Risk and ruin theory features of **actuar**

Christophe Dutang
Université du Mans

Vincent Goulet
Université Laval

Mathieu Pigeon
Université du Québec à Montréal

1 Introduction

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 read more on these subjects in Klugman et al. (2012); Gerber (1979); Denuit and Charpentier (2004); Kaas et al. (2008), among others.

The current version of **actuar** (Dutang et al., 2008) 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.

2 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

$$S = C_1 + \cdots + C_N, \tag{1}$$

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

$$\begin{aligned} F_S(x) &= \Pr[S \leq x] \\ &= \sum_{n=0}^{\infty} \Pr[S \leq x | N = n] p_n \\ &= \sum_{n=0}^{\infty} F_C^{*n}(x) p_n, \end{aligned} \tag{2}$$

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

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

where $I\{\mathcal{A}\} = 1$ if $\mathcal{A}$ is true and $I\{\mathcal{A}\} = 0$ otherwise.

3 Discretization of claim amount distributions

Some numerical techniques to compute the aggregate claim amount distribution (see section 4) 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 $F(x)$:

$$f_x = F(x + h) - F(x) \tag{4}$$

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

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

$$f_x = \begin{cases} F(a), & x = a \\ F(x) - F(x - h), & x = a + h, \ldots, b. \end{cases} \tag{5}$$

   The discretized cdf is always under the true cdf.

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

$$f_x = \begin{cases} F(a+h/2), & x = a \\ F(x+h/2) - F(x-h/2), & x = a+h, \ldots, b-h. \end{cases} \tag{6}$$

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

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

$$f_x = \begin{cases} \dfrac{E[X \wedge a] - E[X \wedge a+h]}{h} + 1 - F(a), & x = a \\ \dfrac{2E[X \wedge x] - E[X \wedge x-h] - E[X \wedge x+h]}{h}, & a < x < b \\ \dfrac{E[X \wedge b] - E[X \wedge b-h]}{h} - 1 + F(b), & x = b. \end{cases} \tag{7}$$

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

Figure 1 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.

4 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

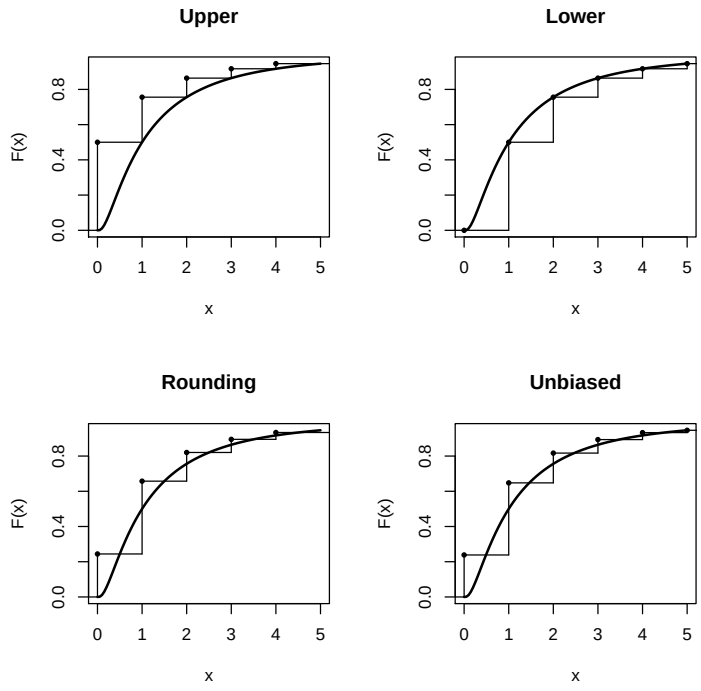

Figure 1: Comparison of four discretization methods

monetary unit and the frequency distribution to be a member of either the $(a, b, 0)$ or $(a, b, 1)$ class of distributions (Klugman et al., 2012). (These classes contain the Poisson, binomial, negative binomial and logarithmic distributions and their zero-truncated and zero-modified extensions allowing for a zero or 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 (2) and (3). 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 (2), 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

$$F_S(x) \approx \Phi\left(\frac{x-\mu_S}{\sigma_S}\right), \tag{8}$$

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 of the cdf, that is

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

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

$$F_n(x) = \frac{1}{n}\sum_{j=1}^{n} I\{x_j \le x\}. \tag{10}$$

The simulation itself is done with function `simul` (see the `"simulation"` vignette). 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 according to the chosen 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 recursive method fails when the expected number of claims is so large that $f_S(0)$ is numerically equal to zero. One solution proposed by Klugman et al. (2012) consists in dividing the appropriate parameter of the frequency distribution by 2^n, with n such that $f_S(0) > 0$ and the recursions can start. One then computes the aggregate claim amount distribution using the recursive method and then convolves the resulting distribution n times with itself to obtain the final distribution. Function `aggregateDist` supports this procedure through its argument `convolve`.

