   22,280   0.3932
## 9  6,259,801       0.972 6,441,698   181,896   35,842   0.1970
## 10 3,862,850       0.704 5,490,850 1,628,000  124,951   0.0768
##
##                 Totals
## Latest:   61,504,836.84
## Dev:               0.97
## Ultimate: 63,409,936.76
## IBNR:      1,905,099.93
## Mack.S.E     136,795.35
## CV(IBNR):          0.07
```

For **Q.3** we compute the prediction error of the chain ladder method.

```r
ultimate <- sum(cum2incr(cl$FullTriangle))
already_paid <- sum(cum2incr(cl$Triangle), na.rm = TRUE)

reserve_covid_cl <-  ultimate - already_paid
sigma_cl <- as.numeric(cl$Total.Mack.S.E)

error = reserve_covid_actual - reserve_covid_cl

round(c(error = error,
        pct_error = error / reserve_covid_actual * 100,
        std.dev = error / sigma_cl),2)
```

```
##     error pct_error   std.dev
## 484177.34     20.26      3.54
```

The reduction in claim frequency in the months March, April and May has reduced the accuracy of the chain ladder method.

Actual triangle:

```
##           [,1]    [,2]      [,3]     [,4]      [,5]     [,6]     [,7]     [,8]     [,9
##  [1,] 4532915 1739699 121876.31 25951.66  9304.915 6703.063 9478.770 6217.266 3468.36
##  [2,] 4414954 1673577 107415.07 19472.27     0.000    0.000    0.000    0.000    0.00
##  [3,] 4786833 1785616 149239.81 59348.18 14857.743 4404.145    0.000    0.000    0.00
##  [4,] 4653451 1614053 127900.21 36213.57  9301.109    0.000    0.000    0.000    0.00
##  [5,] 4321019 1909085 159246.74 49184.25 11727.691    0.000 8367.837 1250.830    0.00
##  [6,] 4417129 1722151 103906.89 65592.12 15129.218    0.000    0.000    0.000    0.00
##  [7,] 4538329 1718509 146138.00 19248.11  8206.603 1255.568    0.000    0.000    0.00
##  [8,] 4578285 1618201  77053.93 20586.47  4744.763    0.000    0.000    0.000    0.00
##  [9,] 4503558 1756243 153002.08 50348.72 10709.289 4574.275    0.000    0.000    0.00
## [10,] 3862850 1801338 214885.47 77394.80 26305.781 6306.792    0.000    0.000    0.00
```

Predicted triangle:

```
##       dev
## origin       1       2         3        4         5        6        7        8
##      1  4532915 1739699 121876.31 25951.66  9304.915 6703.063 9478.770 6217.266 3468.3
##      2  4414954 1673577 107415.07 19472.27     0.000    0.000    0.000    0.000    0.0
##      3  4786833 1785616 149239.81 59348.18 14857.743 4404.145    0.000    0.000 1861.9
##      4  4653451 1614053 127900.21 36213.57  9301.109    0.000    0.000 2057.632 1764.0
##      5  4321019 1909085 159246.74 49184.25 11727.691    0.000 2361.270 2061.371 1767.2
##      6  4417129 1722151 103906.89 65592.12 15129.218 2172.474 2315.810 2021.685 1733.2
##      7  4538329 1718509 146138.00 19248.11 10037.169 2209.697 2355.488 2056.324 1762.9
##      8  4578285 1618201  77053.93 38559.90  9865.059 2171.807 2315.098 2021.064 1732.7
##      9  4503558 1756243 124232.58 39239.04 10038.809 2210.058 2355.873 2056.660 1763.2
##      10 3862850 1472953 105894.83 33447.04  8556.998 1883.835 2008.127 1753.080 1502.9
```

When the chain ladder method fails

Detecting when the chain ladder method fails

The chain ladder method assumes that claims from all occurrence years follow the **same development pattern**.
We construct various triangles to assess this assumption:

Number of open claims begin at the start of the year:

```
triangle_open <- incremental_triangle(
  observed_data_covid %>%
    mutate(open = calendar_year <= settlement_year),
  variable = 'open')
triangle_open
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,] 2933  802   37    7    2    1    1    1    1     0
##  [2,] 2886  756   34    6    1    0    0    0    0    NA
##  [3,] 3000  790   41    9    5    2    0    0   NA    NA
##  [4,] 2937  791   28    8    2    1    0   NA   NA    NA
##  [5,] 2908  795   44    9    2    1   NA   NA   NA    NA
##  [6,] 2862  770   34    7    3   NA   NA   NA   NA    NA
##  [7,] 2866  788   39    7   NA   NA   NA   NA   NA    NA
##  [8,] 2862  744   25   NA   NA   NA   NA   NA   NA    NA
##  [9,] 2867  772   NA   NA   NA   NA   NA   NA   NA    NA
## [10,] 2573   NA   NA   NA   NA   NA   NA   NA   NA    NA
```

Settlement probability:

```
triangle_settlement <- incremental_triangle(
  observed_data_covid %>%
    mutate(settlement = calendar_year == settlement_year),
  variable = 'settlement')

triangle_settlement / triangle_open
```

```
##            [,1]      [,2]      [,3]      [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,] 0.7565632 0.9538653 0.8108108 0.7142857  0.5    0    0    0    1   NaN
##  [2,] 0.7723493 0.9550265 0.8235294 0.8333333  1.0  NaN  NaN  NaN  NaN    NA
##  [3,] 0.7640000 0.9481013 0.7804878 0.4444444  0.6    1  NaN  NaN   NA    NA
##  [4,] 0.7647259 0.9646018 0.7142857 0.7500000  0.5    1  NaN   NA   NA    NA
##  [5,] 0.7568776 0.9446541 0.7954545 0.7777778  0.5    0   NA   NA   NA    NA
##  [6,] 0.7641509 0.9558442 0.7941176 0.5714286  1.0   NA   NA   NA   NA    NA
##  [7,] 0.7616888 0.9505076 0.8205128 0.5714286   NA   NA   NA   NA   NA    NA
##  [8,] 0.7672956 0.9663978 0.7600000        NA   NA   NA   NA   NA   NA    NA
##  [9,] 0.7697942 0.9572539        NA        NA   NA   NA   NA   NA   NA    NA
## [10,] 0.7298873        NA        NA        NA   NA   NA   NA   NA   NA    NA
```

The percentage of settled claims is lower in the last accident year. Indicating that more claims are still open.

Detecting when the chain ladder method fails

The chain ladder method assumes that claims from all occurrence years follow the **same development pattern**.
We construct various triangles to assess this assumption:

Percentage of open claims for which a payment is made.

```
triangle_payment <- incremental_triangle(
  observed_data_covid,
  variable = 'payment')

triangle_payment / triangle_open
```

```
##            [,1]      [,2]      [,3]      [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,] 0.7719059 0.6857855 0.5405405 0.7142857  1.0  1.0    1    1    1   NaN
##  [2,] 0.7862093 0.7050265 0.7941176 0.8333333  0.0  NaN  NaN  NaN  NaN    NA
##  [3,] 0.7690000 0.7050633 0.6341463 0.8888889  0.6  0.5  NaN  NaN   NA    NA
##  [4,] 0.7793667 0.6788875 0.8214286 1.0000000  0.5  0.0  NaN   NA   NA    NA
##  [5,] 0.7699450 0.7308176 0.7500000 0.8888889  0.5  0.0   NA   NA   NA    NA
##  [6,] 0.7725367 0.6909091 0.7058824 0.8571429  1.0   NA   NA   NA   NA    NA
##  [7,] 0.7665736 0.6814721 0.7692308 0.7142857   NA   NA   NA   NA   NA    NA
##  [8,] 0.7861635 0.6854839 0.7200000        NA   NA   NA   NA   NA   NA    NA
##  [9,] 0.7921172 0.7046632        NA        NA   NA   NA   NA   NA   NA    NA
## [10,] 0.7765255        NA        NA        NA   NA   NA   NA   NA   NA    NA
```

Average amount paid given a payment is made.

