actuar: An R Package for Actuarial Science

Vincent Goulet

École d'actuariat, Université Laval

Abstract

The **actuar** project is a package of Actuarial Science functions for the R statistical system. The project was launched in 2005 and the package is available on CRAN (Comprehensive R Archive Network) since February 2006. The current version of the package contains functions for use in the fields of risk theory, loss distributions and credibility theory. This paper presents in detail but in non technical terms the most recent version of the package.

Keywords: R, actuar, loss distributions, risk theory, credibility theory, software, programming language.

1 Introduction

R is a programming language and an environment for statistical computing and graphics (R Development Core Team, 2007). It is a free software version of the award winning S system whose commercial counterpart is S-Plus by Insightful Corporation. Despite some important fundamental differences between S-Plus and R, the two systems are largely compatible and most code written for one runs unaltered in the other.

R can also be considered as a dialect of the S programming language, a language for "programming with data" developed at Bell Laboratories starting from the late 1970s. R is not just another statistical environment (like SPSS or SAS, for example), but a full fledged and self contained programming language with a strong mathematical orientation. It is based on the notion of vector that the many actuaries who have worked, or are still working, with APL or SAS IML have come to know and love.

R is under very active development and is constantly gaining "market shares" in many fields related to statistics. One of its strengths is the possibility to add functionalities to the base system by means of add-on *packages*. Simply put, a package is a coherent set of functions and data. The basic R systems consists of a dozen packages; hundreds of contributed packages covering a wide array of modern statistical methods are available from the *Comprehensive R Archive Network* (CRAN; `http://cran.r-project.org`).

The **actuar** project (Goulet, 2007) is a package of Actuarial Science functions for R. Although various packages on CRAN provide functions that may be of use to actuaries, **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.

This paper reviews, in non technical terms, the various features of the current version of the **actuar** package.

2 Current status of the package

As of this writing, the version of **actuar** available on CRAN is 0.9-3. The feature set of the package can be split in three main categories: loss distributions modeling, risk theory 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 means that instead of merely returning, say, a vector of probabilities, many functions will return an object containing, among other things, the said probabilities. The object can then be manipulated at one's will using various extraction, summary or plotting functions. This allows for a very dynamic modeling–estimation–diagnosis–prediction process that few other statistical packages provide.

The package is released under the GNU General Public License (GPL), version 2 or newer, thereby making it free software that anyone can use, modify and redistribute, to the extent that the derivative work is also released under the GPL.

3 Documentation

It is a requirement of the `R` packaging system that every function and data set in a package has a help page. The **actuar** package follows this requirement strictly. The help page of function `foo` is accessible by typing

```
> ?foo
```

or

```
> help("foo")
```

at the `R` command prompt. Most help pages provide usage examples.

In addition to the help pages, the package includes *vignettes*, longer PDF documents on one or many topics. Running

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

will give the list of available vignettes in the package.

Finally, one will find more comprehensive examples for the various features of the package in the demo scripts (see `?demo`). The list of demos available in the package is given by

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

4 Loss distributions modeling features

Loss distributions is the subset of **actuar** containing the largest number of functions. Some complement features of base `R`, while others provide support for entirely untouched procedures common in Actuarial Science. The following subsections detail the following **actuar** features:

1. introduction of 17 additional probability laws and functions to get the kth true raw and limited moments;
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).

4.1 Probability laws

`R` already includes functions to compute the density function, cumulative distribution function, quantile function of, and to generate variates from a fair number of probability laws. For some root `foo`, the functions are named `dfoo`, `pfoo`, `qfoo` and `rfoo`, 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 et al. (2004) and not already present in base R, excluding the inverse Gaussian and log-t but including the loggamma distribution (Hogg and Klugman, 1984). Among others, most welcome additions are functions for the Pareto distribution.

Tables 1–3 list the supported distributions classified by families. Each table details the names of the distributions as given in Klugman et al. (2004), the root name of the `R` functions and the names of the arguments corresponding to each parameter in the parametrization of Klugman et al. (2004). One will note that by default all functions (except those for the Pareto distribution) use a rate parameter equal to the inverse of the scale parameter. This differs from Klugman et al. (2004) but is better in line with the functions for the gamma, exponential and Weibull distributions in base `R`.

All functions are written in C for speed. In almost every respect, they behave just like the base `R` functions.

In addition to the `d`, `p`, `q` and `r` functions, the package provides `m` and `lev` functions to compute the theoretical raw and limited moments, respectively. All the probability laws of Tables 1– 3 are supported, plus the following ones already in `R`: exponential, gamma, lognormal and Weibull. The `m` and `lev` functions come especially useful with estimation methods based on the matching of raw or limited moments. See Subsection 4.4 for their empirical counterparts.

4.2 Grouped data

Grouped data is data represented in an interval-frequency manner. Typically, a grouped data set will report that there where n_j claims in the interval $(c_{j-1}, c_j]$, $j = 1, \dots, r$. 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, to the point that the material on grouped data present in the first edition of Klugman et al. has been deleted in the second edition.

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

Table 1: Supported distributions from the Transformed Beta family, root name of the R functions and argument name corresponding to each parameter in the parametrization of Klugman et al. (2004).

