actuar: An R Package for Actuarial Science

Vincent Goulet, Christophe Dutang, Mathieu Pigeon

Journal of Statistical Software

March 2008, Volume 25, Issue 7. *http://www.jstatsoft.org/*

actuar: An R Package for Actuarial Science

Christophe Dutang
Université Claude Bernard Lyon 1

Vincent Goulet
Université Laval

Mathieu Pigeon
Université Laval

Abstract

actuar is a package providing additional Actuarial Science functionality to the R statistical system. The project was launched in 2005 and the package is available on the Comprehensive R Archive Network since February 2006. The current version of the package contains functions for use in the fields of loss distributions modeling, risk theory (including ruin theory), simulation of compound hierarchical models and credibility theory. This paper presents in detail but with few technical terms the most recent version of the package.

Keywords: insurance, loss distributions, risk theory, ruin theory, credibility theory, distributions, phase-type, matrix exponential, simulation, hierarchical, R.

1. Introduction

actuar is a package providing additional Actuarial Science functionality to the R statistical system (R Development Core Team 2008). Various packages on the Comprehensive R Archive Network (CRAN, `http://CRAN.R-project.org/`) provide functions useful to actuaries, e.g., **copula** (Yan 2007), **Rmetrics** (Wuertz 2007), **SuppDists** (Wheeler 2008) or the R recommended packages **nlme** (Pinheiro, Bates, DebRoy, Sarkar, and the R Development Core Team 2007) and **survival** (Lumley 2008) just to name a few. However, **actuar** aims to serve as a central location for more specifically actuarial functions and data sets. The project was officially launched in 2005 and is under active development.

The feature set of the package can be split in four main categories: loss distributions modeling, risk theory (including ruin theory), simulation of compound hierarchical models and credibility theory. As much as possible, the developers have tried to keep the "user interface" of the various functions of the package consistent. Moreover, the package follows the general R philosophy of working with model objects.

This paper reviews the various features of version 0.9-7 of **actuar**. We provide enough actuarial

background where needed to make the paper self-contained, but otherwise give numerous references for the reader interested to dive more into the subject.

Future versions of the package can be obtained from CRAN at `http://CRAN.R-project.org/package=actuar`.

2. Loss distributions modeling

One important task of actuaries is the modeling of claim amount distributions for ratemaking, loss reserving or other risk evaluation purposes. Package **actuar** offers many functions for loss distributions modeling. The present section details the following **actuar** features:

1. introduction of 18 additional probability laws and utility functions to get raw moments, limited moments and the moment generating function;
2. fairly extensive support of grouped data;
3. calculation of the empirical raw and limited moments;
4. minimum distance estimation using three different measures;
5. treatment of coverage modifications (deductibles, limits, inflation, coinsurance).

2.1. Probability laws

R already includes functions to compute the probability density function (pdf), the cumulative distribution function (cdf) and the quantile function of a fair number of probability laws, as well as functions to generate variates from these laws. For some root *foo*, the utility functions are named `d`*foo*, `p`*foo*, `q`*foo* and `r`*foo*, respectively.

The **actuar** package provides `d`, `p`, `q` and `r` functions for all the probability laws useful for loss severity modeling found in Appendix A of Klugman, Panjer, and Willmot (2004) and not already present in base R, excluding the inverse Gaussian and log-t but including the loggamma distribution (Hogg and Klugman 1984). Table 1 lists the supported distributions as named in Klugman *et al.* (2004) along with the root names of the R functions.

In addition to the `d`, `p`, `q` and `r` functions, the package provides `m`, `lev` and `mgf` functions to compute, respectively, theoretical raw moments

$$m_k = \mathsf{E}[X^k], \tag{1}$$

theoretical limited moments

$$\mathsf{E}[(X \wedge x)^k] = \mathsf{E}[\min(X, x)^k] \tag{2}$$

and the moment generating function

$$M_X(t) = \mathsf{E}[e^{tX}], \tag{3}$$

when it exists. Every probability law of Table 1 is supported, plus the following ones: beta, exponential, chi-square, gamma, lognormal, normal (no `lev`), uniform and Weibull of base R

Family	Distribution	Root
Transformed beta	Transformed beta	`trbeta`
	Burr	`burr`
	Loglogistic	`llogis`
	Paralogistic	`paralogis`
	Generalized Pareto	`genpareto`
	Pareto	`pareto`
	Inverse Burr	`invburr`
	Inverse Pareto	`invpareto`
	Inverse paralogistic	`invparalogis`
Transformed gamma	Transformed gamma	`trgamma`
	Inverse transformed gamma	`invtrgamma`
	Inverse gamma	`invgamma`
	Inverse Weibull	`invweibull`
	Inverse exponential	`invexp`
Other	Loggamma	`lgamma`
	Single parameter Pareto	`pareto1`
	Generalized beta	`genbeta`

Table 1: Probability laws supported by **actuar** classified by family and root names of the R functions.

and the inverse Gaussian distribution of the package **SuppDists** (Wheeler 2008). The `m` and `lev` functions are especially useful with estimation methods based on the matching of raw or limited moments; see Section 2.4 for their empirical counterparts. The `mgf` functions come in handy to compute the adjustment coefficient in ruin theory; see Section 3.5.

In addition to the 17 distributions of Table 1, the package provides support for a family of distributions deserving a separate presentation. Phase-type distributions (Neuts 1981) are defined as the distribution of the time until absorption of continuous time, finite state Markov processes with m transient states and one absorbing state. Let

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{T} & \boldsymbol{t} \\ \boldsymbol{0} & 0 \end{bmatrix} \tag{4}$$

be the transition rates (or intensity) matrix of such a process and let $(\boldsymbol{\pi}, \pi_{m+1})$ be the initial probability vector. Here, $\boldsymbol{T}$ is an $m \times m$ non-singular matrix with $t_{ii} < 0$ for $i = 1, \ldots, m$ and $t_{ij} \geq 0$ for $i \neq j$, $\boldsymbol{t} = -\boldsymbol{T}\boldsymbol{e}$ and $\boldsymbol{e}$ is a column vector with all components equal to 1. Then the cdf of the time until absorption random variable with parameters $\boldsymbol{\pi}$ and $\boldsymbol{T}$ is

$$F(x) = \begin{cases} \pi_{m+1}, & x = 0 \\ 1 - \boldsymbol{\pi} e^{\boldsymbol{T}x} \boldsymbol{e}, & x > 0, \end{cases} \tag{5}$$

where

$$e^{\boldsymbol{M}} = \sum_{n=0}^{\infty} \frac{\boldsymbol{M}^n}{n!} \tag{6}$$

is the matrix exponential of matrix $\boldsymbol{M}$.

The exponential, the Erlang (gamma with integer shape parameter) and discrete mixtures thereof are common special cases of phase-type distributions.

The package provides functions `{d,p,r,m,mgf}phtype` for phase-type distributions. Function `pphtype` is central to the evaluation of ruin probabilities; see Section 3.6.

The core of all the functions presented in this subsection is written in C for speed. The matrix exponential C routine is based on `expm()` from the package **Matrix** (Bates and Maechler 2008).

2.2. Grouped data

What is commonly referred to in Actuarial Science as grouped data is data represented in an interval-frequency manner. In insurance applications, a grouped data set will typically report that there were n_j claims in the interval $(c_{j-1}, c_j]$, $j = 1, \ldots, r$ (with the possibility that $c_r = \infty$). This representation is much more compact than an individual data set — where the value of each claim is known — but it also carries far less information. Now that storage space in computers has almost become a non issue, grouped data has somewhat fallen out of fashion.

Still, grouped data remains in use in some fields of actuarial practice and also of interest in teaching. For this reason, **actuar** provides facilities to store, manipulate and summarize grouped data. A standard storage method is needed since there are many ways to represent grouped data in the computer: using a list or a matrix, aligning the n_js with the c_{j-1}s or with the c_js, omitting c_0 or not, etc. Moreover, with appropriate extraction, replacement and summary methods, manipulation of grouped data becomes similar to that of individual data.

First, function `grouped.data` creates a grouped data object similar to — and inheriting from — a data frame. The input of the function is a vector of group boundaries $c_0, c_1, \ldots, c_r$ and one or more vectors of group frequencies $n_1, \ldots, n_r$. Note that there should be one group boundary more than group frequencies. Furthermore, the function assumes that the intervals are contiguous. For example, the following data

Group	Frequency (Line 1)	Frequency (Line 2)
$(0, 25]$	30	26
$(25, 50]$	31	33
$(50, 100]$	57	31
$(100, 150]$	42	19
$(150, 250]$	65	16
$(250, 500]$	84	11

is entered and represented in R as

```
R> x <- grouped.data(Group = c(0, 25, 50, 100, 150, 250,
+     500), Line.1 = c(30, 31, 57, 42, 65, 84), Line.2 = c(26,
+     33, 31, 19, 16, 11))
```

Object `x` is stored internally as a list with class

```
R> class(x)

[1] "grouped.data" "data.frame"
```

With a suitable `print` method, these objects can be displayed in an unambiguous manner:

```
R> x

       Group Line.1 Line.2
1   (0,  25]     30     26
2  (25,  50]     31     33
3  (50, 100]     57     31
4 (100, 150]     42     19
5 (150, 250]     65     16
6 (250, 500]     84     11
```

Second, the package supports the most common extraction and replacement methods for `"grouped.data"` objects using the usual `[` and `[<-` operators. In particular, the following extraction operations are supported.

(i) Extraction of the vector of group boundaries (the first column):

```
R> x[, 1]

[1]   0  25  50 100 150 250 500
```

(ii) Extraction of the vector or matrix of group frequencies (the second and third columns):

```
R> x[, -1]

  Line.1 Line.2
1     30     26
2     31     33
3     57     31
4     42     19
5     65     16
6     84     11
```

(iii) Extraction of a subset of the whole object (first three lines):

```
R> x[1:3, ]

      Group Line.1 Line.2
1  (0,  25]     30     26
2 (25,  50]     31     33
3 (50, 100]     57     31
```

Notice how extraction results in a simple vector or matrix if either of the group boundaries or the group frequencies are dropped.

As for replacement operations, the package implements the following.