A common problem with the recursive method is failure to obtain a cumulative distribution function that reaching (close to) 1. This is usually due to too coarse a discretization of the severity distribution. One should make sure to use a small enough discretization step and to discretize the severity distribution far in the right tail.

The function `aggregateDist` 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`:

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

Although useless here, the following is essentially equivalent, except in the far right tail for numerical reasons:

```
> Fsc <- aggregateDist("recursive", model.freq = "poisson",
+                      model.sev = fx, lambda = 5, convolve = 1,
+                      x.scale = 0.5)
> summary(Fsc)                             # summary method
Aggregate Claim Amount Empirical CDF:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0.0    14.5    19.5    20.0    25.0   103.0
```

We return to object `Fs`. It contains an empirical cdf with support

```
> knots(Fs)                        # support of Fs.b (knots)
  [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
```

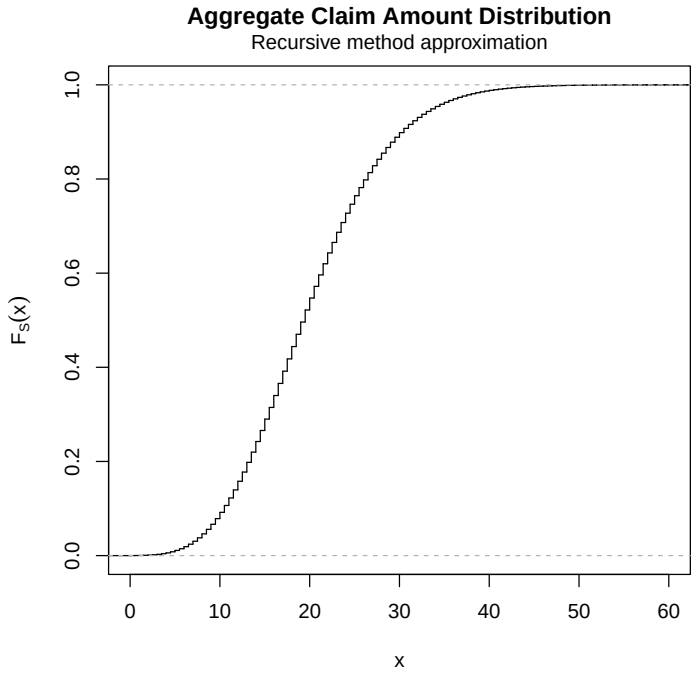

Figure 2: Graphic of the empirical cdf of S obtained with the recursive method

```
[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 a method of `plot` (see Figure 2):

```
> 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:

```
> mean(Fs)                                # empirical mean
[1] 20
> quantile(Fs)                            # quantiles
  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)                     # quantiles
```

```
99.9%
 49.5
```

