Simulation of insurance data with **actuar**

Christophe Dutang
Université Paris Dauphine

Vincent Goulet
Université Laval

Mathieu Pigeon
Université du Québec à Montréal

Louis-Philippe Pouliot
Université Laval

1 Introduction

Package **actuar** provides functions to facilitate the generation of random variates from various probability models commonly used in actuarial applications. From the simplest to the most sophisticated, these functions are:

1. `rmixture` to simulate from discrete mixtures;

2. `rcompound` to simulate from compound models (and a simplified version, `rcompois` to simulate from the very common compound Poisson model);

3. `rcomphierarc`[1] to simulate from compound models where both the frequency and the severity components can have a hierarchical structure.

2 Simulation from discrete mixtures

A random variable is said to be a discrete mixture of the random variables with probability density functions $f_1, \ldots, f_n$ if its density can be written as

$$f(x) = p_1 f_1(x) + \cdots + p_n f_n(x) = \sum_{i=1}^{n} p_i f_i(x), \tag{1}$$

where $p_1, \ldots, p_n$ are probabilities (or weights) such that $p_i \geq 0$ and $\sum_{i=1}^{n} p_i = 1$.

---

[1] An alias for `simul` introduced in **actuar** 2.0-0.

Function `rmixture` makes it easy to generate random variates from such mixtures. The arguments of the function are:

1. `n` the number of variates to generate;

2. `probs` a vector of values that will be normalized internally to create the probabilities $p_1, \ldots, p_n$;

3. `models` a vector of expressions specifying the simulation models corresponding to the densities $f_1, \ldots, f_n$.

The specification of simulation models follows the syntax of `rcomphierarc` (explained in greater detail in section 4). In a nutshell, the models are expressed in a semi-symbolic fashion using an object of mode `"expression"` where each element is a complete call to a random number generation function, with the number of variates omitted.

The following example should clarify this concept.

Example 1. Let X be a mixture between two exponentials: one with mean $1/3$ and one with mean $1/7$. The first exponential has twice as much weight as the second one in the mixture. Therefore, the density of X is

$$f(x) = \frac{2}{3}(3e^{-3x}) + \frac{1}{3}(7e^{-7x}) = 2e^{-3x} + \frac{7}{3}e^{-7x}.$$

The following expression generates 10 random variates from this density using `rmixture`.

```
> rmixture(10, probs = c(2, 1),
+          models = expression(rexp(3), rexp(7)))
 [1] 1.29241 0.46852 0.14594 0.37973 0.07673 0.16719
 [7] 0.46209 0.23907 0.03211 0.05700
```

□

See also example 3 for a more involved application combining simulation from a mixture and simulation from a compound Poisson model.

3   Simulation from compound models

Actuaries often need to simulate separately the frequency and the severity of claims for compound models of the form

$$S = C_1 + \cdots + C_N, \tag{2}$$

where $C_1, C_2, \ldots$ are the mutually independent and identically distributed random variables of the claim amounts, each independent of the frequency random variable N.

Function `rcompound` generates variates from the random variable S when the distribution of both random variables N and C is non hierarchical; for the

more general hierarchical case, see section 4. The function has three arguments:

1. `n` the number of variates to generate;

2. `model.freq` the frequency model (random variable N);

3. `model.sev` the severity model (random variable C).

Arguments `model.freq` and `model.sev` are simple R expressions consisting of calls to a random number generation function with the number of variates omitted. This is of course similar to argument `models` of `rmixture`, only with a slightly simpler syntax since one does not need to wrap the calls in `expression`.

Function `rcomppois` is a simplified interface for the common case where N has a Poisson distribution and, therefore, S is compound Poisson. In this function, argument `model.freq` is replaced by `lambda` that takes the value of the Poisson parameter.