```
triangle_size <- incremental_triangle(
  observed_data_covid,
  variable = 'size')

triangle_size / triangle_payment
```

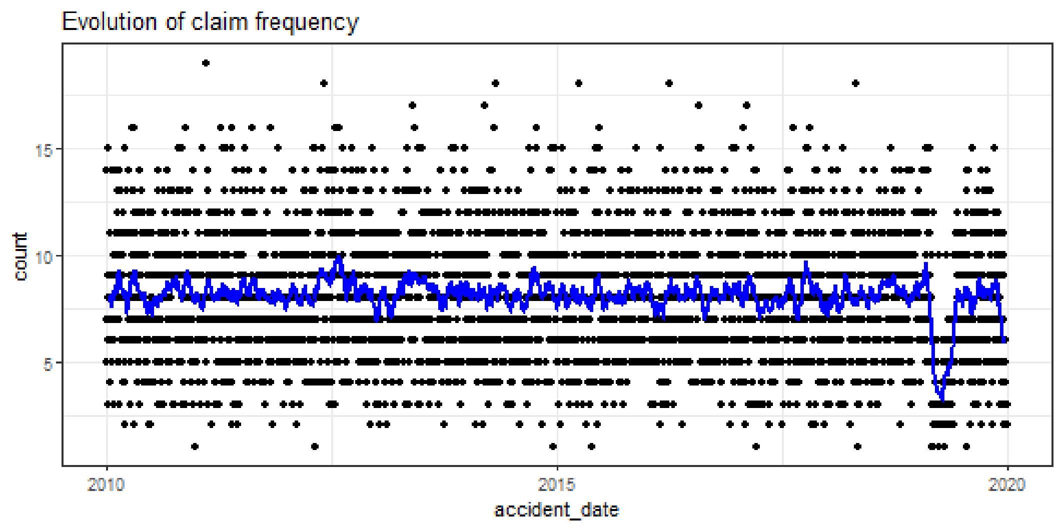
```
##           [,1]     [,2]     [,3]      [,4]      [,5]     [,6]    [,7]     [,8]     [,
##  [1,] 2002.171 3163.090 6093.815  5190.332  4652.457 6703.063 9478.77 6217.266 3468.3
##  [2,] 1945.771 3139.920 3978.336  3894.454       NaN      NaN     NaN      NaN      N
##  [3,] 2074.917 3205.774 5739.993  7418.522  4952.581 4404.145     NaN      NaN
##  [4,] 2032.962 3005.686 5560.879  4526.696  9301.109      NaN     NaN       NA
##  [5,] 1929.888 3285.860 4825.659  6148.031 11727.691      NaN      NA       NA
##  [6,] 1997.797 3237.126 4329.454 10932.020  5043.073       NA      NA       NA
##  [7,] 2065.694 3200.202 4871.267  3849.622        NA       NA      NA       NA
##  [8,] 2034.793 3172.943 4280.774        NA        NA       NA      NA       NA
##  [9,] 1983.073 3228.389       NA        NA        NA       NA      NA       NA
## [10,] 1933.359       NA       NA        NA        NA       NA      NA       NA
```

Detecting when the chain ladder method fails

The chain ladder method assumes that claims from all occurrence years follow the **same development pattern**.

Inspecting the (granular) daily data:

```r
claims_covid <- observed_data_covid %>%
  group_by(accident_number) %>%
  slice(1) %>%
  ungroup()


occ_intensity <- claims_covid %>%
  group_by(accident_date) %>%
  summarise(count = n())

require(zoo)
occ_intensity$moving_average <-
  rollmean(occ_intensity$count, 30, na.pad = TRUE)

ggplot(occ_intensity) +
  theme_bw() +
  geom_point(aes(x = accident_date, y = count)) +
  geom_line(aes(x = accident_date, y = moving_average),
            size = 1, color = 'blue') +
  ggtitle('Evolution of claim frequency')
```



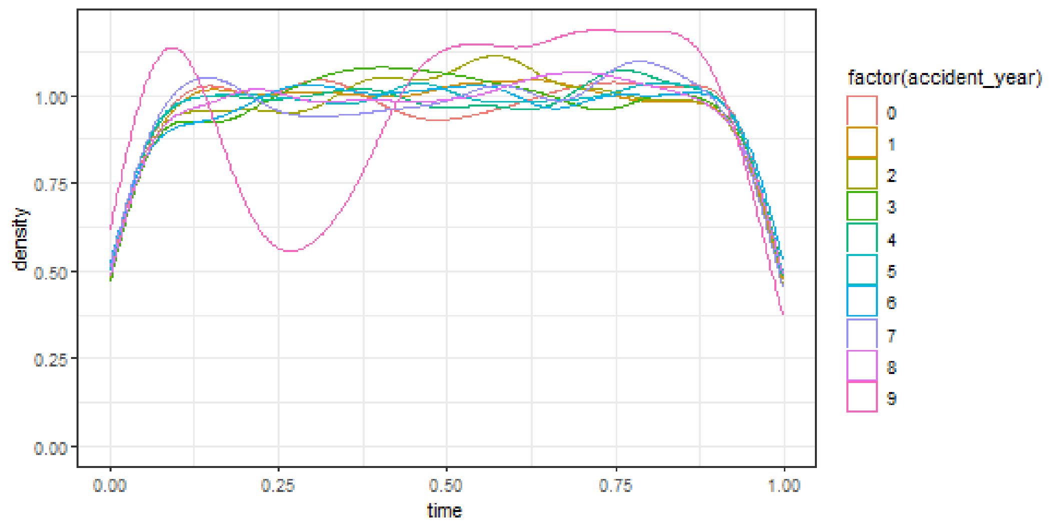
The moving average indicates a period with decreased claim frequency in 2019

Detecting when the chain ladder method fails

The chain ladder method assumes that claims from all occurrence years follow the **same development pattern**.
Inspecting the (granular) daily data:

```
require(lubridate)
claims_covid <- claims_covid %>%
  mutate(start_year = floor_date(accident_date, unit = 'year'),
         time = as.numeric(accident_date - start_year) / 366)

ggplot(claims_covid) +
  theme_bw() +
  geom_density(aes(x = time,
                   group = factor(accident_year),
                   color = factor(accident_year)))
```

Density of when claims occur within each accident year.
The chain ladder method works best when the densities look similar for all accident years.



The occurrence intensity in accident year 9 deviates from the other accident years.

Fixing the chain ladder method

Improve the performance of the chain ladder method by making the **data more homogeneous**:

- **Group** claims by:
  - claim characteristics, e.g. separate triangle for fire and water related claims in home insurance.
  - occurrence time within a calendar year
- **Omit**:
  - deviating years/cells, especially if they occurred long ago
- **Adjust** for:
  - inflation

The appropriate method(s) depend on the data inspection from the previous slides.

Fixing the chain ladder method (continued)

Applied to the **covid data set**:

- claim frequency has changed in the months March, April, May
- only 2019, accident year 9, is affected

As 2019 is a recent calendar year, we cannot omit accident year 9 from the data.

We improve homogeneity in the data by splitting claims in two groups:

- group A: claims that have occurred in March, April or May
- group B: claims that have not occurred in March, April or May

```
observed_data_covid <- observed_data_covid %>%
  mutate(occurrence_month = as.numeric(format(accident_date, '%m')))

observed_data_covidA <- observed_data_covid %>%
  filter(occurrence_month %in% c(3, 4, 5))

observed_data_covidB <- observed_data_covid %>%
  filter(!(occurrence_month %in% c(3, 4, 5)))
```

Your turn

In this exercise you compute the chain ladder reserve for claims in group A and B.

1. Choose data set A or B and analyse the stability of the chosen data set using some of the presented detection tools.

2. Compute the chain ladder reserve for group A and B separately.
   Combine these estimates to obtain an estimate for the total reserve.

3. Compare the new reserve estimate with the estimate obtained earlier using the chain ladder method without splitting the data.

4. Compute the standard error of the reserve estimate using the formula

$$\sigma^2 = \sigma_A^2 + \sigma_B^2.$$

For **Q.1**, analysing the stability of data set A.

Number of open claims begin dev.year:

```
triangle_open <- incremental_triangle(
  observed_data_covidA %>%
    mutate(open = calendar_year <= settlement_year),
  variable = 'open')
triangle_open
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,]  775   53    7    1    1    0    0    0    0     0
##  [2,]  748   38    0    0    0    0    0    0    0    NA
##  [3,]  773   47    5    1    1    0    0    0   NA    NA
##  [4,]  781   52    3    0    0    0    0   NA   NA    NA
##  [5,]  757   63    7    2    1    0   NA   NA   NA    NA
##  [6,]  730   38    5    1    1   NA   NA   NA   NA    NA
##  [7,]  750   59    2    1   NA   NA   NA   NA   NA    NA
##  [8,]  704   51    8   NA   NA   NA   NA   NA   NA    NA
##  [9,]  770   40   NA   NA   NA   NA   NA   NA   NA    NA
## [10,]  367   NA   NA   NA   NA   NA   NA   NA   NA    NA
```

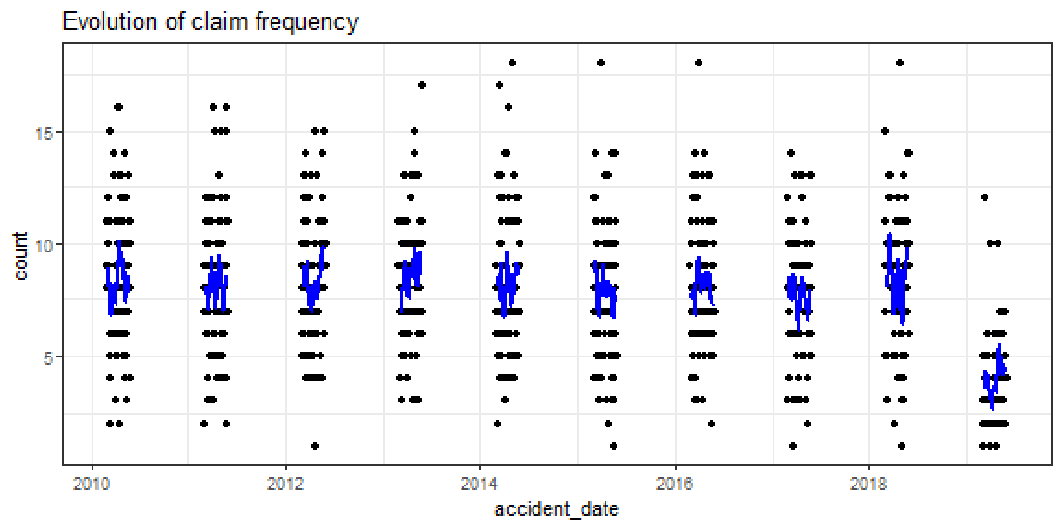
Settlement probability:

```
triangle_settlement <- incremental_triangle(
  observed_data_covidA %>%
    mutate(settlement = calendar_year == settlement_year),
  variable = 'settlement')

triangle_settlement / triangle_open
```

```
##            [,1]      [,2]      [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,] 0.9316129 0.8679245 0.8571429  0.0    1  NaN  NaN  NaN  NaN   NaN
##  [2,] 0.9491979 1.0000000       NaN  NaN  NaN  NaN  NaN  NaN  NaN    NA
##  [3,] 0.9391979 0.8936170 0.8000000  0.0    1  NaN  NaN  NaN   NA    NA
##  [4,] 0.9334187 0.9423077 1.0000000  NaN  NaN  NaN  NaN   NA   NA    NA
##  [5,] 0.9167768 0.8888889 0.7142857  0.5    1  NaN   NA   NA   NA    NA
##  [6,] 0.9479452 0.8684211 0.8000000  0.0    1   NA   NA   NA   NA    NA
##  [7,] 0.9213333 0.9661017 0.5000000  0.0   NA   NA   NA   NA   NA    NA
##  [8,] 0.9275568 0.8431373 0.6250000   NA   NA   NA   NA   NA   NA    NA
##  [9,] 0.9480519 0.8750000        NA   NA   NA   NA   NA   NA   NA    NA
## [10,] 0.9455041        NA        NA   NA   NA   NA   NA   NA   NA    NA
```

For **Q.1**, analysing the stability of data set A.

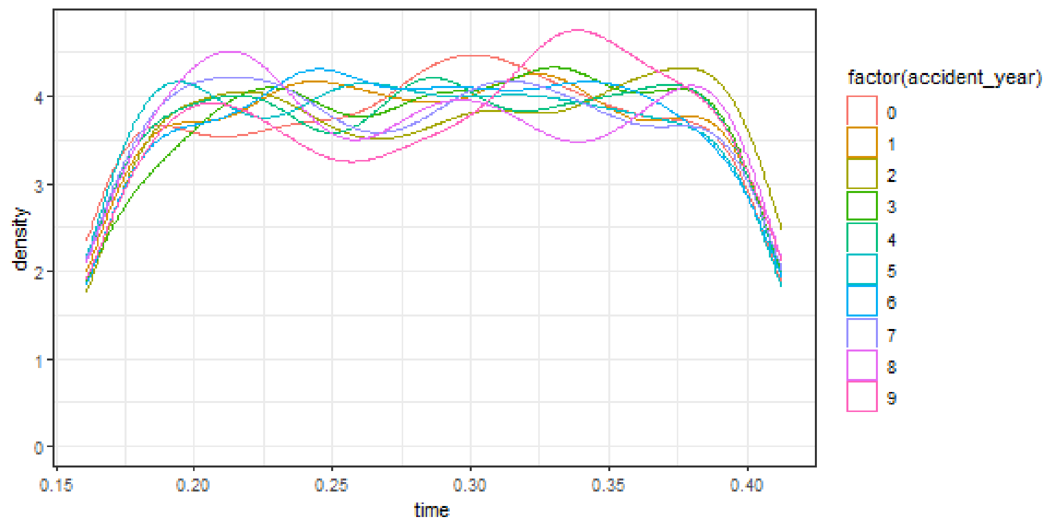
```
claims_covidA <- observed_data_covidA %>%
  group_by(accident_number) %>%
  slice(1) %>%
  ungroup()

occ_intensity <- claims_covidA %>%
  group_by(accident_date, accident_year) %>%
  summarise(count = n())

require(zoo)
occ_intensity <- occ_intensity %>%
  group_by(accident_year) %>%
  mutate(moving_average = rollmean(count, 14, na.pad = TRUE))


ggplot(occ_intensity) +
  theme_bw() +
  geom_point(aes(x = accident_date, y = count)) +
  geom_line(aes(x = accident_date, y = moving_average),
            size = 1, color = 'blue') +
  ggtitle('Evolution of claim frequency')
```



We compute moving averages within each accident year. Otherwise `rollmean` would naively assume that May 31, 2010 is followed by March 1, 2011.

For **Q.1**, analysing the stability of data set A.