Distribution name	Root (alias)	Arguments
Transformed beta	`trbeta` (`pearson6`)	`shape1` (α) `shape2` (γ) `shape3` (τ) `rate` ($\lambda = 1/\theta$) `scale` (θ)
Burr	`burr`	`shape1` (α) `shape2` (γ) `rate` ($\lambda = 1/\theta$) `scale` (θ)
Loglogistic	`llogis`	`shape` (γ) `rate` ($\lambda = 1/\theta$) `scale` (θ)
Paralogistic	`paralogis`	`shape` (α) `rate` ($\lambda = 1/\theta$) `scale` (θ)
Generalized Pareto	`genpareto`	`shape1` (α), `shape2` (τ), `rate` ($\lambda = 1/\theta$) `scale` (θ)
Pareto	`pareto` (`pareto2`)	`shape` (α) `scale` (θ)
Inverse Burr	`invburr`	`shape1` (τ) `shape2` (γ) `rate` ($\lambda = 1/\theta$) `scale` (θ)
Inverse Pareto	`invpareto`	`shape` (τ) `scale` (θ)
Inverse paralogistic	`invparalogis`	`shape` (τ) `rate` ($\lambda = 1/\theta$) `scale` (θ)

Table 2: Supported distributions from the Transformed Gamma family, root name of the `R` functions and argument name corresponding to each parameter in the parametrization of Klugman et al. (2004).

Distribution name	Root (alias)	Arguments
Transformed gamma	`trgamma`	`shape1` (α), `shape2` (τ) `rate` ($\lambda = 1/\theta$), `scale` (θ),
Inverse transformed gamma	`invtrgamma`	`shape1` (α), `shape2` (τ) `rate` ($\lambda = 1/\theta$), `scale` (θ),
Inverse gamma	`invgamma`	`shape` (α), `rate` ($\lambda = 1/\theta$), `scale` (θ)
Inverse Weibull	`invweibull` (`lgompertz`)	`shape` (τ), `rate` ($\lambda = 1/\theta$), `scale` (θ)
Inverse exponential	`invexp`	`rate` ($\lambda = 1/\theta$) `scale` (θ)

Table 3: Other supported distributions, root name of the `R` functions and argument name corresponding to each parameter in the parametrization of Klugman et al. (2004).

Distribution name	Root (alias)	Arguments
Loggamma[1]	`lgamma`	`shapelog` (α) `ratelog` (λ)
Single parameter Pareto	`pareto1`	`shape` (α) `min` (θ)
Generalized beta	`genbeta`	`shape1` (α) `shape2` (β) `shape3` (τ) `rate` ($\lambda = 1/\theta$) `scale` (θ)

[1] See Hogg and Klugman (1984).

with the c_js, omitting c_0 or not, etc. Moreover, with appropriate extraction, replacement and summary functions, manipulation of grouped data becomes similar to that of individual data.

First, function `grouped.data` creates a grouped data object similar to — an inheriting from — a data frame. The input of the function is a vector of group boundaries $c_0, c_1, \dots, c_r$ and one or more vectors of group frequencies $n_1, \dots, n_r$. 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

```
> 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))
> 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

> class(x)

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

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):

```
> 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):

```
> 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):

```
> 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:

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

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

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

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

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

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

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

```
> mean(x)

Line.1 Line.2
 188.0  108.2
```

Higher empirical moments can be computed with `emm`; see Subsection 4.4.

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):

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

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

$$F_n(x) = \frac{1}{n} \sum_{i=1}^{n} I\{x_j \leq x\}, \tag{2}$$

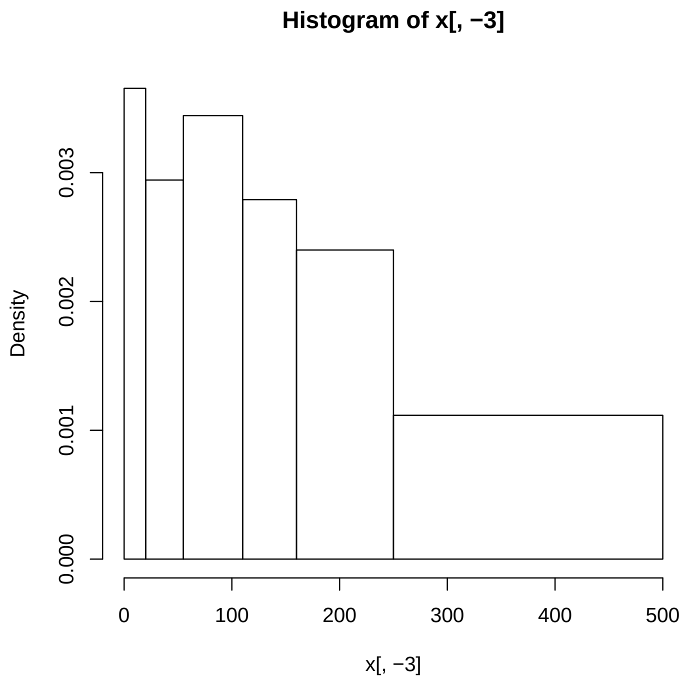

Figure 1: Histogram of a grouped data object

where $I\{\mathcal{A}\} = 1$ if $\mathcal{A}$ is true and $I\{\mathcal{A}\} = 0$ otherwise. The design and operation of `ecdf` is rather unusual, but very clever and handy: the function does not return a vector of probabilities as one would expect, but rather 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 et al., 1998; Hogg and Klugman, 1984). It is obtained by joining the known values of $F_n(x)$ at group boundaries with straight line segments:

$$\tilde{F}_n(x) = \begin{cases} 0, & x \leq c_0 \\ \dfrac{(c_j - x)F_n(c_{j-1}) + (x - c_{j-1})F_n(c_j)}{c_j - c_{j-1}}, & c_{j-1} < x \leq c_j \\ 1, & x > c_r. \end{cases} \tag{3}$$