Example 2. Let $S \sim$ Compound Poisson$(1.5, F)$, where 1.5 is the value of the Poisson parameter and F is the cumulative distribution function of a gamma distribution with shape parameter $\alpha = 3$ and rate parameter $\lambda = 2$. We obtain variates from the random variable S using `rcompound` or `rcomppois` as follows:

```
> rcompound(10, rpois(1.5), rgamma(3, 2))
 [1] 0.000 0.000 0.000 3.967 6.698 0.000 0.000 0.000
 [9] 6.388 1.625
> rcomppois(10, 1.5, rgamma(3, 2))
 [1] 2.290 0.000 1.992 2.046 6.749 5.428 1.510 0.000
 [9] 2.365 0.000
```

Specifying argument `SIMPLIFY = FALSE` to either function will return not only the variates from S, but also the underlying variates from the random variables N and $C_1, \ldots, C_N$:

```
> rcomppois(10, 1.5, rgamma(3, 2), SIMPLIFY = FALSE)
$aggregate
 [1] 0.0000 4.4194 0.0000 3.9928 1.6938 1.2349
 [7] 5.9995 0.0000 1.9216 0.7615

$frequency
 [1] 0 2 0 2 2 1 3 0 1 1

$severity
 [1] 2.2554 2.1640 0.6178 3.3750 0.8121 0.8817
 [7] 1.2349 1.7885 3.1585 1.0524 1.9216 0.7615
```

□

Example 3. Theorem 9.7 of Klugman et al. (2012) states that the sum of compound Poisson random variables is itself compound Poisson with Poisson parameter equal to the sum of the Poisson parameters and severity distribution equal to the mixture of the severity models.

Let $S = S_1 + S_2 + S_3$, where S_1 is compound Poisson with mean frequency $\lambda = 2$ and severity Gamma$(3, 1)$; S_2 is compound Poisson with $\lambda = 1$ and severity Gamma$(5, 4)$; S_3 is compound Poisson with $\lambda = 1/2$ and severity Lognormal$(2, 1)$. By the aforementioned theorem, S is compound Poisson with $\lambda = 2 + 1 + 1/2 = 7/2$ and severity density

$$f(x) = \frac{4}{7}\left(\frac{1}{\Gamma(3)}x^2 e^{-x}\right) + \frac{2}{7}\left(\frac{4^5}{\Gamma(5)}x^4 e^{-4x}\right) + \frac{1}{7}\phi(\ln x - 2).$$

Combining `rcomppois` and `rmixture` we can generate variates of S using the following elegant expression.

```
> x <- rcomppois(1e5, 3.5,
+                rmixture(probs = c(2, 1, 0.5),
+                         expression(rgamma(3),
+                                    rgamma(5, 4),
+                                    rlnorm(2, 1))))
```

One can verify that the theoretical mean of S is $6 + 5/4 + (e^{5/2})/2 = 13.34$. Now, the empirical mean based on the above sample of size 10^5 is:

```
> mean(x)
[1] 13.4
```

□

4 Simulation from compound hierarchical models

Hierarchical probability models are widely used for data classified in a tree-like structure and in Bayesian inference. The main characteristic of such models is to have the probability law at some level in the classification structure be conditional on the outcome in previous levels. For example, adopting a bottom to top description of the model, a simple hierarchical model could be written as

$$\begin{aligned}
X_t | \Lambda, \Theta &\sim \text{Poisson}(\Lambda) \\
\Lambda | \Theta &\sim \text{Gamma}(3, \Theta) \\
\Theta &\sim \text{Gamma}(2, 2),
\end{aligned} \tag{3}$$

where X_t represents actual data. The random variables Θ and Λ are generally seen as uncertainty, or risk, parameters in the actuarial literature; in the sequel, we refer to them as mixing parameters.

The example above is merely a multi-level mixture of models, something that is simple to simulate "by hand". The following R expression will yield n variates of the random variable X_t:

```
> rpois(n, rgamma(n, 3, rgamma(n, 2, 2)))
```

However, for categorical data common in actuarial applications there will usually be many categories — or *nodes* — at each level. Simulation is then complicated by the need to always use the correct parameters for each variate. Furthermore, one may need to simulate both the frequency and the severity of claims for compound models of the form (2).

This section briefly describes function `rcomphierarc` and its usage. Goulet and Pouliot (2008) discuss in more details the models supported by the function and give more thorough examples.

4.1 Description of hierarchical models

We consider simulation of data from hierarchical models. We want a method to describe these models in R that meets the following criteria:

1. simple and intuitive to go from the mathematical formulation of the model to the R formulation and back;

2. allows for any number of levels and nodes;

3. at any level, allows for any use of parameters higher in the hierarchical structure.