Second, a method of `diff` gives easy access to the underlying probability mass function:

```
> diff(Fs)
  [1] 6.293e-05 8.934e-05 1.767e-04 2.954e-04 4.604e-04
  [6] 6.811e-04 9.662e-04 1.324e-03 1.760e-03 2.282e-03
 [11] 2.893e-03 3.594e-03 4.387e-03 5.269e-03 6.235e-03
 [16] 7.280e-03 8.395e-03 9.570e-03 1.079e-02 1.205e-02
 [21] 1.333e-02 1.462e-02 1.590e-02 1.715e-02 1.837e-02
 [26] 1.953e-02 2.063e-02 2.166e-02 2.259e-02 2.343e-02
 [31] 2.417e-02 2.479e-02 2.531e-02 2.570e-02 2.598e-02
 [36] 2.614e-02 2.618e-02 2.612e-02 2.594e-02 2.567e-02
 [41] 2.530e-02 2.484e-02 2.431e-02 2.370e-02 2.303e-02
 [46] 2.230e-02 2.153e-02 2.072e-02 1.988e-02 1.901e-02
 [51] 1.813e-02 1.725e-02 1.636e-02 1.547e-02 1.460e-02
 [56] 1.374e-02 1.290e-02 1.208e-02 1.128e-02 1.052e-02
 [61] 9.780e-03 9.074e-03 8.401e-03 7.761e-03 7.155e-03
 [66] 6.583e-03 6.044e-03 5.538e-03 5.065e-03 4.623e-03
 [71] 4.213e-03 3.831e-03 3.478e-03 3.152e-03 2.851e-03
 [76] 2.575e-03 2.321e-03 2.090e-03 1.878e-03 1.685e-03
 [81] 1.509e-03 1.350e-03 1.205e-03 1.075e-03 9.571e-04
 [86] 8.510e-04 7.556e-04 6.699e-04 5.931e-04 5.244e-04
 [91] 4.630e-04 4.083e-04 3.596e-04 3.162e-04 2.778e-04
 [96] 2.437e-04 2.135e-04 1.869e-04 1.633e-04 1.426e-04
[101] 1.243e-04 1.083e-04 9.423e-05 8.189e-05 7.108e-05
[106] 6.164e-05 5.339e-05 4.619e-05 3.993e-05 3.447e-05
[111] 2.973e-05 2.562e-05 2.205e-05 1.896e-05 1.629e-05
[116] 1.398e-05 1.199e-05 1.027e-05 8.786e-06 7.512e-06
[121] 6.416e-06 5.476e-06 4.668e-06 3.977e-06 3.385e-06
[126] 2.878e-06 2.445e-06 2.076e-06 1.761e-06 1.492e-06
[131] 1.263e-06 1.069e-06 9.035e-07 7.632e-07 6.441e-07
[136] 5.432e-07 4.577e-07 3.854e-07 3.243e-07 2.726e-07
[141] 2.290e-07 1.923e-07 1.613e-07
```

Of course, this is defined (and makes sense) for the recursive, direct convolution and simulation methods only.

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

$$\Pr[S \leq \text{VaR}_\alpha] = \alpha, \tag{11}$$

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

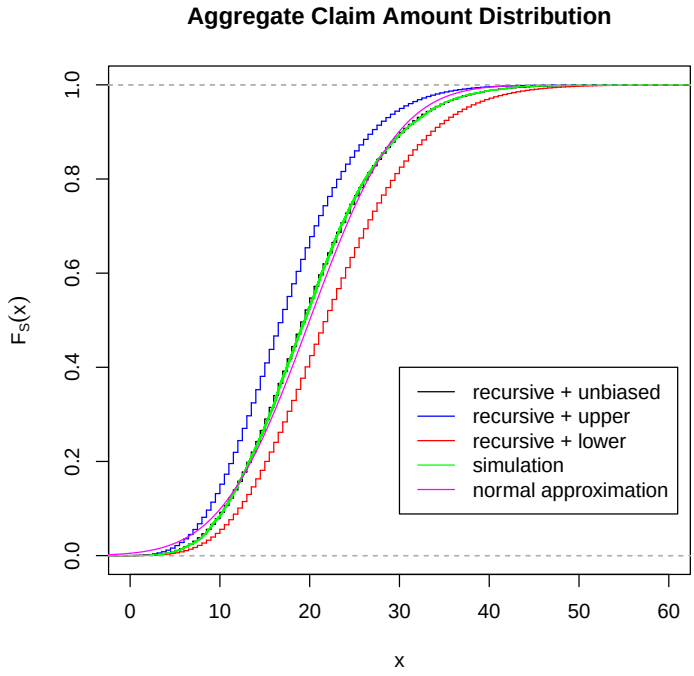

Figure 3: Comparison between the empirical or approximate cdf of S obtained with five different methods

(also called Tail Value-at-Risk)

$$\text{CTE}_\alpha = E[S|S > \text{VaR}_\alpha]. \tag{12}$$

Here are examples using object `Fs` obtained above:

```
> VaR(Fs)
 90%  95%  99%
30.5 34.0 41.0
> CTE(Fs)
  90%   95%   99%
35.42 38.55 45.01
```

To conclude on the subject, Figure 3 shows the cdf of S using five of the many combinations of discretization and calculation method supported by **actuar**.