```
require(lubridate)
claims_covidA <- claims_covidA %>%
  mutate(start_year = floor_date(accident_date, unit = 'year'),
         time = as.numeric(accident_date - start_year) / 366)

ggplot(claims_covidA) +
  theme_bw() +
  geom_density(aes(x = time,
                   group = factor(accident_year),
                   color = factor(accident_year)))
```



All densities have a similar, indicating that within each accident year the distributon of when claims occur is the same.

For **Q.1**, analysing the stability of data set B.

Number of open claims begin at the start of the year:

```
triangle_open <- incremental_triangle(
  observed_data_covidB %>%
    mutate(open = calendar_year <= settlement_year),
  variable = 'open')
triangle_open
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,] 2158  749   30    6    1    1    1    1    1     0
##  [2,] 2138  718   34    6    1    0    0    0    0    NA
##  [3,] 2227  743   36    8    4    2    0    0   NA    NA
##  [4,] 2156  739   25    8    2    1    0   NA   NA    NA
##  [5,] 2151  732   37    7    1    1   NA   NA   NA    NA
##  [6,] 2132  732   29    6    2   NA   NA   NA   NA    NA
##  [7,] 2116  729   37    6   NA   NA   NA   NA   NA    NA
##  [8,] 2158  693   17   NA   NA   NA   NA   NA   NA    NA
##  [9,] 2097  732   NA   NA   NA   NA   NA   NA   NA    NA
## [10,] 2206   NA   NA   NA   NA   NA   NA   NA   NA    NA
```

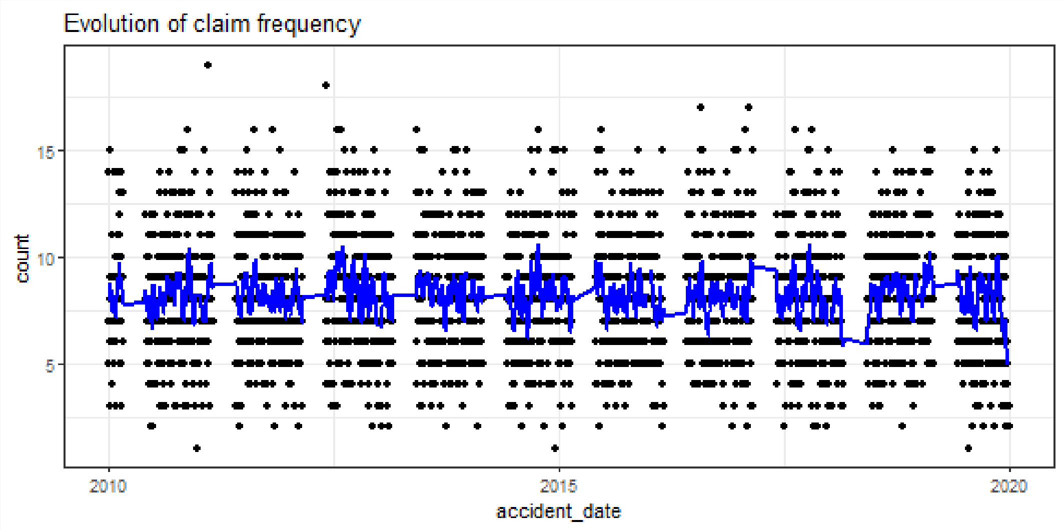
Settlement probability:

```
triangle_settlement <- incremental_triangle(
  observed_data_covidB %>%
    mutate(settlement = calendar_year == settlement_year),
  variable = 'settlement')

triangle_settlement / triangle_open
```

```
##            [,1]      [,2]      [,3]      [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,] 0.6936979 0.9599466 0.8000000 0.8333333  0.0    0    0    0    1   NaN
##  [2,] 0.7104771 0.9526462 0.8235294 0.8333333  1.0  NaN  NaN  NaN  NaN    NA
##  [3,] 0.7031881 0.9515478 0.7777778 0.5000000  0.5    1  NaN  NaN   NA    NA
##  [4,] 0.7036178 0.9661705 0.6800000 0.7500000  0.5    1  NaN   NA   NA    NA
##  [5,] 0.7006044 0.9494536 0.8108108 0.8571429  0.0    0   NA   NA   NA    NA
##  [6,] 0.7012195 0.9603825 0.7931034 0.6666667  1.0   NA   NA   NA   NA    NA
##  [7,] 0.7051040 0.9492455 0.8378378 0.6666667   NA   NA   NA   NA   NA    NA
##  [8,] 0.7150139 0.9754690 0.8235294        NA   NA   NA   NA   NA   NA    NA
##  [9,] 0.7043395 0.9617486        NA        NA   NA   NA   NA   NA   NA    NA
## [10,] 0.6940163        NA        NA        NA   NA   NA   NA   NA   NA    NA
```

For **Q.1**, analysing the stability of data set B.

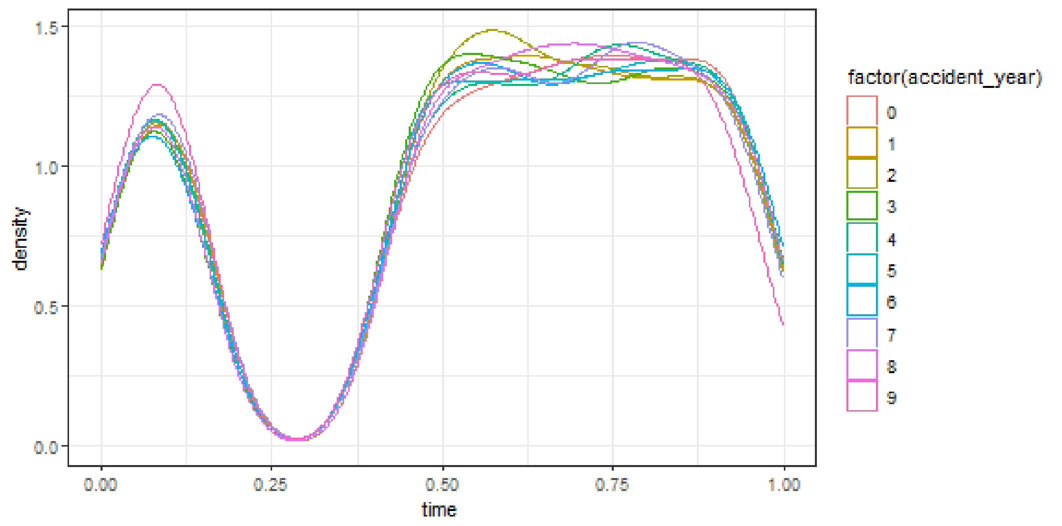
```
claims_covidB <- observed_data_covidB %>%
  group_by(accident_number) %>%
  slice(1) %>%
  ungroup()


occ_intensity <- claims_covidB %>%
  group_by(accident_date, accident_year) %>%
  summarise(count = n())

require(zoo)
occ_intensity <- occ_intensity %>%
  group_by(accident_year) %>%
  mutate(moving_average = rollmean(count, 14, na.pad = TRUE))

ggplot(occ_intensity) +
  theme_bw() +
  geom_point(aes(x = accident_date, y = count)) +
  geom_line(aes(x = accident_date, y = moving_average),
            size = 1, color = 'blue') +
  ggtitle('Evolution of claim frequency')
```



Claim frequency is stable over all accident years. No sudden jumps/decreases.

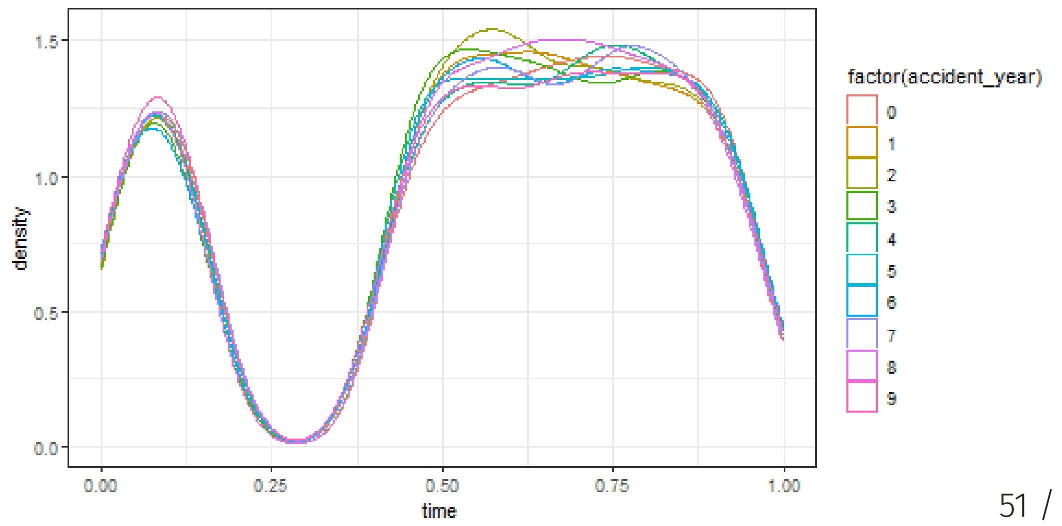
For **Q.1**, analysing the stability of data set B.