A hierarchical model is completely specified by the number of nodes at each level and by the probability laws at each level. The number of nodes is passed to `rcomphierarc` by means of a named list where each element is a vector of the number of nodes at a given level. Vectors are recycled when the number of nodes is the same throughout a level.

Probability models are expressed in a semi-symbolic fashion using an object of mode "expression". Each element of the object must be named — with names matching those of the number of nodes list — and should be a complete call to an existing random number generation function, but with the number of variates omitted. Hierarchical models are achieved by replacing one or more parameters of a distribution at a given level by any combination of the names of the levels above. If no mixing is to take place at a level, the model for this level can be `NULL`.

Example 4. Consider the following expanded version of model (3):

$$
\begin{aligned}
X_{ijt}|\Lambda_{ij}, \Theta_i &\sim \text{Poisson}(\Lambda_{ij}), & t &= 1, \ldots, n_{ij} \\
\Lambda_{ij}|\Theta_i &\sim \text{Gamma}(3, \Theta_i), & j &= 1, \ldots, J_i \\
\Theta_i &\sim \text{Gamma}(2, 2), & i &= 1, \ldots, I,
\end{aligned}
$$

with $I = 3$, $J_1 = 4$, $J_2 = 5$, $J_3 = 6$ and $n_{ij} \equiv n = 10$. Then the number of nodes and the probability model are respectively specified by the following expressions.

5

```
list(Theta = 3, Lambda = c(4, 5, 6), Data = 10)
```

```
expression(Theta = rgamma(2, 2),
           Lambda = rgamma(3, Theta),
           Data = rpois(Lambda))
```

□

Storing the probability model requires an expression object in order to avoid evaluation of the incomplete calls to the random number generation functions. Function rcomphierarc builds and executes the calls to the random generation functions from the top of the hierarchical model to the bottom. At each level, the function

1. infers the number of variates to generate from the number of nodes list, and

2. appropriately recycles the mixing parameters simulated previously.

The actual names in the list and the expression object can be anything; they merely serve to identify the mixing parameters. Furthermore, any random generation function can be used. The only constraint is that the name of the number of variates argument is n.

In addition, rcomphierarc supports usage of weights in models. These usually modify the frequency parameters to take into account the "size" of an entity. Weights are used in simulation wherever the name weights appears in a model.

4.2 Usage of `rcomphierarc`

Function rcomphierarc can simulate data for structures where both the frequency model and the severity model are hierarchical. It has four main arguments:

1. nodes for the number of nodes list;

2. model.freq for the frequency model;

3. model.sev for the severity model;

4. weights for the vector of weights in lexicographic order, that is all weights of entity 1, then all weights of entity 2, and so on.

The function returns the variates in a list of class "portfolio" with a dim attribute of length two. The list contains all the individual claim amounts for each entity. Since every element can be a vector, the object can be seen as a three-dimension array with a third dimension of potentially varying length. The function also returns a matrix of integers giving the classification indexes of each entity in the portfolio.

The package also defines methods for four generic functions to easily access key quantities for each entity of the simulated portfolio:

1. a method of `aggregate` to compute the aggregate claim amounts S;

2. a method of `frequency` to compute the number of claims N;

3. a method of `severity` (a generic function introduced by the package) to return the individual claim amounts C_j;

4. a method of `weights` to extract the weights matrix.

In addition, all methods have a `classification` and a `prefix` argument. When the first is `FALSE`, the classification index columns are omitted from the result. The second argument overrides the default column name prefix; see the `rcomphierarc.summaries` help page for details.

The following example illustrates these concepts in detail.

Example 5. Consider the following compound hierarchical model:

$$S_{ijt} = C_{ijt1} + \cdots + C_{ijtN_{ijt}},$$

for $i = 1, \ldots, I$, $j = 1, \ldots, J_i$, $t = 1, \ldots, n_{ij}$ and with

$$N_{ijt}|\Lambda_{ij}, \Phi_i \sim \text{Poisson}(w_{ijt}\Lambda_{ij}) \qquad C_{ijtu}|\Theta_{ij}, \Psi_i \sim \text{Lognormal}(\Theta_{ij}, 1)$$
$$\Lambda_{ij}|\Phi_i \sim \text{Gamma}(\Phi_i, 1) \qquad\qquad \Theta_{ij}|\Psi_i \sim N(\Psi_i, 1)$$
$$\Phi_i \sim \text{Exponential}(2) \qquad\qquad \Psi_i \sim N(2, 0.1).$$

(Note how weights modify the Poisson parameter.) Using as convention to number the data level 0, the above is a two-level compound hierarchical model.