(i) Replacement of one or more group frequencies:

```
R> x[1, 2] <- 22
R> x

       Group Line.1 Line.2
1   (0,  25]     22     26
2  (25,  50]     31     33
3  (50, 100]     57     31
4 (100, 150]     42     19
5 (150, 250]     65     16
6 (250, 500]     84     11

R> x[1, c(2, 3)] <- c(22, 19)
R> x

       Group Line.1 Line.2
1   (0,  25]     22     19
2  (25,  50]     31     33
3  (50, 100]     57     31
4 (100, 150]     42     19
5 (150, 250]     65     16
6 (250, 500]     84     11
```

(ii) Replacement of the boundaries of one or more groups:

```
R> x[1, 1] <- c(0, 20)
R> x

       Group Line.1 Line.2
1   (0,  20]     22     19
2  (20,  50]     31     33
3  (50, 100]     57     31
4 (100, 150]     42     19
5 (150, 250]     65     16
6 (250, 500]     84     11

R> x[c(3, 4), 1] <- c(55, 110, 160)
R> x

       Group Line.1 Line.2
1   (0,  20]     22     19
2  (20,  55]     31     33
3  (55, 110]     57     31
4 (110, 160]     42     19
5 (160, 250]     65     16
6 (250, 500]     84     11
```

It is not possible to replace the boundaries and the frequencies simultaneously.

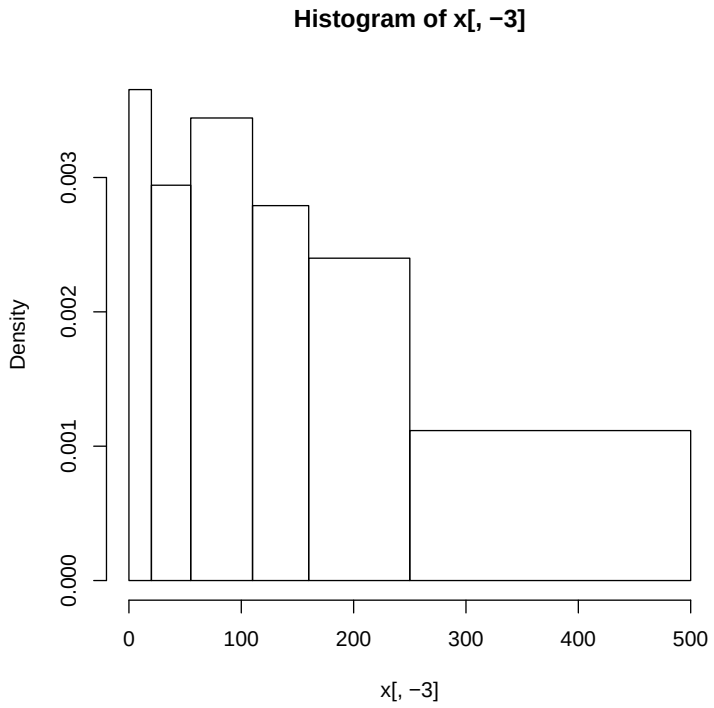

Figure 1: Histogram of a grouped data object

The package defines methods of a few existing summary functions for grouped data objects. Computing the mean

$$\sum_{j=1}^{r} \left(\frac{c_{j-1} + c_j}{2} \right) n_j \tag{7}$$

is made simple with a method for the `mean` function:

```
R> mean(x)

Line.1 Line.2
 188.0  108.2
```

Higher empirical moments can be computed with `emm`; see Section 2.4.

The R function `hist` splits individual data into groups and draws an histogram of the frequency distribution. The package introduces a method for already grouped data. Only the first frequencies column is considered (see Figure 1 for the resulting graph):

```
R> hist(x[, -3])
```

R has a function `ecdf` to compute the empirical cdf of an individual data set,

$$F_n(x) = \frac{1}{n} \sum_{j=1}^{n} I\{x_j \leq x\}, \tag{8}$$

where $I\{\mathcal{A}\} = 1$ if $\mathcal{A}$ is true and $I\{\mathcal{A}\} = 0$ otherwise. The function returns a `"function"` object to compute the value of $F_n(x)$ in any x.

The approximation of the empirical cdf for grouped data is called an ogive (Klugman, Panjer, and Willmot 1998; Hogg and Klugman 1984). It is obtained by joining the known values of $F_n(x)$ at group boundaries with straight line segments:

$$\tilde{F}_n(x) = \begin{cases} 0, & x \leq c_0 \\ \dfrac{(c_j - x)F_n(c_{j-1}) + (x - c_{j-1})F_n(c_j)}{c_j - c_{j-1}}, & c_{j-1} < x \leq c_j \\ 1, & x > c_r. \end{cases} \tag{9}$$

The package includes a function `ogive` that otherwise behaves exactly like `ecdf`. In particular, methods for functions `knots` and `plot` allow, respectively, to obtain the knots $c_0, c_1, \ldots, c_r$ of the ogive and a graph (see Figure 2):

```
R> Fnt <- ogive(x)
R> knots(Fnt)

[1]   0  20  55 110 160 250 500

R> Fnt(knots(Fnt))

[1] 0.00000 0.07309 0.17608 0.36545 0.50498 0.72093 1.00000

R> plot(Fnt)
```

2.3. Data sets

This is certainly not the most spectacular feature of **actuar**, but it remains useful for illustrations and examples: the package includes the individual dental claims and grouped dental claims data of Klugman *et al.* (2004):

```
R> data("dental")
R> dental

 [1]  141   16   46   40  351  259  317 1511  107  567

R> data("gdental")
R> gdental

             cj nj
1      (0,  25] 30
2    ( 25,  50] 31
3    ( 50, 100] 57
4    (100, 150] 42
5    (150, 250] 65
```

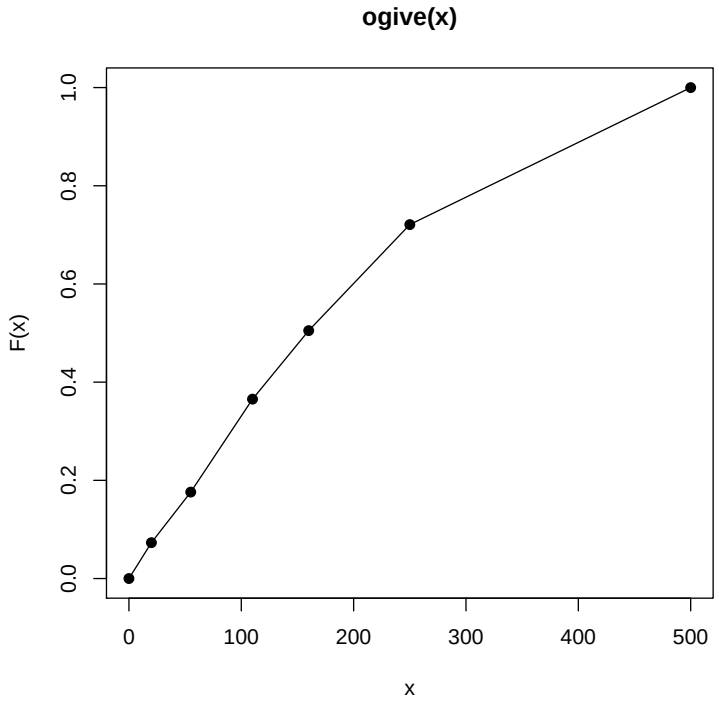

Figure 2: Ogive of a grouped data object

```
6    (250, 500] 84
7   (500, 1000] 45
8  (1000, 1500] 10
9  (1500, 2500] 11
10 (2500, 4000]  3
```

2.4. Calculation of empirical moments

The package provides two functions useful for estimation based on moments. First, function `emm` computes the kth empirical moment of a sample, whether in individual or grouped data form:

```
R> emm(dental, order = 1:3)

[1] 3.355e+02 2.931e+05 3.729e+08

R> emm(gdental, order = 1:3)

[1] 3.533e+02 3.577e+05 6.586e+08
```

Second, in the same spirit as `ecdf` and `ogive`, function `elev` returns a function to compute the empirical limited expected value — or first limited moment — of a sample for any limit. Again, there are methods for individual and grouped data (see Figure 3 for the graphs):

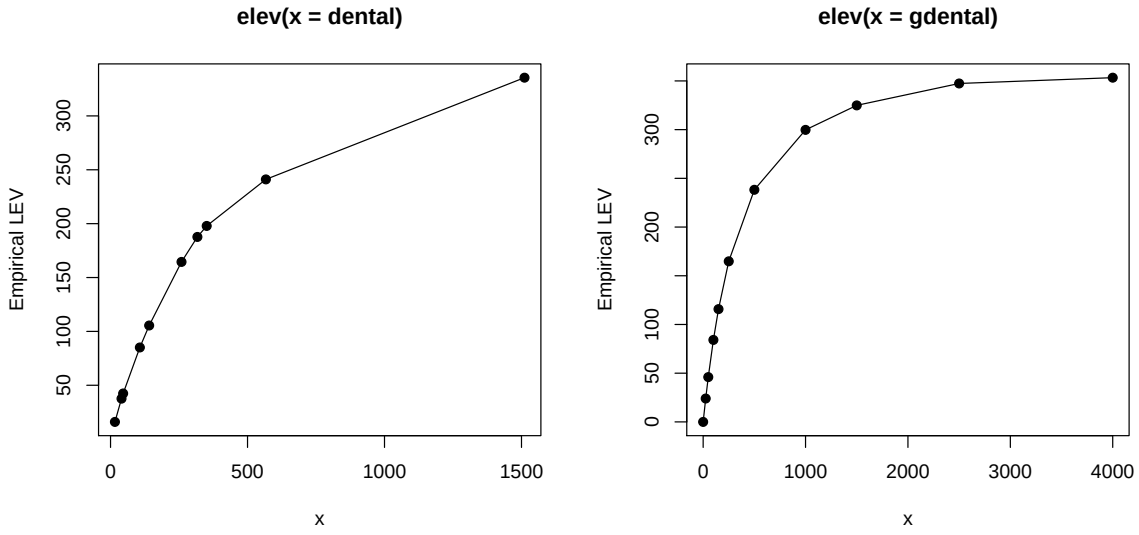

Figure 3: Empirical limited expected value function of an individual data object (left) and a grouped data object (right)

```
R> lev <- elev(dental)
R> lev(knots(lev))

 [1]  16.0  37.6  42.4  85.1 105.5 164.5 187.7 197.9 241.1 335.5

R> plot(lev, type = "o", pch = 19)
R> lev <- elev(gdental)
R> lev(knots(lev))

 [1]   0.00  24.01  46.00  84.16 115.77 164.85 238.26 299.77 324.90
[10] 347.39 353.34

R> plot(lev, type = "o", pch = 19)
```

2.5. Minimum distance estimation

Two methods are widely used by actuaries to fit models to data: maximum likelihood and minimum distance. The first technique applied to individual data is well covered by function `fitdistr` of the package **MASS** (Venables and Ripley 2002). The second technique minimizes a chosen distance function between theoretical and empirical distributions. Package **actuar** provides function `mde`, very similar in usage and inner working to `fitdistr`, to fit models according to any of the following three distance minimization methods.