5 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

$$U(t) = u + c(t) - S(t). \tag{13}$$

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

$$\psi(u) = \Pr[U(t) < 0 \text{ for some } t \geq 0]. \tag{14}$$

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 (1):

$$S(t) = C_1 + \cdots + C_{N(t)}, \tag{15}$$

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

$$W_j = T_j - T_{j-1}, \quad j \geq 2. \tag{16}$$

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.

6 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

$$h(t) = E[e^{tC - tcW}] = 1, \tag{17}$$

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

$$h(t) = M_C(t)M_W(-tc) = 1. \tag{18}$$

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

```
> 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

$$h(t) = E[e^{t\alpha C - tc(\alpha)W}] = 1. \tag{19}$$

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

$$h(t) = E[e^{t\min(C,L) - tc(L)W}] = 1. \tag{20}$$

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 4 for the graph):

```
> mgfx <- function(x, y) mgfexp(x * y)
> p <- function(x) 2.6 * x - 0.2
> rho <- adjCoef(mgfx, mgfexp(x, 2), premium = p, upper = 1,
+                reins = "prop", from = 0, to = 1)
> rho(c(0.75, 0.8, 0.9, 1))
[1] 0.1905 0.1862 0.1765 0.1667
> plot(rho)
```

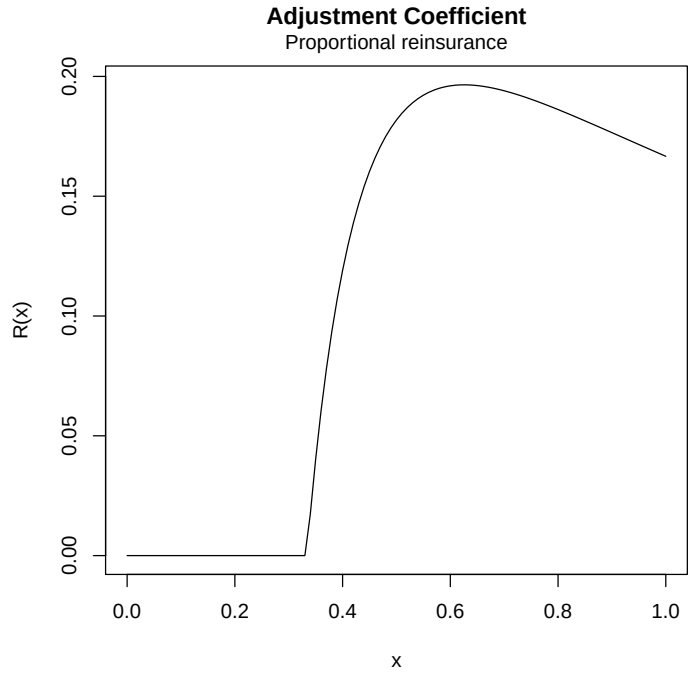

Figure 4: Adjustment coefficient as a function of the retention rate

7 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

$$\psi(u) = \frac{\lambda}{c\beta}\, e^{-(\beta-\lambda/c)u}. \tag{21}$$

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

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

$$\begin{aligned} \boldsymbol{\pi}_+ &= -\frac{\lambda}{c}\,\boldsymbol{\pi}\boldsymbol{T}^{-1} \\ \boldsymbol{Q} &= \boldsymbol{T} + \boldsymbol{t}\boldsymbol{\pi}_+, \end{aligned} \tag{22}$$

and $\boldsymbol{t} = -\boldsymbol{T}\boldsymbol{e}$, $\boldsymbol{e}$ is a column vector with all components equal to 1; see the `"lossdist"` vignette for details.

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

$$\boldsymbol{\pi}_+ = \frac{\boldsymbol{e}'(\boldsymbol{Q} - \boldsymbol{T})}{c\boldsymbol{e}'\boldsymbol{t}} \tag{23}$$

and $\boldsymbol{Q}$ solution of

$$\begin{aligned} \boldsymbol{Q} &= \Psi(\boldsymbol{Q}) \\ &= \boldsymbol{T} - \boldsymbol{t}\boldsymbol{\pi}\left[(\boldsymbol{I}_n \otimes \boldsymbol{\nu})(\boldsymbol{Q} \oplus \boldsymbol{S})^{-1}(\boldsymbol{I}_n \otimes \boldsymbol{s})\right]. \end{aligned} \tag{24}$$

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

$$\boldsymbol{A}_{m\times m} \oplus \boldsymbol{B}_{n\times n} = \boldsymbol{A} \otimes \boldsymbol{I}_n + \boldsymbol{B} \otimes \boldsymbol{I}_m. \tag{25}$$

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 (21) 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