Assuming that $I = 2$, $J_1 = 4$, $J_2 = 3$, $n_{11} = \cdots = n_{14} = 4$ and $n_{21} = n_{22} = n_{23} = 5$ and that weights are simply simulated from a uniform distribution on $(0.5, 2.5)$, then simulation of a data set with `rcomphierarc` is achieved with the following expressions.

```
> nodes <- list(cohort = 2,
+               contract = c(4, 3),
+               year = c(4, 4, 4, 4, 5, 5, 5))
> mf <- expression(cohort = rexp(2),
+                  contract = rgamma(cohort, 1),
+                  year = rpois(weights * contract))
> ms <- expression(cohort = rnorm(2, sqrt(0.1)),
+                  contract = rnorm(cohort, 1),
+                  year = rlnorm(contract, 1))
> wijt <- runif(31, 0.5, 2.5)
> pf <- rcomphierarc(nodes = nodes,
+                    model.freq = mf, model.sev = ms,
+                    weights = wijt)
```

Object `pf` is a list of class `"portfolio"` containing, among other things, the aforementioned two-dimension list as element `data` and the classification matrix (subscripts i and j) as element `classification`:

7

```
> class(pf)
[1] "portfolio"
> pf$data
      year.1    year.2    year.3    year.4
[1,] Numeric,2 Numeric,2 11.38     Numeric,0
[2,] Numeric,0 Numeric,0 Numeric,0 Numeric,0
[3,] Numeric,0 Numeric,3 Numeric,0 Numeric,2
[4,] Numeric,0 98.13     50.62     55.7
[5,] Numeric,0 11.79     2.253     2.397
[6,] Numeric,0 Numeric,0 Numeric,0 Numeric,0
[7,] Numeric,3 Numeric,4 Numeric,2 Numeric,2
      year.5
[1,] NA
[2,] NA
[3,] NA
[4,] NA
[5,] Numeric,2
[6,] Numeric,0
[7,] Numeric,0
> pf$classification
      cohort contract
[1,]       1        1
[2,]       1        2
[3,]       1        3
[4,]       1        4
[5,]       2        1
[6,]       2        2
[7,]       2        3
```

The output of pf$data is not much readable. If we were to print the results of rcomphierarc this way, many users would wonder what Numeric,n means. (It is actually R's way to specify that a given element in the list is a numeric vector of length n — the third dimension mentioned above.) To ease reading, the print method for objects of class "portfolio" only prints the simulation model and the number of claims in each node:

```
> pf
Portfolio of claim amounts

  Frequency model
    cohort   ~ rexp(2)
    contract ~ rgamma(cohort, 1)
    year     ~ rpois(weights * contract)
  Severity model
    cohort   ~ rnorm(2, sqrt(0.1))
```

```
    contract ~ rnorm(cohort, 1)
    year     ~ rlnorm(contract, 1)

  Number of claims per node:

      cohort contract year.1 year.2 year.3 year.4
[1,]       1        1      2      2      1      0
[2,]       1        2      0      0      0      0
[3,]       1        3      0      3      0      2
[4,]       1        4      0      1      1      1
[5,]       2        1      0      1      1      1
[6,]       2        2      0      0      0      0
[7,]       2        3      3      4      2      2
      year.5
[1,]      NA
[2,]      NA
[3,]      NA
[4,]      NA
[5,]       2
[6,]       0
[7,]       0
```

By default, the method of aggregate returns the values of S_{ijt} in a regular matrix (subscripts i and j in the rows, subscript t in the columns). The method has a by argument to get statistics for other groupings and a FUN argument to get statistics other than the sum:

```
> aggregate(pf)
      cohort contract year.1 year.2 year.3 year.4
[1,]       1        1  31.37  7.521 11.383  0.000
[2,]       1        2   0.00  0.000  0.000  0.000
[3,]       1        3   0.00 72.706  0.000 23.981
[4,]       1        4   0.00 98.130 50.622 55.705
[5,]       2        1   0.00 11.793  2.253  2.397
[6,]       2        2   0.00  0.000  0.000  0.000
[7,]       2        3  44.81 88.737 57.593 14.589
      year.5
[1,]      NA
[2,]      NA
[3,]      NA
[4,]      NA
[5,]   10.48
[6,]    0.00
[7,]    0.00
> aggregate(pf, by = c("cohort", "year"), FUN = mean)
      cohort year.1 year.2 year.3 year.4 year.5
```

```
[1,]      1  15.69  29.73  31.00 26.562      NA
[2,]      2  14.94  20.11  19.95  5.662  5.238
```