```r
require(lubridate)
claims_covidB <- claims_covidB %>%
  mutate(start_year = floor_date(accident_date, unit = 'year'),
         time = as.numeric(accident_date - start_year) / 366)

ggplot(claims_covidB) +
  theme_bw() +
  geom_density(aes(x = time,
                   group = factor(accident_year),
                   color = factor(accident_year)))
```



Since we only observe reported claims, we observe less claims from recent accident years around the end of the year. Correct for this by filtering claims that are reported in the year of occurrence.

```r
ggplot(claims_covidB %>%
         filter(reporting_year == accident_year)) +
  theme_bw() +
  geom_density(aes(x = time,
                   group = factor(accident_year),
                   color = factor(accident_year)))
```



For **Q.2** computing the chain ladder reserve

```r
require(ChainLadder)
triangleA <- cumulative_triangle(observed_data_covidA,
                                 variable = 'size')
clA <- MackChainLadder(triangleA)
clA
```

```
## MackChainLadder(Triangle = triangleA)
##
##       Latest Dev.To.Date  Ultimate      IBNR Mack.S.E CV(IBNR)
## 1  1,619,982       1.000 1,619,982  0.00e+00 0.00e+00      NaN
## 2  1,528,185       1.000 1,528,185  0.00e+00 1.44e-13      Inf
## 3  1,608,274       1.000 1,608,274 -4.66e-10 3.54e-10   -0.761
## 4  1,671,348       1.000 1,671,348 -4.66e-10 3.64e-10   -0.781
## 5  1,684,656       1.000 1,684,656 -4.66e-10 3.66e-10   -0.785
## 6  1,470,984       1.000 1,470,984 -4.66e-10 4.23e-10   -0.908
## 7  1,640,175       0.999 1,641,391  1.22e+03 2.38e+03    1.962
## 8  1,666,274       0.996 1,672,544  6.27e+03 4.74e+03    0.756
## 9  1,615,375       0.987 1,636,318  2.09e+04 1.18e+04    0.561
## 10   724,846       0.893   811,877  8.70e+04 3.14e+04    0.361
##
##                   Totals
## Latest:   15,230,099.47
## Dev:               0.99
## Ultimate: 15,345,559.70
## IBNR:        115,460.23
## Mack.S.E      34,373.24
## CV(IBNR):          0.30
```

```r
require(ChainLadder)
triangleB <- cumulative_triangle(observed_data_covidB,
                                 variable = 'size')
clB <- MackChainLadder(triangleB)
clB
```

```
## MackChainLadder(Triangle = triangleB)
##
##       Latest Dev.To.Date  Ultimate      IBNR Mack.S.E CV(IBNR)
## 1  4,835,633       1.000 4,835,633         0        0      NaN
## 2  4,687,234       1.000 4,687,234         0    1,212      Inf
## 3  5,192,025       1.000 5,193,916     1,892    3,398   1.7965
## 4  4,769,571       0.999 4,773,326     3,755    5,225   1.3914
## 5  4,765,606       0.999 4,771,681     6,074    7,422   1.2219
## 6  4,852,924       0.998 4,861,338     8,414    8,249   0.9803
## 7  4,782,048       0.996 4,799,125    17,077    9,690   0.5675
## 8  4,607,266       0.989 4,656,956    49,690   19,630   0.3950
## 9  4,644,426       0.967 4,804,801   160,374   39,868   0.2486
## 10 3,138,005       0.640 4,902,249 1,764,244  132,487   0.0751
##
##                   Totals
## Latest:   46,274,737.36
## Dev:               0.96
## Ultimate: 48,286,257.75
## IBNR:      2,011,520.39
## Mack.S.E     144,671.48
## CV(IBNR):          0.07
```

For **Q.3**, we combine the estimated reserves from A and B

```
ultimateA <- sum(cum2incr(clA$FullTriangle))
already_paidA <- sum(cum2incr(clA$Triangle), na.rm = TRUE)
reserveA <-  ultimateA - already_paidA

ultimateB <- sum(cum2incr(clB$FullTriangle))
already_paidB <- sum(cum2incr(clB$Triangle), na.rm = TRUE)
reserveB <-  ultimateB - already_paidB

reserve_covid_cl_split <- reserveA + reserveB

c(reserve_actual = reserve_covid_actual,
  reserve_cl = reserve_covid_cl,
  reserve_cl_split = reserve_covid_cl_split)
```

```
##   reserve_actual       reserve_cl reserve_cl_split
##          2389277          1905100          2126981
```

Splitting the data results in a more accurate reserve estimate.

For **Q.4**, we evaluate the relative performance of the model

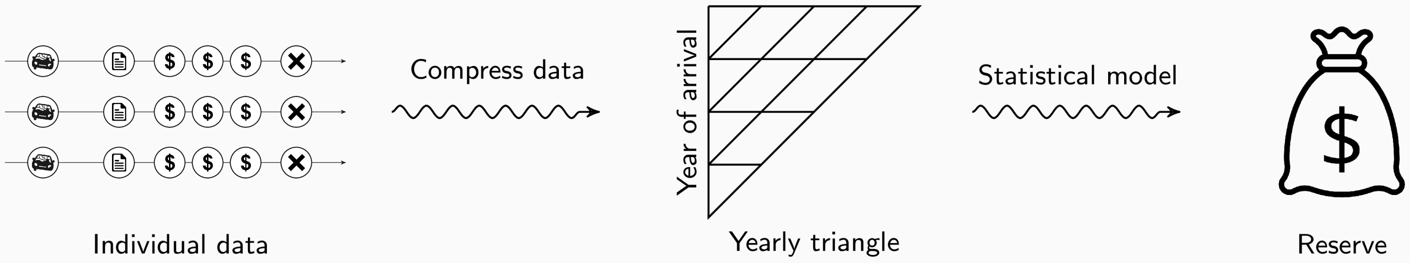
```
sigma = as.numeric(
  sqrt(clA$Total.Mack.S.E^2 + clB$Total.Mack.S.E^2))