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 $c_0, c_1, \dots, c_r$ of the ogive and a graph (see Figure 2):

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

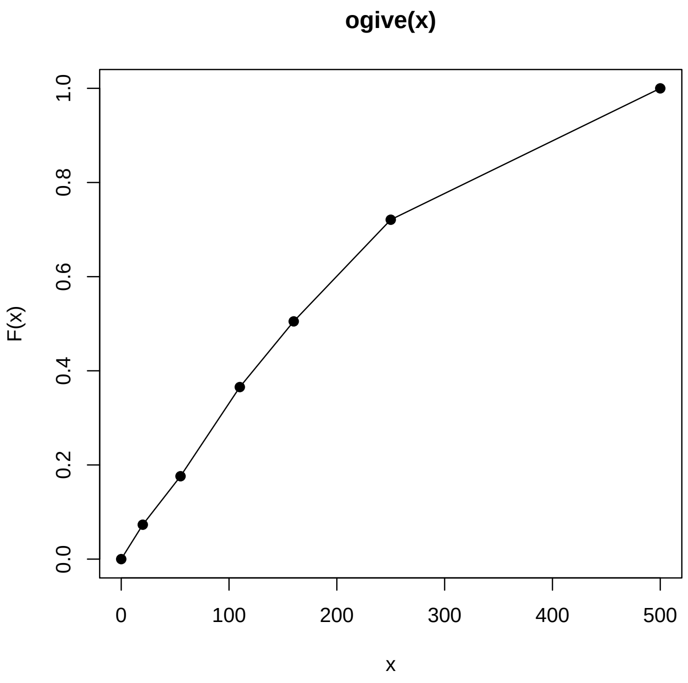

Figure 2: Ogive of a grouped data object

```
> knots(Fnt)

[1]   0  20  55 110 160 250 500

> Fnt(knots(Fnt))

[1] 0.00000 0.07309 0.17608 0.36545 0.50498 0.72093 1.00000

> plot(Fnt)
```

4.3 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):

```
> data(dental)
> dental

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

```
> data(gdental)
> gdental

              cj nj
1      (0,  25] 30
2    ( 25,  50] 31
3    ( 50, 100] 57
4    (100, 150] 42
5    (150, 250] 65
6    (250, 500] 84
7   (500, 1000] 45
8  (1000, 1500] 10
9  (1500, 2500] 11
10 (2500, 4000]  3
```

4.4 Calculation of empirical moments

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

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

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

> 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):

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

 [1]  16.0  37.6  42.4  85.1 105.5 164.5 187.7 197.9 241.1
[10] 335.5

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

 [1]   0.00  24.01  46.00  84.16 115.77 164.85 238.26 299.77
 [9] 324.90 347.39 353.34

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

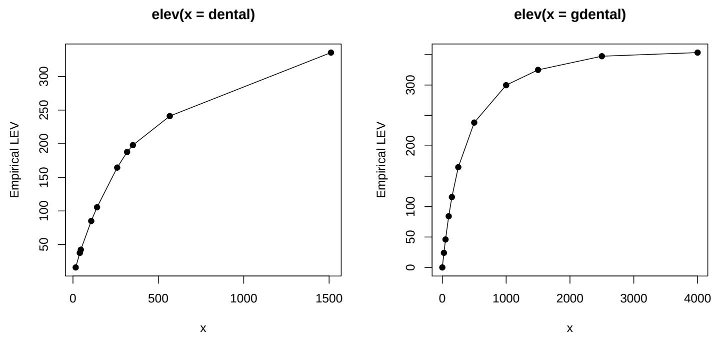

Figure 3: Empirical limited expected value function of an individual data object (left) and a grouped data object (right)

4.5 Minimum distance estimation

Maximum likelihood estimation (for individual data) is well covered by function `fitdistr` of package **MASS**. Package **actuar** provides function `mde`, very similar in usage and inner working to `fitdistr`, to fit models using three distance minimization techniques.

1. The Cramér-von Mises method (`CvM`) minimizes the squared difference between the theoretical cdf and the empirical cdf or ogive at their knots:
$$d(\theta) = \sum_{j=1}^{n} w_j (F(x_j; \theta) - F_n(x_j; \theta))^2 \tag{4}$$
for individual data and
$$d(\theta) = \sum_{j=1}^{r} w_j (F(c_j; \theta) - \tilde{F}_n(c_j; \theta))^2 \tag{5}$$
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 $w_1 \geq 0, w_2 \geq 0, \dots$ 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:
$$d(\theta) = \sum_{j=1}^{r} w_j [n(F(c_j; \theta) - F(c_{j-1}; \theta)) - n_j]^2, \tag{6}$$
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:

$$d(\theta) = \sum_{j=1}^{r} w_j (\mathrm{LAS}(c_{j-1}, c_j; \theta) - \tilde{\mathrm{LAS}}_n(c_{j-1}, c_j; \theta))^2, \qquad (7)$$

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

The arguments of `mde` are a data set, a function to compute $F(x)$ or $E[X \wedge x]$, 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):

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

    rate
  0.003551

 distance
  0.002842

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

    rate
  0.00364

 distance
     13.54

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

    rate
  0.002966

 distance
     694.5
```