1. The Cramér-von Mises method (`CvM`) minimizes the squared difference between the theoretical cdf and the empirical cdf or ogive at their knots:

$$d(\theta) = \sum_{j=1}^{n} w_j [F(x_j; \theta) - F_n(x_j; \theta)]^2 \tag{10}$$

for individual data and

$$d(\theta) = \sum_{j=1}^{r} w_j [F(c_j; \theta) - \tilde{F}_n(c_j; \theta)]^2 \tag{11}$$

for grouped data. Here, $F(x)$ is the theoretical cdf of a parametric family, $F_n(x)$ is the empirical cdf, $\tilde{F}_n(x)$ is the ogive and $w_1 \geq 0, w_2 \geq 0, \ldots$ are arbitrary weights (defaulting to 1).

2. The modified chi-square method (`chi-square`) applies to grouped data only and minimizes the squared difference between the expected and observed frequency within each group:

$$d(\theta) = \sum_{j=1}^{r} w_j [n(F(c_j; \theta) - F(c_{j-1}; \theta)) - n_j]^2, \tag{12}$$

where $n = \sum_{j=1}^{r} n_j$. By default, $w_j = n_j^{-1}$.

3. The layer average severity method (`LAS`) applies to grouped data only and minimizes the squared difference between the theoretical and empirical limited expected value within each group:

$$d(\theta) = \sum_{j=1}^{r} w_j [\mathsf{LAS}(c_{j-1}, c_j; \theta) - \tilde{\mathsf{LAS}}_n(c_{j-1}, c_j; \theta)]^2, \tag{13}$$

where $\mathsf{LAS}(x, y) = \mathsf{E}[X \wedge y] - \mathsf{E}[X \wedge x]$, $\tilde{\mathsf{LAS}}_n(x, y) = \tilde{\mathsf{E}}_n[X \wedge y] - \tilde{\mathsf{E}}_n[X \wedge x]$ and $\tilde{\mathsf{E}}_n[X \wedge x]$ is the empirical limited expected value for grouped data.

The arguments of `mde` are a data set, a function to compute $F(x)$ or $\mathsf{E}[X \wedge x]$, starting values for the optimization procedure and the name of the method to use. The empirical functions are computed with `ecdf`, `ogive` or `elev`.

The expressions below fit an exponential distribution to the grouped dental data set, as per Example 2.21 of Klugman *et al.* (1998):

```
R> mde(gdental, pexp, start = list(rate = 1/200), measure = "CvM")

   rate
 0.003551

 distance
 0.002842

R> mde(gdental, pexp, start = list(rate = 1/200), measure = "chi-square")

   rate
 0.00364

 distance
    13.54
```

Coverage modification	Per-loss variable (Y^L)	Per-payment variable (Y^P)
Ordinary deductible (d)	$\begin{cases} 0, & X \leq d \\ X - d, & X > d \end{cases}$	$\begin{cases} X - d, & X > d \end{cases}$
Franchise deductible (d)	$\begin{cases} 0, & X \leq d \\ X, & X > d \end{cases}$	$\begin{cases} X, & X > d \end{cases}$
Limit (u)	$\begin{cases} X, & X \leq u \\ u, & X > u \end{cases}$	$\begin{cases} X, & X \leq u \\ u, & X > u \end{cases}$
Coinsurance (α)	αX	αX
Inflation (r)	$(1 + r)X$	$(1 + r)X$

Table 2: Coverage modifications for the per-loss and per-payment variables as defined in Klugman *et al.* (2004).

```
R> mde(gdental, levexp, start = list(rate = 1/200), measure = "LAS")

   rate
 0.002966

 distance
    694.5
```

2.6. Coverage modifications

Let X be the random variable of the actual claim amount for an insurance policy, Y^L be the random variable of the amount paid per loss and Y^P be the random variable of the amount paid per payment. The terminology for the last two random variables refers to whether or not the insurer knows that a loss occurred. Now, the random variables X, Y^L and Y^P will differ if any of the following coverage modifications are present for the policy: an ordinary or a franchise deductible, a limit, coinsurance or inflation adjustment (see Klugman *et al.* 2004, Chapter 5 for precise definitions of these terms). Table 2 summarizes the definitions of Y^L and Y^P.

Often, one will want to use data $Y_1^L, \ldots, Y_n^L$ (or $Y_1^P, \ldots, Y_n^P$) from the random variable Y^L (Y^P) to fit a model on the unobservable random variable X. This requires expressing the pdf or cdf of Y^L (Y^P) in terms of the pdf or cdf of X. Function `coverage` of **actuar** does just that: given a pdf or cdf and any combination of the coverage modifications mentioned above, `coverage` returns a function object to compute the pdf or cdf of the modified random variable. The function can then be used in modeling like any other `dfoo` or `pfoo` function.

For example, let Y^P represent the amount paid by an insurer for a policy with an ordinary deductible d and a limit $u - d$ (or maximum covered loss of u). Then the definition of Y^P is

$$Y^P = \begin{cases} X - d, & d \leq X \leq u \\ u - d, & X \geq u \end{cases} \tag{14}$$

and its pdf is

$$f_{Y^P}(y) = \begin{cases} 0, & y = 0 \\ \dfrac{f_X(y+d)}{1-F_X(d)}, & 0 < y < u-d \\ \dfrac{1-F_X(u)}{1-F_X(d)}, & y = u-d \\ 0, & y > u-d. \end{cases} \tag{15}$$

Assume X has a gamma distribution. Then an R function to compute the pdf (15) in any y for a deductible $d = 1$ and a limit $u = 10$ is obtained with `coverage` as follows:

```
R> f <- coverage(pdf = dgamma, cdf = pgamma, deductible = 1,
+     limit = 10)
R> f

function (x, shape, rate = 1, scale = 1/rate)
ifelse(x == 0, 0, ifelse(0 < x & x < 9, do.call("dgamma", list(
    x + 1, shape = shape, rate = rate, scale = scale))/do.call("pgamma",
    list(1, shape = shape, rate = rate, scale = scale, lower.tail = FALSE)),
    ifelse(x == 9, do.call("pgamma", list(10, shape = shape,
        rate = rate, scale = scale, lower.tail = FALSE))/do.call("pgamma",
        list(1, shape = shape, rate = rate, scale = scale,
        lower.tail = FALSE)), 0)))
<environment: 0x1bab75c>

R> f(0, shape = 5, rate = 1)

[1] 0

R> f(5, shape = 5, rate = 1)

[1] 0.1343

R> f(9, shape = 5, rate = 1)

[1] 0.02936

R> f(12, shape = 5, rate = 1)

[1] 0
```

Note how function `f` is built specifically for the coverage modifications submitted and contains as little useless code as possible. For comparison purpose, the following function contains no deductible and no limit:

```
R> g <- coverage(dgamma, pgamma)
R> g
```

```
function (x, shape, rate = 1, scale = 1/rate)
ifelse(x == 0, 0, ifelse(0 < x & x < Inf, do.call("dgamma", list(
    x, shape = shape, rate = rate, scale = scale)), ifelse(x ==
    Inf, 0, 0)))
<environment: 0x15b8b64>
```

The vignette `"coverage"` contains more detailed pdf and cdf formulas under various combinations of coverage modifications.

3. Risk theory

Risk theory refers to a body of techniques to model and measure the risk associated with a portfolio of insurance contracts. A first approach consists in modeling the distribution of total claims over a fixed period of time using the classical collective model of risk theory. A second input of interest to the actuary is the evolution of the surplus of the insurance company over many periods of time. In *ruin theory*, the main quantity of interest is the probability that the surplus becomes negative, in which case technical ruin of the insurance company occurs.

The interested reader can find more on these subjects in Klugman *et al.* (2004); Gerber (1979); Denuit and Charpentier (2004); Kaas, Goovaerts, Dhaene, and Denuit (2001), among others.

The current version of **actuar** contains four visible functions related to the above problems: two for the calculation of the aggregate claim amount distribution and two for ruin probability calculations.

We briefly expose the underlying models before we introduce each set of functions.

3.1. The collective risk model

Let random variable S represent the aggregate claim amount (or total amount of claims) of a portfolio of independent risks over a fixed period of time, random variable N represent the number of claims (or frequency) in the portfolio over that period, and random variable C_j represent the amount of claim j (or severity). Then, we have the random sum

$$S = C_1 + \cdots + C_N, \tag{16}$$

where we assume that $C_1, C_2, \ldots$ are mutually independent and identically distributed random variables each independent of N. The task at hand consists in evaluating numerically the cdf of S, given by

$$\begin{aligned} F_S(x) &= \mathsf{P}[S \leq x] \\ &= \sum_{n=0}^{\infty} \mathsf{P}[S \leq x | N = n] p_n \\ &= \sum_{n=0}^{\infty} F_C^{*n}(x) p_n, \end{aligned} \tag{17}$$

where $F_C(x) = \mathsf{P}[C \leq x]$ is the common cdf of $C_1, \ldots, C_n$, $p_n = \mathsf{P}[N = n]$ and $F_C^{*n}(x) =$

$\mathsf{P}[C_1 + \cdots + C_n \leq x]$ is the n-fold convolution of $F_C(\cdot)$. If C is discrete on $0, 1, 2, \ldots$, one has

$$F_C^{*k}(x) = \begin{cases} I\{x \geq 0\}, & k = 0 \\ F_C(x), & k = 1 \\ \sum_{y=0}^{x} F_C^{*(k-1)}(x-y) f_C(y), & k = 2, 3, \ldots \end{cases} \tag{18}$$

3.2. Discretization of claim amount distributions

Some numerical techniques to compute the aggregate claim amount distribution (see Section 3.3) require a discrete arithmetic claim amount distribution; that is, a distribution defined on $0, h, 2h, \ldots$ for some step (or span, or lag) h. The package provides function `discretize` to discretize a continuous distribution. (The function can also be used to modify the support of an already discrete distribution, but this requires additional care.)