data.frame(
  method = c('chain ladder', 'chain ladder split'),
  error = c(reserve_covid_cl - reserve_covid_actual,
            reserve_covid_cl_split - reserve_covid_actual),
  sigma = c(sigma_cl, sigma)) %>%
  mutate(pct_error = error / reserve_covid_actual * 100,
         std.dev = error / sigma)
```

	chain ladder	**chain ladder split**
error	-484177.34	-262296.65
sigma	136795.35	148698.88
pct_error	-20.26	-10.98
std.dev	-3.54	-1.76

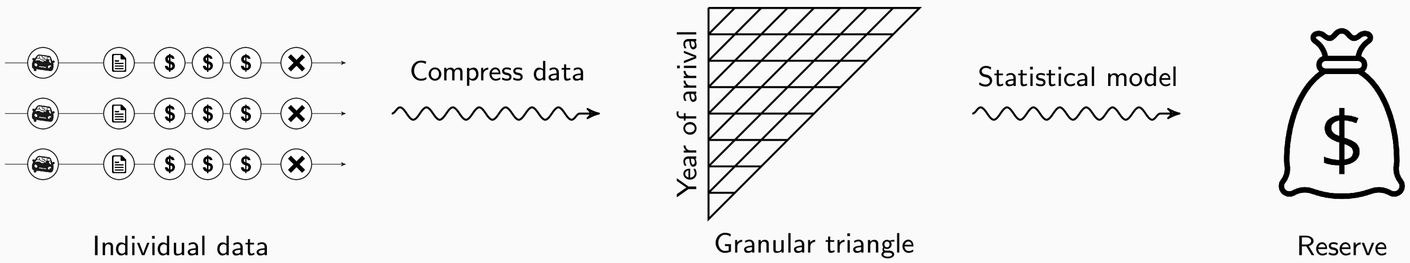
Research outlook

Research outlook

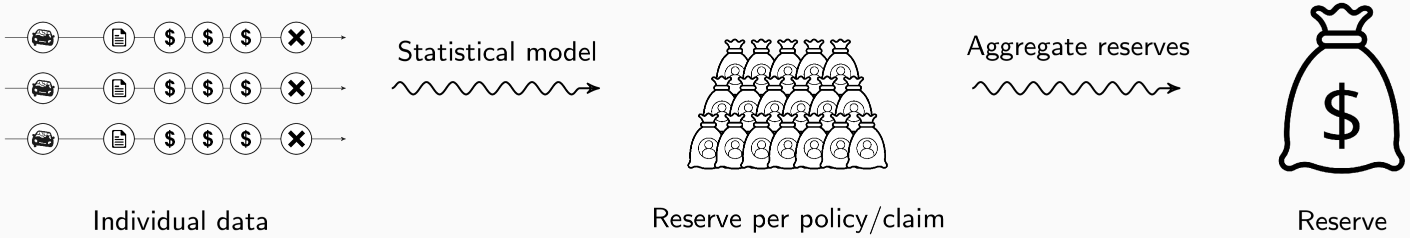
Aggregate reserving



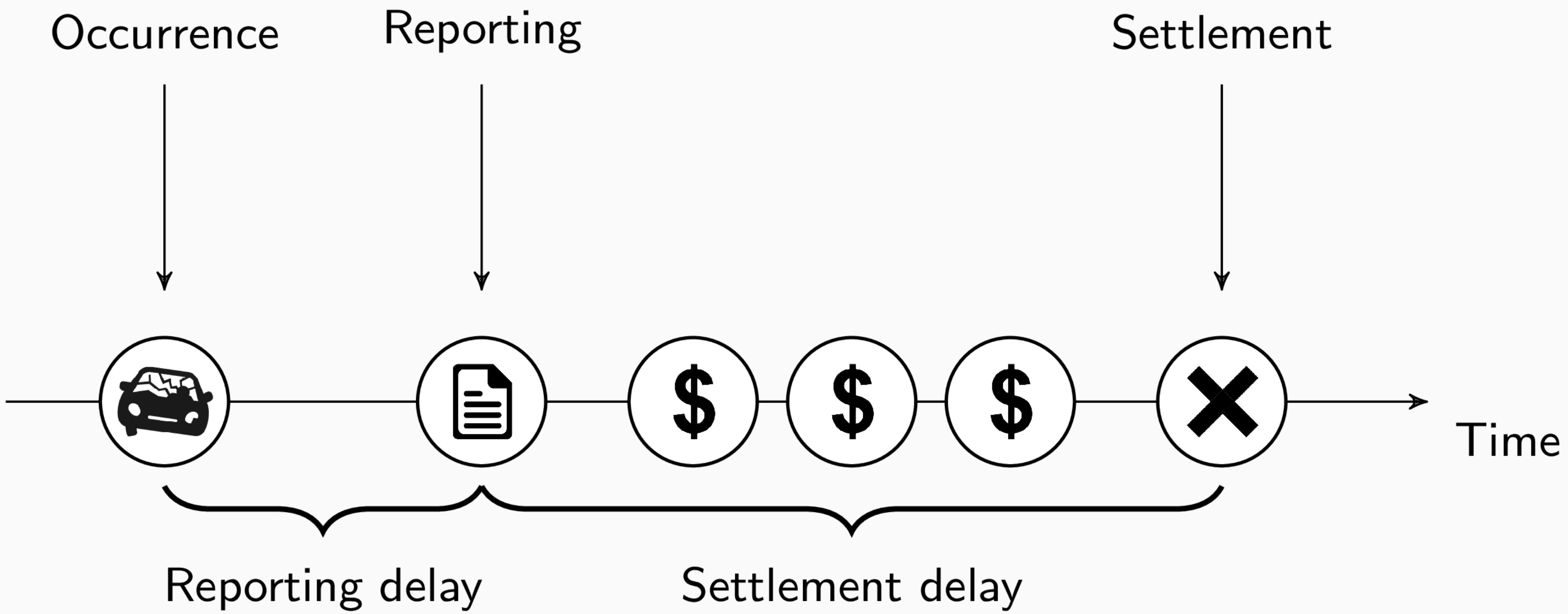
Granular reserving



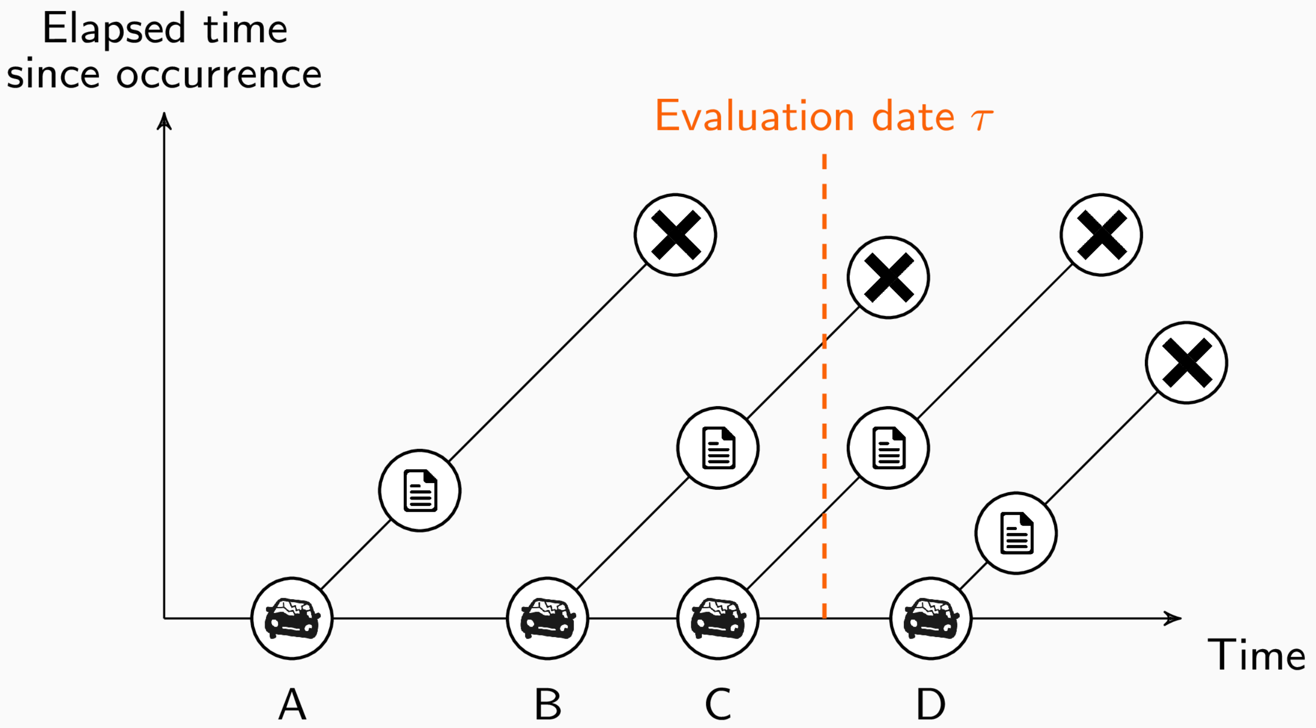
Individual reserving



Development of a individual claims



Development of a individual claims (continued)



Observed claims are **censored** due to **delays** (reporting, settlement) in the claim development process

IBNR reserve: future costs for claims that occurred, but are not yet reported (claim B)

RBNS reserve: future costs for claims that are reported, but are not yet settled (claim C)

Pricing: all costs for claims that will occur in **future** insured **exposure periods** (claim D)

The IBNR reserve

Following a frequency-severity decomposition, the IBNR reserve is the product of the expected **number of unreported claims** times the **expected severity per claim**.

- Insurance pricing covers the estimation of claim severity for new claims
- Reserving methods focus on predicting the **number of unreported claims**.

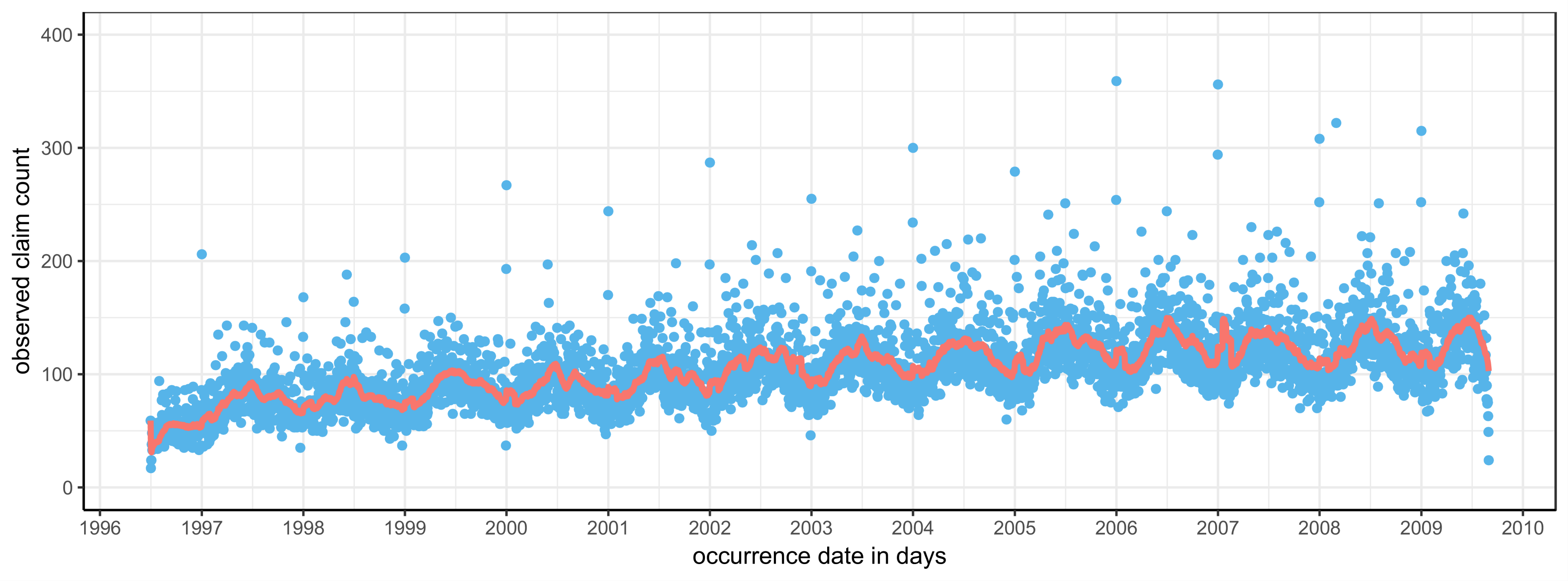
Our work on the topic:

- "Modeling the number of hidden events subject to observation delay". Jonas Crevecoeur, Katrien Antonio and Roel Verbelen. (2019). European Journal of Operational Research.

- "Modeling the occurrence of events subject to a reporting delay via an EM algorithm". Roel Verbelen, Katrien Antonio, Gerda Claeskens and Jonas Crevecoeur. (2019). Submitted.

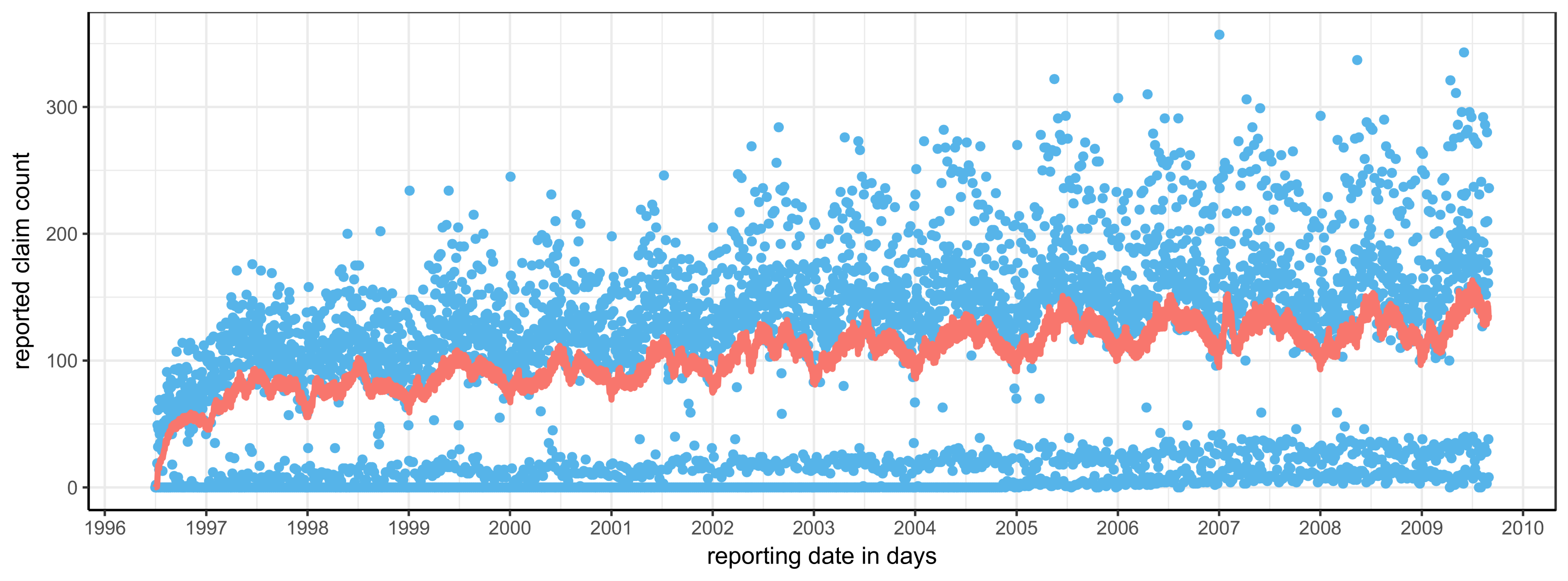
The IBNR reserve (continued)

Number of observed claim occurrences per day



The IBNR reserve (continued)

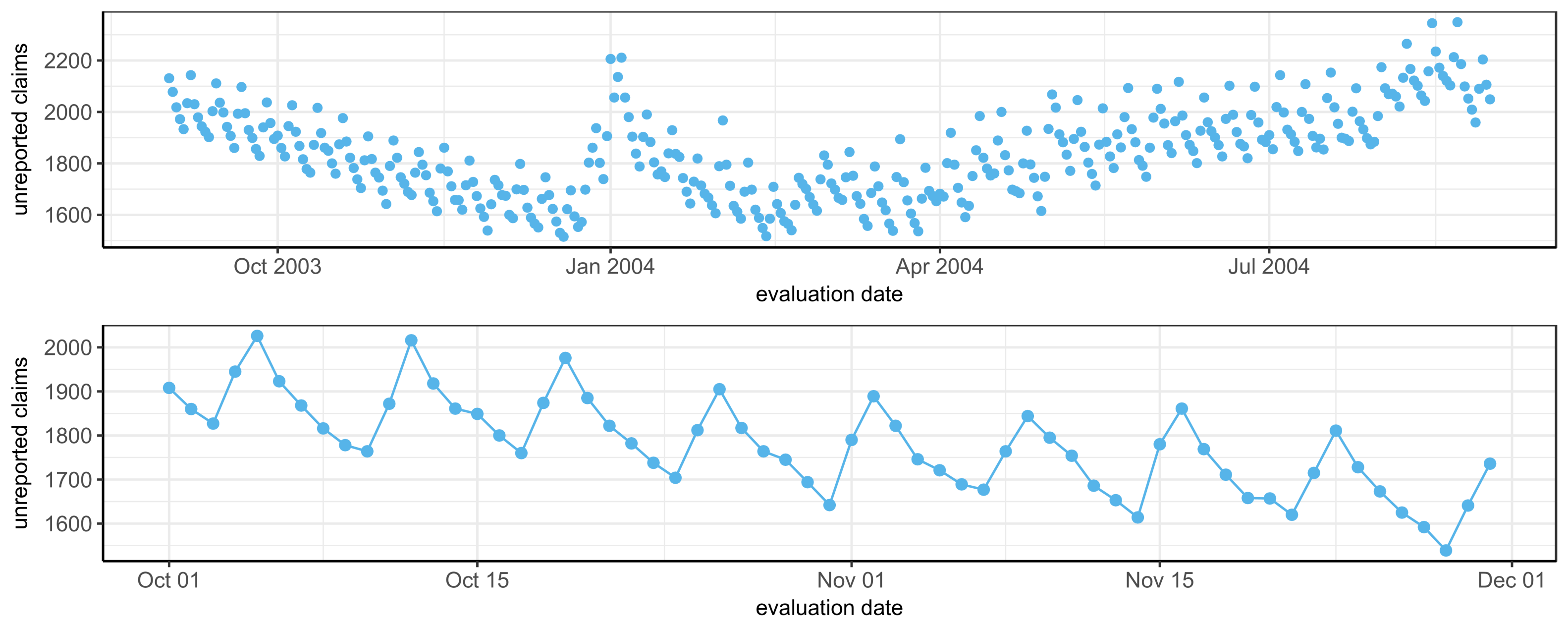
Number of reported claims per day



Almost no claims are reported on Saturdays, Sundays and holidays.

The IBNR reserve (continued)

Total number of unreported claims on each day, i.e. the number of claims that occur before the evaluation date, but are reported afterwards.



The number of unreported claims increases during the weekend (+10% on sunday) and around the end of the year (+30%).

The RBNS reserve

Predict the future costs of individual, reported claims.

Our work on the topic:

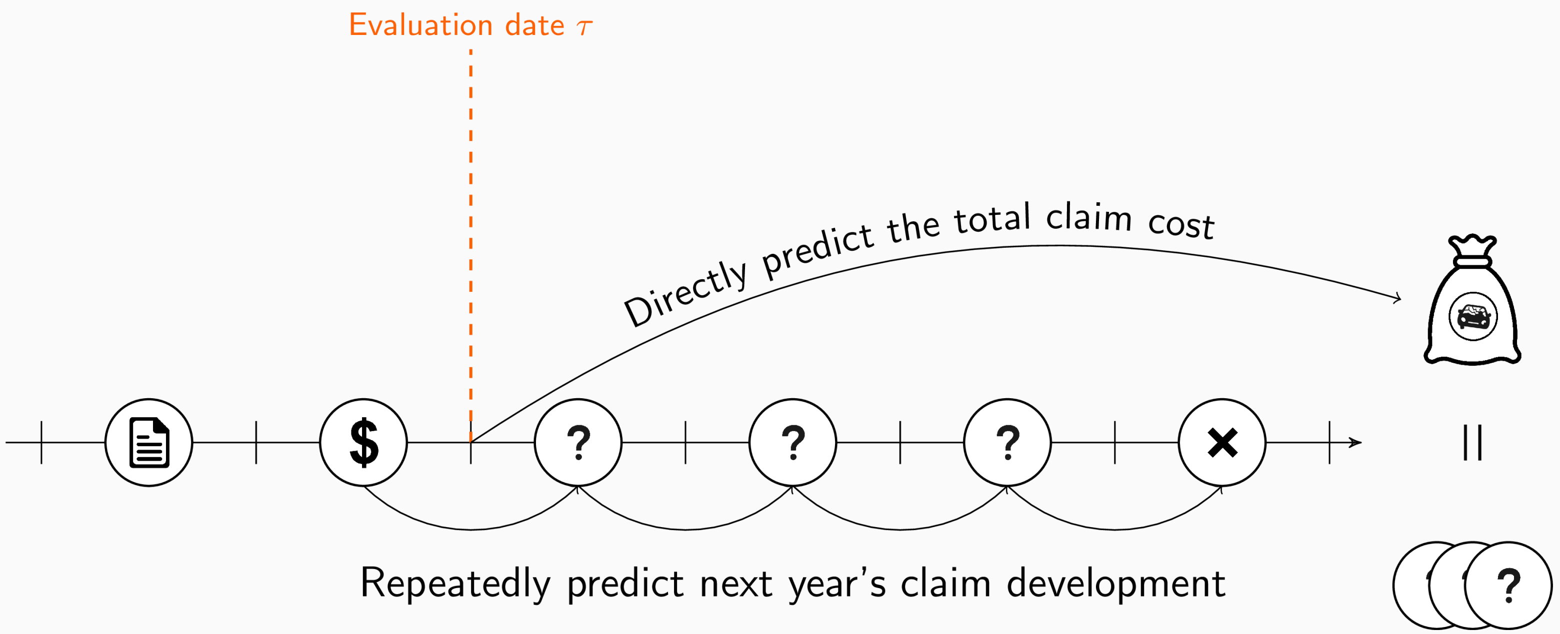
- "A hierarchical reserving model for non-life insurance claims". Jonas Crevecoeur and Katrien Antonio. (2020). Sumbitted.
- "Bridging the gap between pricing and reserving with an occurrence and development model for non-life insurance claims". Katrien Antonio and Jonas Crevecoeur. (2020). Working paper.

Methods implemented in the R package {hirem}, Hierarchical reserving models, available on GitHub.

The RBNS reserve (continued)

The hierarchical model is based on **two key ideas**:

1. Model the development of indiviudal claim in discrete time (steps of one year) using the observed development in previous years.