It should be noted that optimization is not always that simple to achieve. For example, consider the problem of fitting a Pareto distribution to the same data set using the Cramér–von Mises method:

```
> mde(gdental, ppareto, start = list(shape = 3, scale = 600),
+     measure = "CvM")

Error in mde(gdental, ppareto, start = list(shape = 3, scale = 600),
             measure = "CvM") :
        optimization failed
```

Working in the log of the parameters often solves the problem since the optimization routine can then flawlessly work with negative parameter values:

```
> pparetolog <- function(x, logshape, logscale) ppareto(x,
+     exp(logshape), exp(logscale))
> (p <- mde(gdental, pparetolog, start = list(logshape = log(3),
+     logscale = log(600)), measure = "CvM"))

 logshape   logscale
    1.581      7.128

  distance
  0.0007905
```

The actual estimators of the parameters are obtained with

```
> exp(p$estimate)

logshape logscale
   4.861 1246.485
```

4.6 Coverage modifications

Let X be the random variable of the actual claim amount for an insurance policy and Y be the random variable of the amount of the claim as it appears in the insurer's database. These two random variables will differ if any of the following coverage modifications are present for the policy: an ordinary or a franchise deductible, a limit, coinsurance, inflation (see Klugman et al., 2004, Chapter 5 for precise definitions of these terms).

Often, one will want to use data $Y_1, \dots, Y_n$ from the random variable Y to fit a model on the unobservable random variable X. This requires to express the probability density function (pdf) or cdf of Y 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 `d` or `p` function.

For example, let Y 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 is

$$Y = \begin{cases} \text{undefined}, & X < d \\ X - d, & d \leq X \leq u \\ u - d, & X \geq u \end{cases} \tag{8}$$

and its pdf is

$$f_Y(y) = \begin{cases} 0, & y = 0 \\ \dfrac{f_X(y+d)}{1 - F_X(d)}, & 0 < y < u - d \\ \dfrac{1 - F_X(u)}{1 - F_X(d)}, & y = u - d \\ 0, & y > u - d. \end{cases} \tag{9}$$

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

```
> f <- coverage(dgamma, pgamma, deductible = 1, limit = 10)
> f(0, shape = 5, rate = 1)

[1] 0

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

[1] 0.1343

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

[1] 0.02936

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

[1] 0
```

See the "coverage" vignette for the detailed pdf and cdf formulas under various combinations of coverage modifications.

5 Risk theory

The current version of **actuar** addresses only one risk theory problem with two user visible functions: the calculation of the aggregate claim amount distribution of an insurance portfolio using the classical collective model of risk theory (Klugman et al., 2004; Gerber, 1979; Denuit and Charpentier, 2004; Kaas et al., 2001). That said, the package offers five different calculation or approximation methods of the distribution and four different techniques to discretize a continuous loss variable. Moreover, we feel the implementation described below makes R shine as a computing and modeling platform for such problems.

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

$$S = C_1 + \cdots + C_N, \tag{10}$$

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

$$\begin{aligned} F_S(x) &= \Pr[S \leq x] \\ &= \sum_{n=0}^{\infty} \Pr[S \leq x | N = n] p_n \\ &= \sum_{n=0}^{\infty} F_C^{*n}(x) p_n, \end{aligned} \tag{11}$$

where $F_C(x) = \Pr[C \leq x]$ is the common cdf of $C_1, \ldots, C_n$, $p_n = \Pr[N = n]$ and $F_C^{*n}(x) = \Pr[C_1 + \cdots + C_n \leq x]$ is the n-fold convolution of $F_C(\cdot)$. If C is discrete on $0, 1, 2, \ldots$, one has

$$F_C^{*k}(x) = \begin{cases} I\{x \geq 0\}, & k = 0 \\ F_C(x), & k = 1 \\ \sum_{y=0}^{x} F_C^{*(k-1)}(x-y) f_C(y), & k = 2, 3, \ldots \end{cases} \tag{12}$$

5.1 Discretization of claim amount distributions

Some numerical techniques to compute the aggregate claim amount distribution (see Subsection 5.2) 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), $E[X \wedge x]$ the limited expected value of X at x and f_x the probability mass at x in the discretized distribution. Currently, `discretize` supports the following four discretization methods.

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

   $$f_x = F(x + h) - F(x) \tag{13}$$

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

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

$$f_x = \begin{cases} F(a), & x = a \\ F(x) - F(x-h), & x = a+h, \dots, b. \end{cases} \tag{14}$$

   The discretized cdf is always under the true cdf.