Let $F(x)$ denote the cdf of the distribution to discretize on some interval (a, b) and f_x denote the probability mass at x in the discretized distribution. Currently, `discretize` supports the following four discretization methods.

1. Upper discretization, or forward difference of $F(x)$:

$$f_x = F(x + h) - F(x) \tag{19}$$

   for $x = a, a + h, \ldots, b - h$. The discretized cdf is always above the true cdf.

2. Lower discretization, or backward difference of $F(x)$:

$$f_x = \begin{cases} F(a), & x = a \\ F(x) - F(x - h), & x = a + h, \ldots, b. \end{cases} \tag{20}$$

   The discretized cdf is always under the true cdf.

3. Rounding of the random variable, or the midpoint method:

$$f_x = \begin{cases} F(a + h/2), & x = a \\ F(x + h/2) - F(x - h/2), & x = a + h, \ldots, b - h. \end{cases} \tag{21}$$

   The true cdf passes exactly midway through the steps of the discretized cdf.

4. Unbiased, or local matching of the first moment method:

$$f_x = \begin{cases} \dfrac{\mathsf{E}[X \wedge a] - \mathsf{E}[X \wedge a + h]}{h} + 1 - F(a), & x = a \\ \dfrac{2\mathsf{E}[X \wedge x] - \mathsf{E}[X \wedge x - h] - \mathsf{E}[X \wedge x + h]}{h}, & a < x < b \\ \dfrac{\mathsf{E}[X \wedge b] - \mathsf{E}[X \wedge b - h]}{h} - 1 + F(b), & x = b. \end{cases} \tag{22}$$

   The discretized and the true distributions have the same total probability and expected value on (a, b).

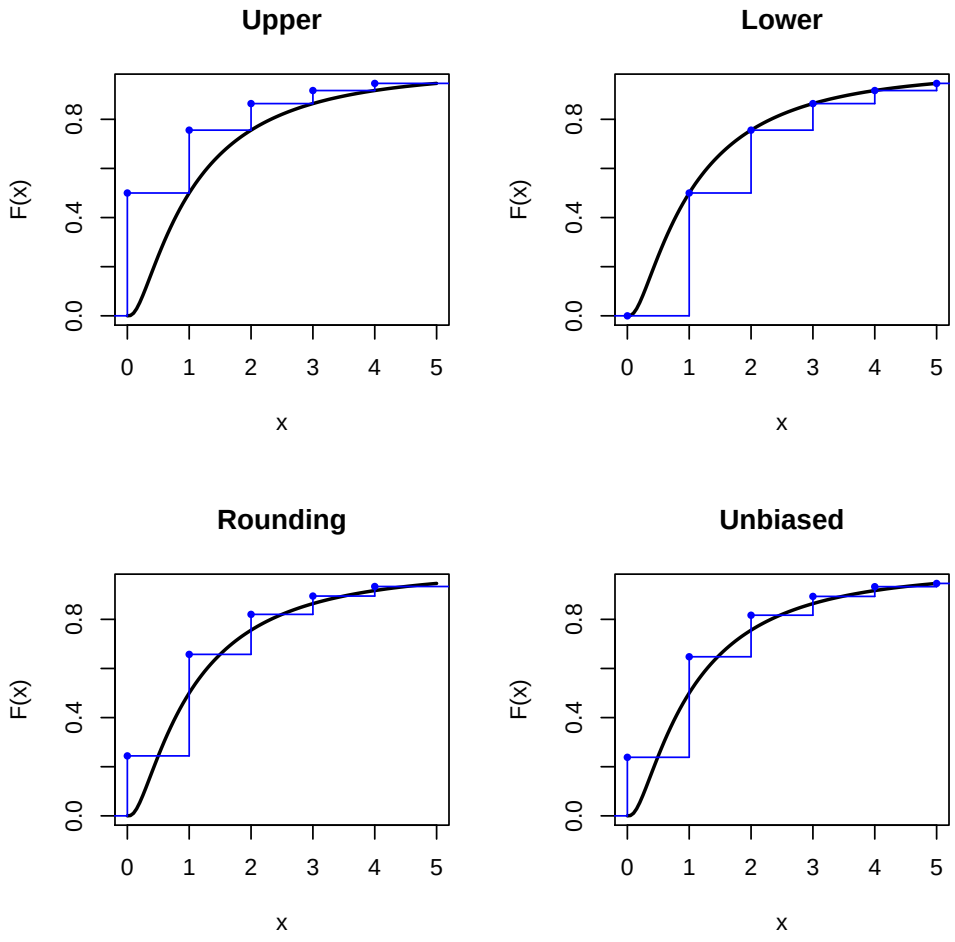

Figure 4: Comparison of four discretization methods

Figure 4 illustrates the four methods. It should be noted that although very close in this example, the rounding and unbiased methods are not identical.

Usage of `discretize` is similar to R's plotting function `curve`. The cdf to discretize and, for the unbiased method only, the limited expected value function are passed to `discretize` as expressions in `x`. The other arguments are the upper and lower bounds of the discretization interval, the step h and the discretization method. For example, upper and unbiased discretizations of a Gamma$(2, 1)$ distribution on $(0, 17)$ with a step of 0.5 are achieved with, respectively,

```
R> fx <- discretize(pgamma(x, 2, 1), method = "upper",
+     from = 0, to = 17, step = 0.5)
R> fx <- discretize(pgamma(x, 2, 1), method = "unbiased",
+     lev = levgamma(x, 2, 1), from = 0, to = 17, step = 0.5)
```

Function `discretize` is written in a modular fashion making it simple to add other discretization methods if needed.

3.3. Calculation of the aggregate claim amount distribution

Function `aggregateDist` serves as a unique front end for various methods to compute or approximate the cdf of the aggregate claim amount random variable S. Currently, five methods are supported.

1. Recursive calculation using the algorithm of Panjer (1981). This requires the severity distribution to be discrete arithmetic on $0, 1, 2, \ldots, m$ for some monetary unit and the frequency distribution to be a member of either the $(a, b, 0)$ or $(a, b, 1)$ family of distributions (Klugman *et al.* 2004). (These families contain the Poisson, binomial, negative binomial and logarithmic distributions and their extensions with an arbitrary mass at $x = 0$.) The general recursive formula is:

$$f_S(x) = \frac{(p_1 - (a+b)p_0)f_C(x) + \sum_{y=1}^{\min(x,m)}(a + by/x)f_C(y)f_S(x-y)}{1 - af_C(0)},$$

with starting value $f_S(0) = P_N(f_C(0))$, where $P_N(\cdot)$ is the probability generating function of N. Probabilities are computed until their sum is arbitrarily close to 1.

The recursions are done in C to dramatically increase speed. One difficulty the programmer is facing is the unknown length of the output. This was solved using a common, simple and fast technique: first allocate an arbitrary amount of memory and double this amount each time the allocated space gets full.

2. Exact calculation by numerical convolutions using (17) and (18). This also requires a discrete severity distribution. However, there is no restriction on the shape of the frequency distribution. The package merely implements the sum (17), the convolutions being computed with R's function `convolve`, which in turn uses the Fast Fourier Transform. This approach is practical for small problems only, even on today's fast computers.

3. Normal approximation of the cdf, that is

$$F_S(x) \approx \Phi\left(\frac{x - \mu_S}{\sigma_S}\right), \tag{23}$$

where $\mu_S = \mathsf{E}[S]$ and $\sigma_S^2 = \mathsf{VAR}[S]$. For most realistic models, this approximation is rather crude in the tails of the distribution.

4. Normal Power II approximation of the cdf, that is

$$F_S(x) \approx \Phi\left(-\frac{3}{\gamma_S} + \sqrt{\frac{9}{\gamma_S^2} + 1 + \frac{6}{\gamma_S}\frac{x - \mu_S}{\sigma_S}}\right), \tag{24}$$

where $\gamma_S = \mathsf{E}[(S-\mu_S)^3]/\sigma_S^{3/2}$. The approximation is valid for $x > \mu_S$ only and performs reasonably well when $\gamma_S < 1$. See Daykin, Pentikäinen, and Pesonen (1994) for details.

5. Simulation of a random sample from S and approximation of $F_S(x)$ by the empirical cdf (8). The simulation itself is done with function `simul` (see Section 4). This function admits very general hierarchical models for both the frequency and the severity components.

Here also, adding other methods to `aggregateDist` is simple due to its modular conception.

The arguments of `aggregateDist` differ depending on the calculation method; see the help page for details. One interesting argument to note is `x.scale` to specify the monetary unit of the severity distribution. This way, one does not have to mentally do the conversion between the support of $0, 1, 2, \ldots$ assumed by the recursive and convolution methods and the true support of S.

The function returns an object of class `"aggregateDist"` inheriting from the `"function"` class. Thus, one can use the object as a function to compute the value of $F_S(x)$ in any x.