The method of `frequency` returns the values of N_{ijt}. It is mostly a wrapper for the `aggregate` method with the default sum statistic replaced by `length`. Hence, arguments `by` and `FUN` remain available:

```
> frequency(pf)
      cohort contract year.1 year.2 year.3 year.4
[1,]       1        1      2      2      1      0
[2,]       1        2      0      0      0      0
[3,]       1        3      0      3      0      2
[4,]       1        4      0      1      1      1
[5,]       2        1      0      1      1      1
[6,]       2        2      0      0      0      0
[7,]       2        3      3      4      2      2
      year.5
[1,]      NA
[2,]      NA
[3,]      NA
[4,]      NA
[5,]       2
[6,]       0
[7,]       0
> frequency(pf, by = "cohort")
      cohort freq
[1,]       1   13
[2,]       2   16
```

The method of `severity` returns the individual variates C_{ijtu} in a matrix similar to those above, but with a number of columns equal to the maximum number of observations per entity,

$$\max_{i,j} \sum_{t=1}^{n_{ij}} N_{ijt}.$$

Thus, the original period of observation (subscript t) and the identifier of the severity within the period (subscript u) are lost and each variate now constitute a "period" of observation. For this reason, the method provides an argument `splitcol` in case one would like to extract separately the individual severities of one or more periods:

```
> severity(pf)
$main
      cohort contract claim.1 claim.2 claim.3
[1,]       1        1   7.974  23.401   3.153
[2,]       1        2      NA      NA      NA
```

10

```
[3,]        1          3   3.817  41.979  26.910
[4,]        1          4  98.130  50.622  55.705
[5,]        2          1  11.793   2.253   2.397
[6,]        2          2      NA      NA      NA
[7,]        2          3  14.322  11.522  18.966
      claim.4 claim.5 claim.6 claim.7 claim.8
[1,]   4.368  11.383      NA      NA      NA
[2,]      NA      NA      NA      NA      NA
[3,]   4.903  19.078      NA      NA      NA
[4,]      NA      NA      NA      NA      NA
[5,]   9.472   1.004      NA      NA      NA
[6,]      NA      NA      NA      NA      NA
[7,]  33.108  15.532   14.99   25.11   40.15
      claim.9 claim.10 claim.11
[1,]      NA      NA       NA
[2,]      NA      NA       NA
[3,]      NA      NA       NA
[4,]      NA      NA       NA
[5,]      NA      NA       NA
[6,]      NA      NA       NA
[7,]   17.44   4.426    10.16


$split
NULL
> severity(pf, splitcol = 1)
$main
      cohort contract claim.1 claim.2 claim.3
[1,]        1          1   3.153   4.368  11.383
[2,]        1          2      NA      NA      NA
[3,]        1          3   3.817  41.979  26.910
[4,]        1          4  98.130  50.622  55.705
[5,]        2          1  11.793   2.253   2.397
[6,]        2          2      NA      NA      NA
[7,]        2          3  33.108  15.532  14.990
      claim.4 claim.5 claim.6 claim.7 claim.8
[1,]      NA      NA      NA      NA      NA
[2,]      NA      NA      NA      NA      NA
[3,]   4.903  19.078      NA      NA      NA
[4,]      NA      NA      NA      NA      NA
[5,]   9.472   1.004      NA      NA      NA
[6,]      NA      NA      NA      NA      NA
[7,]  25.107  40.150   17.44   4.426   10.16


$split
      cohort contract claim.1 claim.2 claim.3
```

```
[1,]      1         1     7.974     23.40       NA
[2,]      1         2        NA        NA       NA
[3,]      1         3        NA        NA       NA
[4,]      1         4        NA        NA       NA
[5,]      2         1        NA        NA       NA
[6,]      2         2        NA        NA       NA
[7,]      2         3    14.322     11.52    18.97
```