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

$$f_x = \begin{cases} F(a+h/2), & x = a \\ F(x+h/2) - F(x-h/2), & x = a+h, \dots, b-h. \end{cases} \tag{15}$$

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

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

$$f_x = \begin{cases} \dfrac{E[X \wedge a] - E[X \wedge a+h]}{h} + 1 - F(a), & x = a \\ \dfrac{2E[X \wedge x] - E[X \wedge x-h] - E[X \wedge x+h]}{h}, & a < x < b \\ \dfrac{E[X \wedge b] - E[X \wedge b-h]}{h} - 1 + F(b), & x = b. \end{cases} \tag{16}$$

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

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,

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

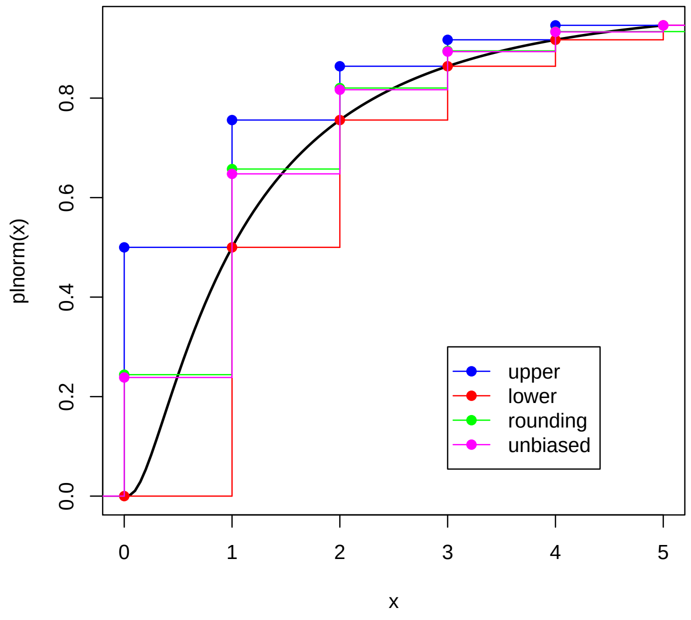

Figure 4: Comparison of four discretization methods

5.2 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 well known 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).

2. Exact calculation by numerical convolutions using (11) and (12). Hence, 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 (11), the convolutions being computed with `R`'s function `convolve`. This approach is practical for small problems only, even on today's fast computers.

3. Normal approximation of the cdf, that is
$$F_S(x) \approx \Phi\left(\frac{S - \mu_S}{\sigma_S}\right), \tag{17}$$
where $\mu_S = E[S]$ and $\sigma_S^2 = \mathrm{Var}[S]$. For most realistic models, this approximation is rather crude in the tails of the distribution.

4. Normal Power II approximation:
$$F(x) \approx \Phi\left(-\frac{3}{\gamma_S} + \sqrt{\frac{9}{\gamma_S^2} + 1 + \frac{6}{\gamma_S}\frac{x - \mu_S}{\sigma_S}}\right), \tag{18}$$
where $\gamma_S = E[(S - \mu_S)^3]/\sigma_S^{3/2}$. The approximation is valid for $x > \mu_S$ only and performs reasonably well when $\gamma_S < 1$. See Daykin et al. (1994) for details.

5. Simulation of a random sample from S and approximation of $F_S(x)$ by the empirical cdf (2). The simulation itself is done with function `simpf` (see Subsection 6.2). 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 $0, 1, 2, \dots$ assumed by the recursive and convolution methods and the true support of S.

Function `aggregateDist` returns a function object to compute the value of $F_S(x)$ in any x. Moreover, the package defines a few summary functions to extract information from this object.

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 a step of 0.5 and then use the recursive method:

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

```
> 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
 [12]  5.5  6.0  6.5  7.0  7.5  8.0  8.5  9.0  9.5 10.0 10.5
 [23] 11.0 11.5 12.0 12.5 13.0 13.5 14.0 14.5 15.0 15.5 16.0
 [34] 16.5 17.0 17.5 18.0 18.5 19.0 19.5 20.0 20.5 21.0 21.5
 [45] 22.0 22.5 23.0 23.5 24.0 24.5 25.0 25.5 26.0 26.5 27.0
 [56] 27.5 28.0 28.5 29.0 29.5 30.0 30.5 31.0 31.5 32.0 32.5
 [67] 33.0 33.5 34.0 34.5 35.0 35.5 36.0 36.5 37.0 37.5 38.0
 [78] 38.5 39.0 39.5 40.0 40.5 41.0 41.5 42.0 42.5 43.0 43.5
 [89] 44.0 44.5 45.0 45.5 46.0 46.5 47.0 47.5 48.0 48.5 49.0
[100] 49.5 50.0 50.5 51.0 51.5 52.0 52.5 53.0 53.5 54.0 54.5
[111] 55.0 55.5 56.0 56.5 57.0 57.5 58.0 58.5 59.0 59.5 60.0
[122] 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 `plot` (see Figure 5):

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

Finally, one can easily compute the mean and obtain the quantiles of the approximate distribution as follows:

```
> mean(Fs)
```

```
[1] 20
```

```
> 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
```

```
> quantile(Fs, 0.999)
```

```
99.9%
 49.5
```

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

6 Credibility theory

The credibility theory facilities of **actuar** consist of one data set and three main functions:

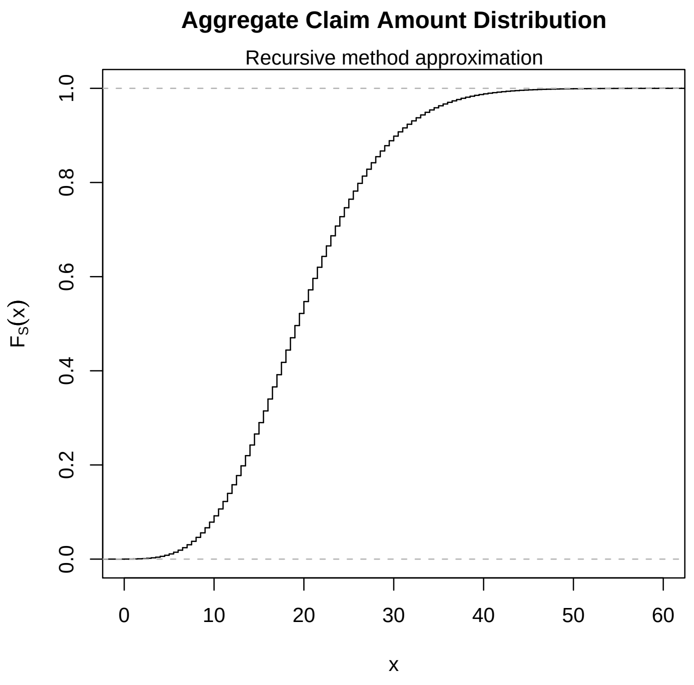

Figure 5: Graphic of the empirical cdf of S obtained with the recursive method

1. matrix `hachemeister` containing the famous data set of Hachemeister (1975);
2. function `simpf` to simulate data from compound hierarchical models;
3. function `cm` to fit linear hierarchical credibility models;
4. function `bstraub`, a faster and simpler version of `cm` to fit Bühlmann and Bühlmann–Straub models.

6.1 Hachemeister data set

The data set of Hachemeister (1975) consists of average claim amounts for private passenger bodily injury insurance for five U.S. states over 12 quarters between July 1970 and June 1973 and the corresponding number of claims. 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, columns 2–13 contain the claim averages and columns 14–25 contain the claim numbers:

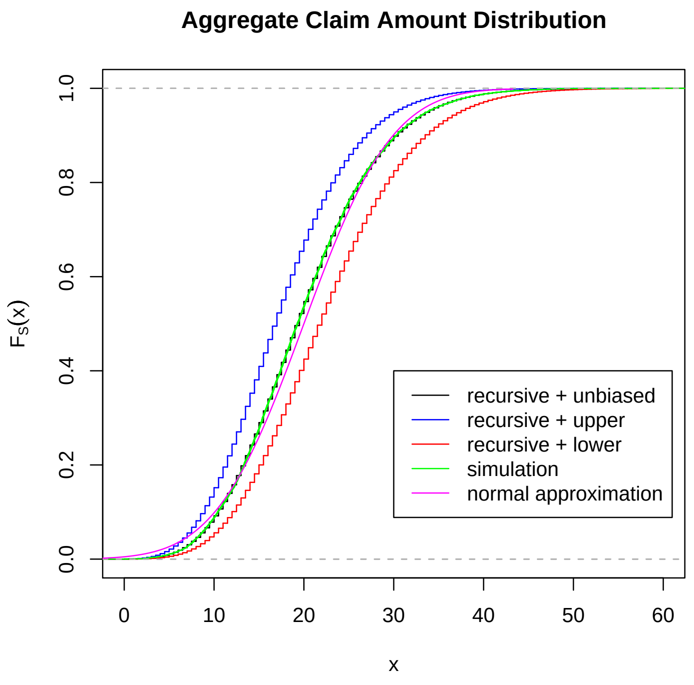

Figure 6: Comparison between the empirical or approximate cdf of S obtained with five different methods

```
> data(hachemeister)
> hachemeister
```

```
     state ratio.1 ratio.2 ratio.3 ratio.4 ratio.5 ratio.6
[1,]     1    1738    1642    1794    2051    2079    2234
[2,]     2    1364    1408    1597    1444    1342    1675
[3,]     3    1759    1685    1479    1763    1674    2103
[4,]     4    1223    1146    1010    1257    1426    1532
[5,]     5    1456    1499    1609    1741    1482    1572
     ratio.7 ratio.8 ratio.9 ratio.10 ratio.11 ratio.12
[1,]    2032    2035    2115     2262     2267     2517
[2,]    1470    1448    1464     1831     1612     1471
[3,]    1502    1622    1828     2155     2233     2059
[4,]    1953    1123    1343     1243     1762     1306
[5,]    1606    1735    1607     1573     1613     1690
     weight.1 weight.2 weight.3 weight.4 weight.5 weight.6
[1,]     7861     9251     8706     8575     7917     8263
[2,]     1622     1742     1523     1515     1622     1602
[3,]     1147     1357     1329     1204      998     1077
```