For illustration purposes, consider the following model: the distribution of S is a compound Poisson with parameter $\lambda = 10$ and severity distribution Gamma$(2, 1)$. To obtain an approximation of the cdf of S we first discretize the gamma distribution on $(0, 22)$ with the unbiased method and a step of 0.5, and then use the recursive method in `aggregateDist`:

```
R> fx <- discretize(pgamma(x, 2, 1), from = 0, to = 22,
+     step = 0.5, method = "unbiased", lev = levgamma(x, 2, 1))
R> Fs <- aggregateDist("recursive", model.freq = "poisson",
+     model.sev = fx, lambda = 10, x.scale = 0.5)
R> summary(Fs)

Aggregate Claim Amount Empirical CDF:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0.0    14.5    19.5    20.0    25.0    71.0
```

Hence, object `Fs` contains an empirical cdf with support

```
R> knots(Fs)

  [1]  0.0  0.5  1.0  1.5  2.0  2.5  3.0  3.5  4.0  4.5  5.0  5.5
 [13]  6.0  6.5  7.0  7.5  8.0  8.5  9.0  9.5 10.0 10.5 11.0 11.5
 [25] 12.0 12.5 13.0 13.5 14.0 14.5 15.0 15.5 16.0 16.5 17.0 17.5
 [37] 18.0 18.5 19.0 19.5 20.0 20.5 21.0 21.5 22.0 22.5 23.0 23.5
 [49] 24.0 24.5 25.0 25.5 26.0 26.5 27.0 27.5 28.0 28.5 29.0 29.5
 [61] 30.0 30.5 31.0 31.5 32.0 32.5 33.0 33.5 34.0 34.5 35.0 35.5
 [73] 36.0 36.5 37.0 37.5 38.0 38.5 39.0 39.5 40.0 40.5 41.0 41.5
 [85] 42.0 42.5 43.0 43.5 44.0 44.5 45.0 45.5 46.0 46.5 47.0 47.5
 [97] 48.0 48.5 49.0 49.5 50.0 50.5 51.0 51.5 52.0 52.5 53.0 53.5
[109] 54.0 54.5 55.0 55.5 56.0 56.5 57.0 57.5 58.0 58.5 59.0 59.5
[121] 60.0 60.5 61.0 61.5 62.0 62.5 63.0 63.5 64.0 64.5 65.0 65.5
[133] 66.0 66.5 67.0 67.5 68.0 68.5 69.0 69.5 70.0 70.5 71.0
```

A nice graph of this function is obtained with a method of `plot` (see Figure 5):

```
R> plot(Fs, do.points = FALSE, verticals = TRUE, xlim = c(0, 60))
```

The package defines a few summary methods to extract information from `"aggregateDist"` objects. First, there are methods of `mean` and `quantile` to easily compute the mean and obtain the quantiles of the approximate distribution:

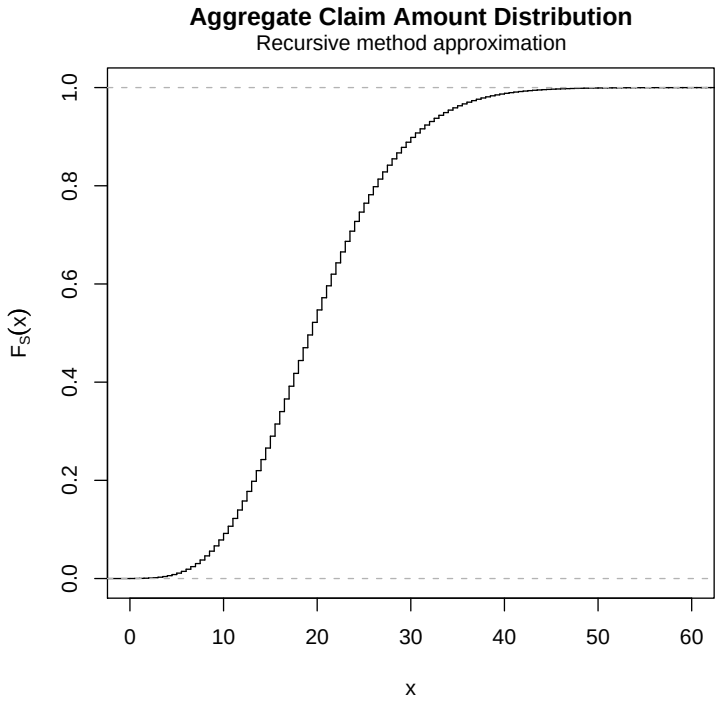

Figure 5: Graphic of the empirical cdf of S obtained with the recursive method

```
R> mean(Fs)

[1] 20

R> quantile(Fs)

  25%   50%   75%   90%   95% 97.5%   99% 99.5%
 14.5  19.5  25.0  30.5  34.0  37.0  41.0  43.5

R> quantile(Fs, 0.999)

99.9%
 49.5
```

Second, the package introduces the generic functions `VaR` and `CTE` with methods for objects of class `"aggregateDist"`. The former computes the value-at-risk VaR_α such that

$$\mathsf{P}[S \leq \mathsf{VaR}_\alpha] = \alpha, \tag{25}$$

where α is the confidence level. Thus, the value-at-risk is nothing else than a quantile. As for the method of `CTE`, it computes the conditional tail expectation

$$\mathsf{CTE}_\alpha = \mathsf{E}[S|S > \mathsf{VaR}_\alpha]. \tag{26}$$

Here are examples using object `Fs` obtained above:

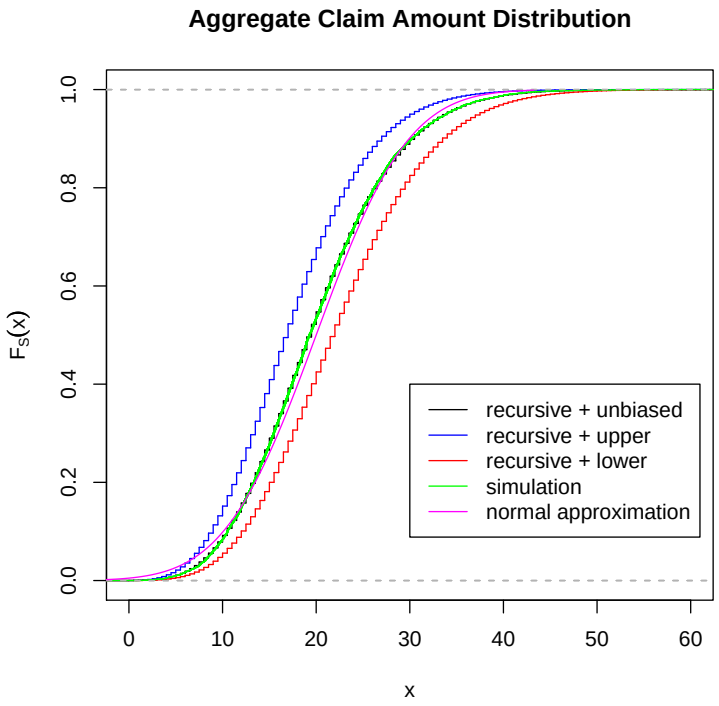

Figure 6: Comparison between the empirical or approximate cdf of S obtained with five different methods

```
R> VaR(Fs)

 90%  95%  99%
30.5 34.0 41.0

R> CTE(Fs)

  90%   95%   99%
35.42 38.55 45.01
```

To conclude on the subject, Figure 6 shows the cdf of S using five of the many combinations of discretization and calculation method supported by **actuar**.

3.4. The continuous time ruin model

We now turn to the multi-period ruin problem. Let $U(t)$ denote the surplus of an insurance company at time t, $c(t)$ denote premiums collected through time t, and $S(t)$ denote aggregate claims paid through time t. If u is the initial surplus at time $t = 0$, then a mathematically convenient definition of $U(t)$ is

$$U(t) = u + c(t) - S(t). \tag{27}$$

As mentioned previously, technical ruin of the insurance company occurs when the surplus becomes negative. Therefore, the definition of the infinite time probability of ruin is

$$\psi(u) = \mathsf{P}[U(t) < 0 \text{ for some } t \geq 0]. \tag{28}$$

We define some other quantities needed in the sequel. Let $N(t)$ denote the number of claims up to time $t \geq 0$ and C_j denote the amount of claim j. Then the definition of $S(t)$ is analogous to (16):

$$S(t) = C_1 + \cdots + C_{N(t)}, \tag{29}$$

assuming $N(0) = 0$ and $S(t) = 0$ as long as $N(t) = 0$. Furthermore, let T_j denote the time when claim j occurs, such that $T_1 < T_2 < T_3 < \ldots$ Then the random variable of the interarrival (or wait) time between claim $j-1$ and claim j is defined as $W_1 = T_1$ and

$$W_j = T_j - T_{j-1}, \quad j \geq 2. \tag{30}$$

For the rest of this discussion, we make the following assumptions:

1. premiums are collected at a constant rate c, hence $c(t) = ct$;
2. the sequence $\{T_j\}_{j\geq 1}$ forms an ordinary renewal process, with the consequence that random variables $W_1, W_2, \ldots$ are independent and identically distributed;
3. claim amounts $C_1, C_2, \ldots$ are independent and identically distributed.

3.5. Adjustment coefficient

The quantity known as the adjustment coefficient ρ hardly has any physical interpretation, but it is useful as an approximation to the probability of ruin since we have the inequality

$$\psi(u) \leq e^{-\rho u}, \quad u \geq 0.$$

The adjustment coefficient is defined as the smallest strictly positive solution (if it exists) of the Lundberg equation

$$h(t) = \mathsf{E}[e^{tC-tcW}] = 1, \tag{31}$$

where the premium rate c satisfies the positive safety loading constraint $\mathsf{E}[C - cW] < 0$. If C and W are independent, as in the most common models, then the equation can be rewritten as

$$h(t) = M_C(t)M_W(-tc) = 1. \tag{32}$$

Function `adjCoef` of **actuar** computes the adjustment coefficient ρ from the following arguments: either the two moment generating functions $M_C(t)$ and $M_W(t)$ (thereby assuming independence) or else function $h(t)$; the premium rate c; the upper bound of the support of $M_C(t)$ or any other upper bound for ρ.

For example, if W and C are independent, $W \sim$ Exponential(2), $C \sim$ Exponential(1) and the premium rate is $c = 2.4$ (for a safety loading of 20% using the expected value premium principle), then the adjustment coefficient is

```
R> adjCoef(mgf.claim = mgfexp(x), mgf.wait = mgfexp(x, 2),
+     premium.rate = 2.4, upper = 1)

[1] 0.1667
```

The function also supports models with proportional or excess-of-loss reinsurance (Centeno 2002). Under the first type of treaty, an insurer pays a proportion α of every loss and the rest is paid by the reinsurer. Then, for fixed α the adjustment coefficient is the solution of

$$h(t) = \mathsf{E}[e^{t\alpha C - tc(\alpha)W}] = 1. \tag{33}$$

Under an excess-of-loss treaty, the primary insurer pays each claim up to a limit L. Again, for fixed L, the adjustment coefficient is the solution of

$$h(t) = \mathsf{E}[e^{t\min(C,L) - tc(L)W}] = 1. \tag{34}$$

For models with reinsurance, `adjCoef` returns an object of class `"adjCoef"` inheriting from the `"function"` class. One can then use the object to compute the adjustment coefficient for any retention rate α or retention limit L. The package also defines a method of `plot` for these objects.