Finally, the weights matrix corresponding to the data in object pf is

```
> weights(pf)
      cohort contract year.1 year.2 year.3 year.4
[1,]       1        1 0.8361  2.115 1.2699 1.1555
[2,]       1        2 1.7042  1.709 0.7493 1.0892
[3,]       1        3 1.6552  1.762 1.5240 1.5100
[4,]       1        4 1.5681  1.614 2.2358 2.1594
[5,]       2        1 0.7229  1.907 2.2950 1.0595
[6,]       2        2 0.5307  0.758 0.6868 0.9738
[7,]       2        3 1.6995  2.320 1.6208 2.0114
      year.5
[1,]      NA
[2,]      NA
[3,]      NA
[4,]      NA
[5,] 0.9564
[6,] 2.0823
[7,] 1.2583
```

Combined with the argument classification = FALSE, the above methods can be used to easily compute loss ratios:

```
> aggregate(pf, classif = FALSE) / weights(pf, classif = FALSE)
      year.1 year.2  year.3 year.4 year.5
[1,]   37.53  3.556  8.9638  0.000     NA
[2,]    0.00  0.000  0.0000  0.000     NA
[3,]    0.00 41.264  0.0000 15.881     NA
[4,]    0.00 60.781 22.6412 25.796     NA
[5,]    0.00  6.183  0.9818  2.263  10.95
[6,]    0.00  0.000  0.0000  0.000   0.00
[7,]   26.37 38.244 35.5328  7.253   0.00
```

□

Example 6. Scollnik (2001) considers the following model for the simulation

of claims frequency data in a Markov Chain Monte Carlo (MCMC) context:

$$S_{it}|\Lambda_i, \alpha, \beta \sim \text{Poisson}(w_{ij}\Lambda_i)$$
$$\Lambda_i|\alpha, \beta \sim \text{Gamma}(\alpha, \beta)$$
$$\alpha \sim \text{Gamma}(5, 5)$$
$$\beta \sim \text{Gamma}(25, 1)$$

for $i = 1, 2, 3$, $j = 1, \ldots, 5$ and with weights w_{it} simulated from

$$w_{it}|a_i, b_i \sim \text{Gamma}(a_i, b_i)$$
$$a_i \sim U(0, 100)$$
$$b_i \sim U(0, 100).$$

Strictly speaking, this is not a hierarchical model since the random variables α and β are parallel rather than nested. Nevertheless, with some minor manual intervention, function rcomphierarc can simulate data from this model.

First, one simulates the weights (in lexicographic order) with

```
> wit <- rgamma(15, rep(runif(3, 0, 100), each = 5),
+               rep(runif(3, 0, 100), each = 5))
```

Second, one calls rcomphierarc to simulate the frequency data. The key here consists in manually inserting the simulation of the shape and rate parameters of the gamma distribution in the model for Λ_i. Finally, wrapping the call to rcomphierarc in frequency will immediately yield the matrix of observations:

```
> frequency(rcomphierarc(list(entity = 3, year = 5),
+           expression(entity = rgamma(rgamma(1, 5, 5),
+                                      rgamma(1, 25, 1)),
+                      year = rpois(weights * entity)),
+           weights = wit))
     entity year.1 year.2 year.3 year.4 year.5
[1,]      1      0      0      0      0      0
[2,]      2      0      0      0      0      0
[3,]      3      0      1      0      1      1
```

□

One will find more examples of rcomphierarc usage in the simulation demo file. The function was used to simulate the data in Forgues et al. (2006).

References

A. Forgues, V. Goulet, and J. Lu. Credibility for severity revisited. *North American Actuarial Journal*, 10(1):49–62, 2006.

V. Goulet and L.-P. Pouliot. Simulation of compound hierarchical models in R. *North American Actuarial Journal*, 12:401–412, 2008.

S. A. Klugman, H. H. Panjer, and G. Willmot. *Loss Models: From Data to Decisions*. Wiley, New York, 4 edition, 2012. ISBN 978-1-118-31532-3.

D. P. M. Scollnik. Actuarial modeling with MCMC and BUGS. *North American Actuarial Journal*, 5(2):96–124, 2001.