```
[4,]       407       396       348       341       315       328
[5,]      2902      3172      3046      3068      2693      2910
     weight.7 weight.8 weight.9 weight.10 weight.11 weight.12
[1,]     9456     8003     7365      7832      7849      9077
[2,]     1964     1515     1527      1748      1654      1861
[3,]     1277     1218      896      1003      1108      1121
[4,]      352      331      287       384       321       342
[5,]     3275     2697     2663      3017      3242      3425
```

6.2 Portfolio simulation

Function `simpf` simulates portfolios of data following compound models of the form (10) 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 three-level compound hierarchical model:

$$S_{ijt} = C_{ijt1} + \cdots + C_{ijtN_{ijt}}, \tag{19}$$

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

$$\begin{aligned} N_{ijt}|\Lambda_{ij}, \Phi_i &\sim \text{Poisson}(w_{ijt}\Lambda_{ij}) & C_{ijtu}|\Theta_{ij}, \Psi_i &\sim \text{Lognormal}(\Theta_{ij}, 1) \\ \Lambda_{ij}|\Phi_i &\sim \text{Gamma}(\Phi_i, 1) & \Theta_{ij}|\Psi_i &\sim N(\Psi_i, 1) \\ \Phi_i &\sim \text{Exponential}(2) & \Psi_i &\sim N(2, 0.1). \end{aligned} \tag{20}$$

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.

Function `simpf` is presented in the credibility theory section because it was originally written in this context, but it has much wider applications. For instance, as mentioned in Subsection 5.2, it is used by `aggregateDist` for the approximation of the cdf of S by simulation.

Goulet and Pouliot (2007) describe in detail the model specification method used in `simpf`. 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, $J_1, \dots, J_I$ and $n_{11}, \dots, n_{IJ}$, above) and by the probability laws at each level. The number of nodes is passed to `simpf` 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. Now, 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 `simpf` also supports usage of weights, or volumes, in models. These usually modify the frequency parameters to take into account the "size" of an insurance contract. The weights will be used in simulation wherever the name `weights` appears in a model.

Hence, function `simpf` 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 contract 1, then all weights of contract 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 (20) above, and that weights are simply simulated from a uniform distribution on $(0.5, 2.5)$, then simulation of a data set with `simpf` is achieved with:

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

The function returns the variates in a two-dimension list containing all the individual claim amounts for each contract. Such an 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 contract in the portfolio (subscripts i and j in the notation above). Displaying the complete content of the object returned by `simpf` can be impractical. For this reason, only the simulation model and the number of claims in each node is printed:

```
> pf

Portfolio of claim amounts

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

```
 Number of claims per node:

     class contract 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 contains three functions to easily compute the matrices of variates of the aggregate random variable S and variates of the frequency random variable N, and to rearrange the list returned by `simpf` into a matrix of the variates of the severity random variables C_j. These are, respectively, `aggregate`, `frequency` and `severity`. In addition, function `weights` extracts the weights matrix from a simulated data set.

By default, `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)

     class contract 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
```

```
> aggregate(pf, by = c("class", "year"), FUN = mean)

     class 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
```

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

```
> frequency(pf)

     class contract 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
```

```
> frequency(pf, by = "class")

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

Finally, function `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 contract,

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

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 function provides an argument `splitcol` in case one would like to extract separately the individual severities of one or more periods:

```
> severity(pf)

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

$last
NULL
```

```
> severity(pf, splitcol = 1)

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

$last
     class 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 of the portfolio is

```
> weights(pf)

     class contract 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
```

Function `simpf` was used to simulate the data in Forgues et al. (2006).

6.3 Fitting of hierarchical credibility models

The linear model fitting function of base `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`.

We hope that, in the long term, `cm` can act as a unified interface for most credibility models. Currently, it supports the models of Bühlmann (1969) and Bühlmann and Straub (1970), as well as the hierarchical model of Jewell (1975). The last model includes the first two as special cases. (According to our current counting scheme, a Bühlmann-Straub model is a two-level hierarchical model.)

There are some variations in the formulas of the hierarchical model in the literature. We estimate the structure parameters as indicated in Goovaerts and Hoogstad (1987) but compute the credibility premiums as given in Bühlmann and Jewell (1987) or Bühlmann and Gisler (2005); Goulet (1998) has all the appropriate formulas for our implementation. For instance, for a three-level hierarchical model like (19)-(20), the best linear prediction of the ratio $X_{ij,n_{ij}+1} = S_{ij,n_{ij}+1}/w_{ij,n_{ij}+1}$ is

$$\begin{aligned} \hat{\pi}_{ij} &= z_{ij} X_{ijw} + (1 - z_{ij}) \hat{\pi}_i \\ \hat{\pi}_i &= z_i X_{izw} + (1 - z_i) m \end{aligned} \tag{21}$$

with

$$z_{ij} = \frac{w_{ij\Sigma}}{w_{ij\Sigma} + s^2/a}, \qquad X_{ijw} = \sum_{t=1}^{n_{ij}} \frac{w_{ijt}}{w_{ij\Sigma}} X_{ijt}$$

$$z_i = \frac{z_{i\Sigma}}{z_{i\Sigma} + a/b}, \qquad X_{izw} = \sum_{j=1}^{J_i} \frac{z_{ij}}{z_{i\Sigma}} X_{ijw}.$$

The (pseudo-)estimators of the structure parameters s^2, a, b and m are, respectively,

$$\hat{s}^2 = \frac{1}{\sum_{i=1}^{I} \sum_{j=1}^{J_i} (n_{ij} - 1)} \sum_{i=1}^{I} \sum_{j=1}^{J_i} \sum_{t=1}^{n_{ij}} w_{ijt} (X_{ijt} - X_{ijw})^2$$

$$\hat{a} = \frac{1}{\sum_{i=1}^{I} (J_i - 1)} \sum_{i=1}^{I} \sum_{j=1}^{J_i} z_{ij} (X_{ijw} - X_{izw})^2$$

$$\hat{b} = \frac{1}{I - 1} \sum_{i=1}^{I} z_i (X_{izw} - X_{zzw})^2$$

and

$$\hat{m} = X_{zzw} = \sum_{i=1}^{I} \frac{z_i}{z_\Sigma} X_{izw}.$$

Function `cm` takes in argument a formula describing the hierarchical interactions in the data set, a data set containing the variables referenced in the formula and the names of the columns where the ratios and the weights are to be found in the data set. The latter should be a matrix or a data frame with one column of indexes (numeric or character) for each hierarchical interaction, at least two nodes in each level and more than one period of experience for at least one contract. Missing values are represented by `NA`s. There can be contracts with no experience (complete lines of `NA`s).