For example, using the same assumptions as above with proportional reinsurance and a 30% safety loading for the reinsurer, the adjustment coefficient as a function of $\alpha \in [0, 1]$ is (see Figure 7 for the graph):

```
R> mgfx <- function(x, y) mgfexp(x * y)
R> p <- function(x) 2.6 * x - 0.2
R> rho <- adjCoef(mgfx, mgfexp(x, 2), premium = p, upper = 1,
+     reins = "prop", from = 0, to = 1)
R> rho(c(0.75, 0.8, 0.9, 1))

[1] 0.1905 0.1862 0.1765 0.1667

R> plot(rho)
```

3.6. Probability of ruin

In this subsection, we always assume that interarrival times and claim amounts are independent.

The main difficulty with the calculation of the infinite time probability of ruin lies in the lack of explicit formulas except for the most simple models. If interarrival times are Exponential(λ) distributed (Poisson claim number process) and claim amounts are Exponential(β) distributed, then

$$\psi(u) = \frac{\lambda}{c\beta}\, e^{-(\beta - \lambda/c)u}. \tag{35}$$

If the frequency assumption of this model is defensible, the severity assumption can hardly be used beyond illustration purposes.

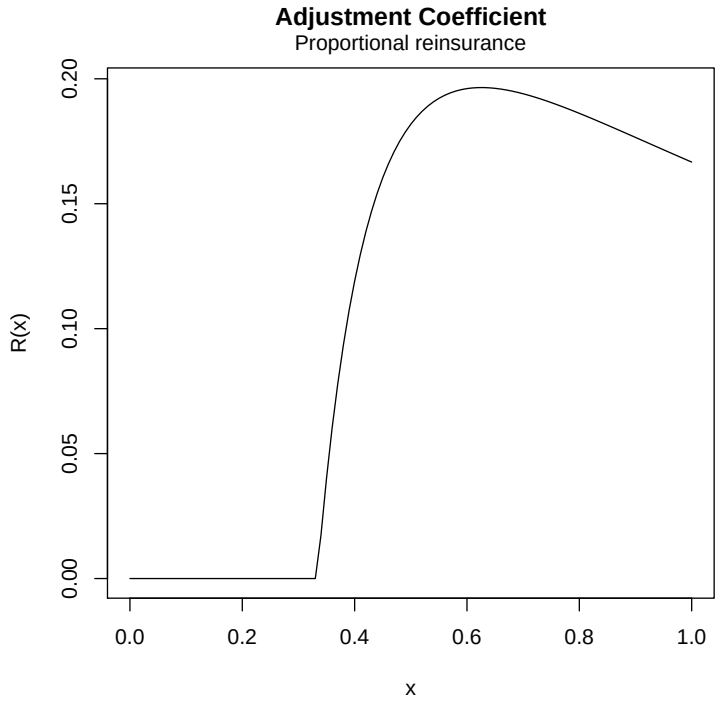

Figure 7: Adjustment coefficient as a function of the retention rate

Fortunately, phase-type distributions have come to the rescue since the early 1990s. Asmussen and Rolski (1991) first show that in the classical Cramér–Lundberg model where interarrival times are Exponential(λ) distributed, if claim amounts are Phase-type($\boldsymbol{\pi}, \boldsymbol{T}$) distributed, then $\psi(u) = 1 - F(u)$, where F is Phase-type($\boldsymbol{\pi}_+, \boldsymbol{Q}$) with

$$\begin{aligned} \boldsymbol{\pi}_+ &= -\frac{\lambda}{c}\,\boldsymbol{\pi}\boldsymbol{T}^{-1} \\ \boldsymbol{Q} &= \boldsymbol{T} + \boldsymbol{t}\boldsymbol{\pi}_+, \end{aligned} \tag{36}$$

with $\boldsymbol{t} = -\boldsymbol{T}\boldsymbol{e}$ as in Section 2.1.

In the more general Sparre Andersen model where interarrival times can have any Phase-type($\boldsymbol{\nu}, \boldsymbol{S}$) distribution, Asmussen and Rolski (1991) also show that using the same claim severity assumption as above, one still has $\psi(u) = 1 - F(u)$ where F is Phase-type($\boldsymbol{\pi}_+, \boldsymbol{Q}$), but with parameters

$$\boldsymbol{\pi}_+ = \frac{\boldsymbol{e}'(\boldsymbol{Q} - \boldsymbol{T})}{c\boldsymbol{e}'\boldsymbol{t}} \tag{37}$$

and $\boldsymbol{Q}$ solution of

$$\begin{aligned} \boldsymbol{Q} &= \Psi(\boldsymbol{Q}) \\ &= \boldsymbol{T} - \boldsymbol{t}\boldsymbol{\pi}\left[(\boldsymbol{I}_n \otimes \boldsymbol{\nu})(\boldsymbol{Q} \oplus \boldsymbol{S})^{-1}(\boldsymbol{I}_n \otimes \boldsymbol{s})\right]. \end{aligned} \tag{38}$$

In the above, $\boldsymbol{s} = -\boldsymbol{S}\boldsymbol{e}$, $\boldsymbol{I}_n$ is the $n \times n$ identity matrix, $\otimes$ denotes the usual Kronecker product between two matrices and $\oplus$ is the Kronecker sum defined as

$$\boldsymbol{A}_{m\times m} \oplus \boldsymbol{B}_{n\times n} = \boldsymbol{A} \otimes \boldsymbol{I}_n + \boldsymbol{B} \otimes \boldsymbol{I}_m. \tag{39}$$

Function `ruin` of **actuar** returns a function object of class `"ruin"` to compute the probability of ruin for any initial surplus u. In all cases except the exponential/exponential model where (35) is used, the output object calls function `pphtype` to compute the ruin probabilities.

Some thought went into the interface of `ruin`. Obviously, all models can be specified using phase-type distributions, but the authors wanted users to have easy access to the most common models involving exponential and Erlang distributions. Hence, one first states the claim amount and interarrival times models with any combination of `"exponential"`, `"Erlang"` and `"phase-type"`. Then, one passes the parameters of each model using lists with components named after the corresponding parameters of `dexp`, `dgamma` and `dphtype`. If a component `"weights"` is found in a list, the model is a mixture of exponential or Erlang (mixtures of phase-type are not supported). Every component of the parameter lists is recycled as needed.

The following examples should make the matter clearer. (All examples use $c = 1$, the default value in `ruin`.) First, for the exponential/exponential model, one has

```
R> psi <- ruin(claims = "e", par.claims = list(rate = 5),
+     wait = "e", par.wait = list(rate = 3))
R> psi

function (u, survival = FALSE, lower.tail = !survival)
{
    res <- 0.6 * exp(-(2) * u)
    if (lower.tail)
        res
    else 0.5 - res + 0.5
}
<environment: 0x19af2150>
attr(,"class")
[1] "ruin"     "function"

R> psi(0:10)

 [1] 6.000e-01 8.120e-02 1.099e-02 1.487e-03 2.013e-04 2.724e-05
 [7] 3.687e-06 4.989e-07 6.752e-08 9.138e-09 1.237e-09
```

Second, for a model with mixture of two exponentials claim amounts and exponential interarrival times, the simplest call to `ruin` is

```
R> ruin(claims = "e", par.claims = list(rate = c(3, 7),
+     weights = 0.5), wait = "e", par.wait = list(rate = 3))

function (u, survival = FALSE, lower.tail = !survival)
pphtype(u, c(0.5, 0.214285714285714), c(-1.5, 3.5, 0.642857142857143,
-5.5), lower.tail = !lower.tail)
<environment: 0x1dfba1c>
attr(,"class")
[1] "ruin"     "function"
```

Finally, one will obtain a function to compute ruin probabilities in a model with phase-type claim amounts and mixture of exponentials interarrival times with

```
R> prob <- c(0.5614, 0.4386)
R> rates <- matrix(c(-8.64, 0.101, 1.997, -1.095), 2, 2)
R> ruin(claims = "p", par.claims = list(prob = prob, rates = rates),
+     wait = "e", par.wait = list(rate = c(5, 1), weights = c(0.4, 0.6)))

function (u, survival = FALSE, lower.tail = !survival)
pphtype(u, c(0.146595513877824, 0.761505562273639), c(-7.66616600130962,
0.246715940794557, 7.05568145018378, -0.338063471100003),
lower.tail = !lower.tail)
<environment: 0x19ba1208>
attr(,"class")
[1] "ruin"     "function"
```

To ease plotting of the probability of ruin function, the package provides a method of `plot` for objects returned by `ruin` that is a simple wrapper for `curve` (see Figure 8):

```
R> psi <- ruin(claims = "p", par.claims = list(prob = prob, rates = rates),
+     wait = "e", par.wait = list(rate = c(5, 1), weights = c(0.4, 0.6)))
R> plot(psi, from = 0, to = 50)
```

4. Simulation of compound hierarchical models

Function `simul` simulates portfolios of data following compound models of the form (16) where both the frequency and the severity components can have a hierarchical structure. The main characteristic of hierarchical models is to have the probability law at some level in the classification structure be conditional on the outcome in previous levels. For example, consider the following compound hierarchical model:

$$S_{ijt} = C_{ijt1} + \cdots + C_{ijtN_{ijt}}, \tag{40}$$

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

$$\begin{aligned} N_{ijt}|\Lambda_{ij}, \Phi_i &\sim \text{Poisson}(w_{ijt}\Lambda_{ij}) & C_{ijtu}|\Theta_{ij}, \Psi_i &\sim \text{Lognormal}(\Theta_{ij}, 1) \\ \Lambda_{ij}|\Phi_i &\sim \text{Gamma}(\Phi_i, 1) & \Theta_{ij}|\Psi_i &\sim N(\Psi_i, 1) \\ \Phi_i &\sim \text{Exponential}(2) & \Psi_i &\sim N(2, 0.1). \end{aligned} \tag{41}$$

The random variables Φ_i, Λ_{ij}, Ψ_i and Θ_{ij} are generally seen as risk parameters in the actuarial literature. The w_{ijt}s are known weights, or volumes. Using as convention to number the data level 0, the above is a two-level hierarchical model.

Goulet and Pouliot (2008) describe in detail the model specification method used in `simul`. For the sake of completeness, we briefly outline this method here.