The RBNS reserve (continued)

The hierarchical model is based on **two key ideas**:

1. Model the development of indiviudal claim in discrete time (steps of one year) using the observed development in previous years.
2. Focus on all events registered over the lifetime of a claim

Common events registered over the lifetime of a claim:

- settlement
- payments
- initial incurred / changes in the incurred
- involvement lawyer

These events are **dependent**:

- if a claim is **settled**, there will be no **payments** in the future
- large **payments** are more likely when the outstanding **reserve** is large

Implementation in the {hirem} package

Implementation readily available from the {hirem} package on GitHub.

Events are added to the model as **layers**.

```
require(hirem)

individual_data <- individual_data %>%
  mutate(calendar_year = accident_year + development_year - 1,
         close = (settlement_year == calendar_year))

model <- hirem(individual_data %>% filter(calendar_year <= settlement_year)) %>%
  layer_glm('close', binomial(link = logit)) %>%
  layer_glm('payment', binomial(link = logit)) %>%
  layer_glm('size', Gamma(link = log),
            filter = function(data){data$payment == 1})

model <- fit(model,
             close = 'close ~ factor(development_year)',
             payment = 'payment ~ close + factor(development_year)',
             size = 'size ~ close + factor(development_year)')
```

Thanks!

Slides created with the R package xaringan.

Course material available via

https://github.com/katrienantonio/workshop-loss-reserv-fraud