The function returns a fitted model object containing the estimators of the structure parameters. To compute the credibility premiums, one calls function `predict` with the said object in argument. One can also obtain a nicely formated view of the most important results with a call to `summary`. These two functions can report for the whole portfolio or for a subset of levels only by means of an argument `"levels"`.

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

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

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

Structure Parameters Estimators

  Collective premium: 1746
  Between class variance: 88981
  Within class/Between state variance: 10952
  Within state variance: 139120026
```

The key results of this fit are obtained with `summary`:

```
> summary(fit)

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

Structure Parameters Estimators

  Collective premium: 1746
  Between class variance: 88981
  Within class/Between state variance: 10952
  Within state variance: 139120026

Detailed premiums
```

```
  Level: class
    class Ind. premium Weight Cred. factor Cred. premium
        1         1967  1.407       0.9196          1949
        2         1528  1.596       0.9284          1543

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

Function `predict` returns only the credibility premiums per level:

```
> predict(fit)

$class
[1] 1949 1543

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

Finally, one can obtain the results above for the class level only as follows:

```
> summary(fit, levels = "class")

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

Structure Parameters Estimators

  Collective premium: 1746
  Between class variance: 88981
  Within class variance: 10952

Detailed premiums

  Level: class
    class Ind. premium Weight Cred. factor Cred. premium
        1         1967  1.407       0.9196          1949
        2         1528  1.596       0.9284          1543
```

```
> predict(fit, levels = "class")

$class
[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.

6.4 Bühlmann and Bühlmann–Straub models

Function `bstraub` is a faster and simpler alternative to `cm` for Bühlmann and Bühlmann–Straub models. There is no formula argument in this function and data is passed by means of a matrix of ratios and a matrix of weights. Furthermore, this function has the option to compute the unbiased estimator of the between variance

$$\hat{a} = \frac{w_{\Sigma\Sigma}}{w_{\Sigma\Sigma}^2 - \sum_{i=1}^I w_{i\Sigma}^2} \left(\sum_{i=1}^I w_{i\Sigma}(X_{iw} - X_{ww})^2 - (I-1)\hat{s}^2 \right).$$

Otherwise, usage of `summary` and `predict` for models fitted with `bstraub` is identical to models fitted with `cm`.

7 Conclusion

The paper presented the facilities of the R package **actuar** version 0.9-3 in the fields of loss distribution modeling, risk theory and credibility theory. We feel this version of the package covers most of the basics needs in these areas. In the future we plan to improve the functions currently available — especially speed wise — but also to start adding more advanced features. For example, future versions of the package should include support for dependence models in risk theory and regression credibility models.

Obviously, the package left many other fields of Actuarial Science untouched so far. 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. This project intends to continue to grow and 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

```
> citation()
```

or

```
> 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 the following students: Sébastien Auclair, Mathieu Pigeon, 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.

References

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.

Bühlmann, H., Jewell, W. S., 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.

Daykin, C., Pentikäinen, T., Pesonen, M., 1994. Practical Risk Theory for Actuaries. Chapman & Hall, London.

Denuit, M., Charpentier, A., 2004. Mathématiques de l'assurance non-vie. Vol. 1, Principes fondamentaux de théorie du risque. Economica, Paris.

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

Gerber, H. U., 1979. An Introduction to Mathematical Risk Theory. Huebner Foundation, Philadelphia.

Goovaerts, M. J., Hoogstad, W. J., 1987. Credibility theory. No. 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., 2007. actuar: An R Package for Actuarial Science, version 0.9-3. École d'actuariat, Université Laval.
URL `http://www.actuar-project.org`

Goulet, V., Pouliot, L.-P., 2007. Simulation of hierarchical models in R. Journal of Statistical Software Submitted for publication.

Hachemeister, C. A., 1975. Credibility for regression models with application to trend. In: Credibility, theory and applications. Proceedings of the Berkeley actuarial research conference on credibility. Academic Press, New York.

Hogg, R. V., Klugman, S. A., 1984. Loss Distributions. Wiley, New York.

Jewell, W. S., 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.

Klugman, S. A., Panjer, H. H., Willmot, G., 1998. Loss Models: From Data to Decisions. Wiley, New York.

Klugman, S. A., Panjer, H. H., Willmot, G., 2004. Loss Models: From Data to Decisions, 2nd Edition. Wiley, New York.

Panjer, H. H., 1981. Recursive evaluation of a family of compound distributions. Astin Bulletin 12, 22–26.

R Development Core Team, 2007. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL `http://www.r-project.org`

Vincent Goulet
École d'actuariat
Pavillon Alexandre-Vachon, bureau 1620
Université Laval
Québec (QC) G1K 7P4
Canada
E-mail: `vincent.goulet@act.ulaval.ca`
URL: `http://vgoulet.act.ulaval.ca/`