A hierarchical model is completely specified by the number of nodes at each level (I, $J_1, \ldots, J_I$ and $n_{11}, \ldots, n_{IJ}$, above) and by the probability laws at each level. The number of nodes is

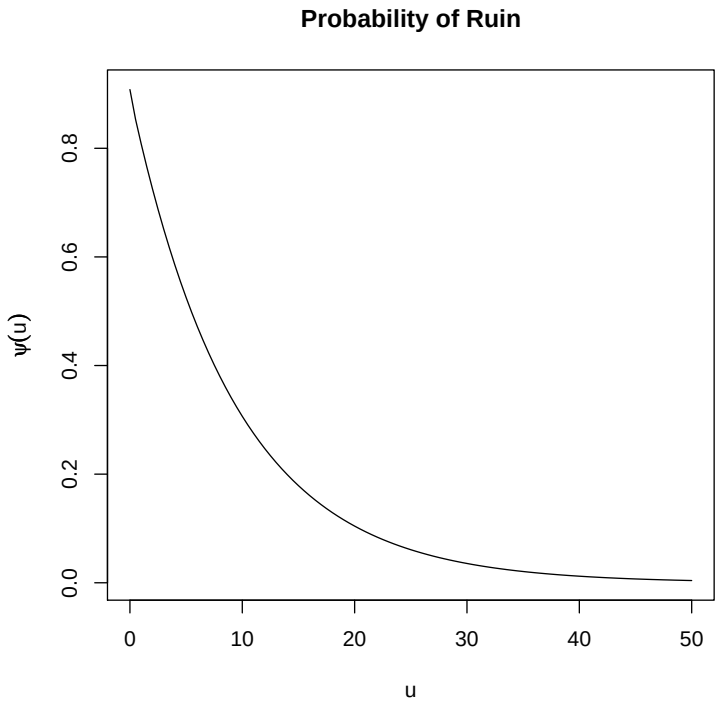

Figure 8: Graphic of the probability of ruin as a function of the initial surplus u

passed to `simul` 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, 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`.

Function `simul` also 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.

Hence, function `simul` 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.

For example, assuming that $I = 2$, $J_1 = 4$, $J_2 = 3$, $n_{11} = \cdots = n_{14} = 4$ and $n_{21} = n_{22} = n_{23} = 5$ in model (41) above, and that weights are simply simulated from a uniform distribution on $(0.5, 2.5)$, then simulation of a data set with `simul` is achieved with:

```
R> nodes <- list(cohort = 2, entity = c(4, 3), year = c(4,
+     4, 4, 4, 5, 5, 5))
```

```
R> mf <- expression(cohort = rexp(2), entity = rgamma(cohort,
+     1), year = rpois(weights * entity))
R> ms <- expression(cohort = rnorm(2, sqrt(0.1)), entity = rnorm(cohort,
+     1), year = rlnorm(entity, 1))
R> wijt <- runif(31, 0.5, 2.5)
R> pf <- simul(nodes = nodes, model.freq = mf, model.sev = ms,
+     weights = wijt)
```

The function returns the variates in a two-dimension list of class `"portfolio"` containing all the individual claim amounts for each entity. Such an object can be seen as a three-dimensional 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 (subscripts i and j in the notation above). Displaying the complete content of the object returned by `simul` can be impractical. For this reason, the `print` method for this class only prints the simulation model and the number of claims in each node:

```
R> pf

Portfolio of claim amounts

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

  Number of claims per node:

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

The package defines methods for four generic functions to easily access key quantities of the simulated portfolio.

1. By default, the method of `aggregate` returns the values of aggregate claim amounts 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:

```
R> aggregate(pf)

     cohort entity year.1 year.2 year.3 year.4 year.5
[1,]      1      1  31.37  7.521 11.383  0.000     NA
[2,]      1      2   0.00  0.000  0.000  0.000     NA
[3,]      1      3   0.00 72.706  0.000 23.981     NA
[4,]      1      4   0.00 98.130 50.622 55.705     NA
[5,]      2      1   0.00 11.793  2.253  2.397  10.48
[6,]      2      2   0.00  0.000  0.000  0.000   0.00
[7,]      2      3  44.81 88.737 57.593 14.589   0.00

R> 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
```

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

```
R> frequency(pf)

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

R> frequency(pf, by = "cohort")

     cohort freq
[1,]      1   17
[2,]      2   16
```

3. The method of `severity` (a generic function introduced by the package) returns the individual claim amounts 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 constitutes a "period" of observation. For this reason, the method provides an argument `splitcol` in case one would like to extract separately the individual claim amounts of one or more periods:

```
R> severity(pf)

$main
     cohort entity claim.1 claim.2 claim.3 claim.4 claim.5 claim.6
[1,]      1      1   7.974  23.401   3.153   4.368  11.383      NA
[2,]      1      2      NA      NA      NA      NA      NA      NA
[3,]      1      3   3.817  41.979  26.910   4.903  19.078      NA
[4,]      1      4  98.130  50.622  55.705      NA      NA      NA
[5,]      2      1  11.793   2.253   2.397   9.472   1.004      NA
[6,]      2      2      NA      NA      NA      NA      NA      NA
[7,]      2      3  14.322  11.522  18.966  33.108  15.532   14.99
     claim.7 claim.8 claim.9 claim.10 claim.11
[1,]      NA      NA      NA       NA       NA
[2,]      NA      NA      NA       NA       NA
[3,]      NA      NA      NA       NA       NA
[4,]      NA      NA      NA       NA       NA
[5,]      NA      NA      NA       NA       NA
[6,]      NA      NA      NA       NA       NA
[7,]   25.11   40.15   17.44    4.426    10.16

$split
NULL

R> severity(pf, splitcol = 1)

$main
     cohort entity claim.1 claim.2 claim.3 claim.4 claim.5 claim.6
[1,]      1      1   3.153   4.368  11.383      NA      NA      NA
[2,]      1      2      NA      NA      NA      NA      NA      NA
[3,]      1      3   3.817  41.979  26.910   4.903  19.078      NA
[4,]      1      4  98.130  50.622  55.705      NA      NA      NA
[5,]      2      1  11.793   2.253   2.397   9.472   1.004      NA
[6,]      2      2      NA      NA      NA      NA      NA      NA
[7,]      2      3  33.108  15.532  14.990  25.107  40.150   17.44
     claim.7 claim.8
[1,]      NA      NA
[2,]      NA      NA
[3,]      NA      NA
[4,]      NA      NA
[5,]      NA      NA
[6,]      NA      NA
[7,]   4.426   10.16

$split
     cohort entity 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
```

4. The method of `weights` extracts the weights matrix from a simulated data set:

```
R> weights(pf)

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

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 `simul.summaries` help page for details.

Function `simul` was used to simulate the data in Forgues, Goulet, and Lu (2006).

5. Credibility theory

Credibility models are actuarial tools to distribute premiums fairly among a heterogeneous group of policyholders (henceforth called *entities*). More generally, they can be seen as prediction methods applicable in any setting where repeated measures are made for subjects with different risk levels.

The credibility theory facilities of **actuar** consist of the matrix `hachemeister` containing the famous data set of Hachemeister (1975) and the function `cm` to fit credibility models.

5.1. Hachemeister data set

The data set of Hachemeister (1975) consists of private passenger bodily injury insurance average claim amounts, and the corresponding number of claims, for five U.S. states over 12 quarters between July 1970 and June 1973. The data set is included in the package in the form of a matrix with 5 rows and 25 columns. The first column contains a state index (`state`), columns 2–13 contain the claim averages (`ratio.1`, ..., `ratio.12`) and columns 14–25 contain the claim numbers (`weight.1`, ..., `weight.12`).

5.2. Hierarchical credibility model

The linear model fitting function of R is named `lm`. Since credibility models are very close in many respects to linear models, and since the credibility model fitting function of **actuar** borrows much of its interface from `lm`, we named the credibility function `cm`.

Function cm acts as a unified interface for all credibility models supported by the package. Currently, these are the unidimensional models of Bühlmann (1969) and Bühlmann and Straub (1970), the hierarchical model of Jewell (1975) (of which the first two are special cases) and the regression model of Hachemeister (1975). The modular design of cm makes it easy to add new models if desired.

This subsection concentrates on usage of cm for hierarchical models.

There are some variations in the formulas of the hierarchical model in the literature. We compute the credibility premiums as given in Bühlmann and Jewell (1987) or Bühlmann and Gisler (2005). We support three types of estimators of the between variance structure parameters: the unbiased estimators of Bühlmann and Gisler (2005) (the default), the slightly different version of Ohlsson (2005) and the iterative pseudo-estimators as found in Goovaerts and Hoogstad (1987) or Goulet (1998). See Belhadj, Goulet, and Ouellet (2008) for further discussion on this topic.

The credibility modeling function assumes that data is available in the format most practical applications would use, namely a rectangular array (matrix or data frame) with entity observations in the rows and with one or more classification index columns (numeric or character). One will recognize the output format of simul and its summary methods.

Then, function cm works much the same as lm. It takes in argument: a formula of the form ~ terms describing the hierarchical interactions in a data set; the data set containing the variables referenced in the formula; the names of the columns where the ratios and the weights are to be found in the data set. The latter should contain at least two nodes in each level and more than one period of experience for at least one entity. Missing values are represented by NAs. There can be entities with no experience (complete lines of NAs).

In order to give an easily reproducible example, we group states 1 and 3 of the Hachemeister data set into one cohort and states 2, 4 and 5 into another. This shows that data does not have to be sorted by level. The fitted model using the iterative estimators is:

```
R> X <- cbind(cohort = c(1, 2, 1, 2, 2), hachemeister)
R> fit <- cm(~cohort + cohort:state, data = X, ratios = ratio.1:ratio.12,
+     weights = weight.1:weight.12, method = "iterative")
R> fit

Call:
cm(formula = ~cohort + cohort:state, data = X, ratios = ratio.1:ratio.12,
    weights = weight.1:weight.12, method = "iterative")

Structure Parameters Estimators

  Collective premium: 1746

  Between cohort variance: 88981
  Within cohort/Between state variance: 10952
  Within state variance: 139120026
```

The function returns a fitted model object of class "cm" containing the estimators of the structure parameters. To compute the credibility premiums, one calls a method of predict for this class:

```
R> predict(fit)

$cohort
[1] 1949 1543

$state
[1] 2048 1524 1875 1497 1585
```

One can also obtain a nicely formatted view of the most important results with a call to `summary`:

```
R> summary(fit)

Call:
cm(formula = ~cohort + cohort:state, data = X, ratios = ratio.1:ratio.12,
    weights = weight.1:weight.12, method = "iterative")

Structure Parameters Estimators

  Collective premium: 1746

  Between cohort variance: 88981
  Within cohort/Between state variance: 10952
  Within state variance: 139120026

Detailed premiums

  Level: cohort
    cohort Indiv. mean Weight Cred. factor Cred. premium
    1      1967        1.407  0.9196       1949
    2      1528        1.596  0.9284       1543

  Level: state
    cohort state Indiv. mean Weight Cred. factor Cred. premium
    1      1     2061        100155 0.8874       2048
    2      2     1511         19895 0.6103       1524
    1      3     1806         13735 0.5195       1875
    2      4     1353          4152 0.2463       1497
    2      5     1600         36110 0.7398       1585
```

The methods of `predict` and `summary` can both report for a subset of the levels by means of an argument `levels`. For example:

```
R> summary(fit, levels = "cohort")

Call:
cm(formula = ~cohort + cohort:state, data = X, ratios = ratio.1:ratio.12,
```

```
    weights = weight.1:weight.12, method = "iterative")

Structure Parameters Estimators

  Collective premium: 1746

  Between cohort variance: 88981
  Within cohort variance: 10952

Detailed premiums

  Level: cohort
    cohort Indiv. mean Weight Cred. factor Cred. premium
    1      1967        1.407  0.9196       1949
    2      1528        1.596  0.9284       1543

R> predict(fit, levels = "cohort")

$cohort
[1] 1949 1543
```

The results above differ from those of Goovaerts and Hoogstad (1987) for the same example because the formulas for the credibility premiums are different.

5.3. Bühlmann and Bühlmann–Straub models

As mentioned above, the Bühlmann and Bühlmann–Straub models are simply one-level hierarchical models. In this case, the Bühlmann–Gisler and Ohlsson estimators of the between variance parameters are both identical to the usual Bühlmann and Straub (1970) estimator, and the iterative estimator is better known as the Bichsel–Straub estimator.

5.4. Regression model of Hachemeister

The regression model of Hachemeister (1975) is a generalization of the Bühlmann–Straub model. If data shows a systematic trend, the latter model will typically under- or overestimate the true premium of an entity. The idea of Hachemeister was to fit to the data a regression model where the parameters are a credibility weighted average of an entity's regression parameters and the group's parameters.

In order to use `cm` to fit a credibility regression model to a data set, one has to specify a vector or matrix of regressors by means of argument `xreg`. For example, fitting the model

$$X_{it} = \beta_0 + \beta_1(12 - t) + \varepsilon_t, \quad t = 1, \ldots, 12$$

to the original data set of Hachemeister is done with

```
R> fit <- cm(~state, hachemeister, xreg = 12:1, ratios = ratio.1:ratio.12,
+     weights = weight.1:weight.12)
R> fit
```

```
Call:
cm(formula = ~state, data = hachemeister, ratios = ratio.1:ratio.12,
    weights = weight.1:weight.12, xreg = 12:1)

Structure Parameters Estimators

  Collective premium: 1885 -32.05

  Between state variance:  145359 -6623.4
                            -6623   301.8
  Within state variance: 49870187
```

Computing the credibility premiums requires to give the "future" values of the regressors as in `predict.lm`, although with a simplified syntax for the one regressor case:

```
R> predict(fit, newdata = 0)

[1] 2437 1651 2073 1507 1759
```

6. Documentation

In addition to the help pages required by the R packaging system, the package includes vignettes and demonstration scripts; running

```
R> vignette(package = "actuar")
```

and

```
R> demo(package = "actuar")
```

at the R prompt will give the list of each.

7. Conclusion

The paper presented the facilities of the R package **actuar** in the fields of loss distribution modeling, risk theory, simulation of compound hierarchical models and credibility theory. We feel this version of the package covers most of the basic needs in these areas. In the future we plan to improve the functions currently available and to start adding more advanced features. For example, future versions of the package should include support for dependence models in risk theory and better handling of regression credibility models.

Obviously, the package left many other fields of Actuarial Science untouched. For this situation to change, we hope that experts in their field will join their efforts to ours and contribute code to the **actuar** project. The project will continue to grow and to improve by and for the community of developers and users.

Finally, if you use R or **actuar** for actuarial analysis, please cite the software in publications. Use

```
R> citation()
```

and

```
R> citation("actuar")
```

for information on how to cite the software.

Acknowledgments

The package would not be at this stage of development without the stimulating contribution of Sébastien Auclair, Louis-Philippe Pouliot and Tommy Ouellet.

This research benefited from financial support from the Natural Sciences and Engineering Research Council of Canada and from the *Chaire d'actuariat* (Actuarial Science Chair) of Université Laval.

Finally, the authors thank two anonymous referees for many improvements to the paper.

References

Asmussen S, Rolski T (1991). "Computational Methods in Risk Theory: A Matrix-Algorithmic Approach." *Insurance: Mathematics and Economics*, **10**, 259–274.

Bates D, Maechler M (2008). ***Matrix**: A Matrix package for R*. R package version 0.999375-7, URL http://CRAN.R-project.org/package=Matrix.

Belhadj H, Goulet V, Ouellet T (2008). "On Parameter Estimation in Hierarchical Credibility." In preparation.

Bühlmann H (1969). "Experience Rating and Credibility." *ASTIN Bulletin*, **5**, 157–165.

Bühlmann H, Gisler A (2005). *A Course in Credibility Theory and its Applications*. Springer-Verlag. ISBN 3-5402575-3-5.

Bühlmann H, Jewell WS (1987). "Hierarchical Credibility Revisited." *Bulletin of the Swiss Association of Actuaries*, **87**, 35–54.

Bühlmann H, Straub E (1970). "Glaubgwürdigkeit für Schadensätze." *Bulletin of the Swiss Association of Actuaries*, **70**, 111–133.

Centeno MdL (2002). "Measuring the Effects of Reinsurance by the Adjustment Coefficient in the Sparre-Anderson Model." *Insurance: Mathematics and Economics*, **30**, 37–49.

Daykin C, Pentikäinen T, Pesonen M (1994). *Practical Risk Theory for Actuaries*. Chapman & Hall, London. ISBN 0-4124285-0-4.

Denuit M, Charpentier A (2004). *Mathématiques de l'assurance non-vie*, volume 1, Principes fondamentaux de théorie du risque. Economica, Paris. ISBN 2-7178485-4-1.

Forgues A, Goulet V, Lu J (2006). "Credibility for Severity Revisited." *North American Actuarial Journal*, **10**(1), 49–62.

Gerber HU (1979). *An Introduction to Mathematical Risk Theory*. Huebner Foundation, Philadelphia.

Goovaerts MJ, Hoogstad WJ (1987). *Credibility Theory*. Number 4 in Surveys of Actuarial Studies. Nationale-Nederlanden N.V., Netherlands.

Goulet V (1998). "Principles and Application of Credibility Theory." *Journal of Actuarial Practice*, **6**, 5–62.

Goulet V, Pouliot LP (2008). "Simulation of Compound Hierarchical Models in R." *North American Actuarial Journal*. Submitted for publication.

Hachemeister CA (1975). "Credibility for Regression Models with Application to Trend." In "Credibility, Theory and Applications," Proceedings of the Berkeley Actuarial Research Conference on Credibility, pp. 129–163. Academic Press, New York.

Hogg RV, Klugman SA (1984). *Loss Distributions*. John Wiley & Sons, New York. ISBN 0-4718792-9-0.

Jewell WS (1975). "The Use of Collateral Data in Credibility Theory: A Hierarchical Model." *Giornale dell'Istituto Italiano degli Attuari*, **38**, 1–16.

Kaas R, Goovaerts M, Dhaene J, Denuit M (2001). *Modern Actuarial Risk Theory*. Kluwer Academic Publishers, Dordrecht. ISBN 0-7923763-6-6.

Klugman SA, Panjer HH, Willmot G (1998). *Loss Models: From Data to Decisions*. John Wiley & Sons, New York. ISBN 0-4712388-4-8.

Klugman SA, Panjer HH, Willmot G (2004). *Loss Models: From Data to Decisions*. John Wiley & Sons, New York, second edition. ISBN 0-4712157-7-5.

Lumley T (2008). ***survival***: *Survival Analysis, Including Penalised Likelihood*. R package version 2.34, URL http://CRAN.R-project.org/package=survival.

Neuts MF (1981). *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach*. Dover Publications. ISBN 978-0-4866834-2-3.

Ohlsson E (2005). "Simplified Estimation of Structure Parameters in Hierarchical Credibility." Presented at the Zurich ASTIN Colloquium, URL http://www.actuaries.org/ASTIN/Colloquia/Zurich/Ohlsson.pdf.

Panjer HH (1981). "Recursive Evaluation of a Family of Compound Distributions." *ASTIN Bulletin*, **12**, 22–26.

Pinheiro J, Bates D, DebRoy S, Sarkar D, the R Development Core Team (2007). ***nlme***: *Linear and Nonlinear Mixed Effects Models*. R package version 3.1-86, URL http://CRAN.R-project.org/package=nlme.

R Development Core Team (2008). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Venables WN, Ripley BD (2002). *Modern Applied Statistics with S*. Springer-Verlag, New York, 4 edition. ISBN 0-3879545-7-0.

Wheeler B (2008). ***SuppDists**: Supplementary Distributions*. R package version 1.1-2, URL http://CRAN.R-project.org/package=SuppDists.

Wuertz D (2007). ***Rmetrics**: Financial Engineering and Computational Finance*. R package version 260.72, URL http://CRAN.R-project.org/package=Rmetrics.

Yan J (2007). "Enjoy the Joy of Copulas: With a Package **copula**." *Journal of Statistical Software*, **21**(4), 1–21. URL http://www.jstatsoft.org/v21/i04/.

Affiliation:

Vincent Goulet
École d'actuariat
Pavillon Alexandre-Vachon
1045, avenue de la Médecine
Université Laval
Québec (QC) G1V 0A6, Canada
E-mail: vincent.goulet@act.ulaval.ca
URL: http://vgoulet.act.ulaval.ca/

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 25, Issue 7 *Submitted:* 2007-11-16
March 2008 *Accepted:* 2008-02-14