```
> psi <- ruin(claims = "e", par.claims = list(rate = 5),
+             wait   = "e", par.wait   = list(rate = 3))
> 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: 0x7f855a885f48>
attr(,"class")
[1] "ruin"     "function"
> psi(0:10)
 [1] 6.000e-01 8.120e-02 1.099e-02 1.487e-03 2.013e-04
 [6] 2.724e-05 3.687e-06 4.989e-07 6.752e-08 9.138e-09
[11] 1.237e-09
```

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

```
> 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: 0x7f855c9ac6d8>
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

```
> prob <- c(0.5614, 0.4386)
> rates <- matrix(c(-8.64, 0.101, 1.997, -1.095), 2, 2)
> 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: 0x7f855bc41470>
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

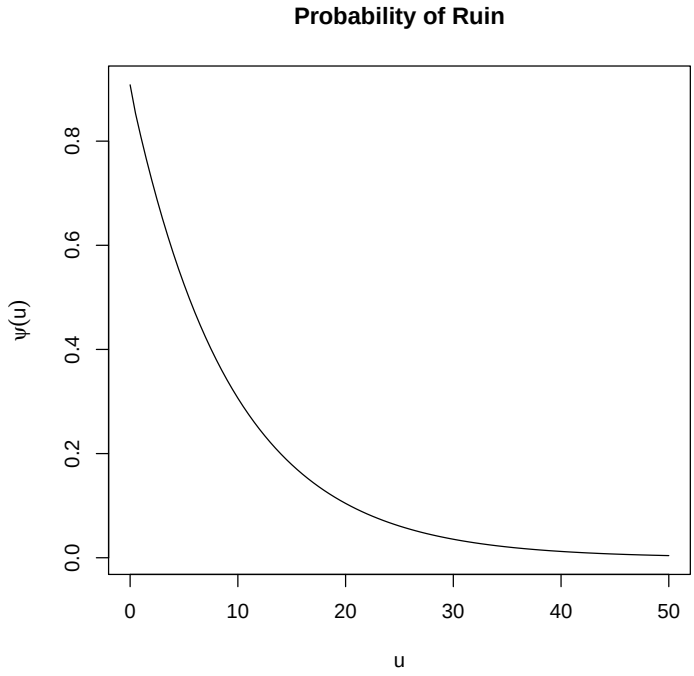

Figure 5: Graphic of the probability of ruin as a function of the initial surplus u

`curve` (see Figure 5):

```
> 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)))
> plot(psi, from = 0, to = 50)
```

8 Approximation to the probability of ruin

When the model for the aggregate claim process (15) does not fit nicely into the framework of the previous section, one can compute ruin probabilities using the so-called Beekman's convolution formula (Beekman, 1968; Kass, 2004).

Let the surplus process and the aggregate claim amount process be defined as in (13) and (15), respectively, and let $\{N(t)\}$ be a Poisson process with mean λ. As before, claim amounts $C_1, C_2, \ldots$ are independent and identically distributed with cdf $P(\cdot)$ and mean $\mu = E[C_1]$. Then the infinite time

probability of ruin is given by

$$\psi(u) = 1 - F(u), \tag{26}$$

where $F(\cdot)$ is Compound Geometric(p, H) with

$$p = 1 - \frac{\lambda\mu}{c} \tag{27}$$

and

$$H(x) = \int_0^x \frac{1 - P(y)}{\mu}\, dy. \tag{28}$$

In other words, we have (compare with (2)):

$$\psi(u) = 1 - \sum_{n=0}^{\infty} H^{*n}(u) p(1-p)^n. \tag{29}$$

In most practical situations, numerical evaluation of (29) is done using Panjer's recursive formula. This usually requires discretization of $H(\cdot)$. In such circumstances, Beekman's formula yields approximate ruin probabilities.

For example, let claim amounts have a Pareto$(5, 4)$ distribution, that is

$$P(x) = 1 - \left(\frac{4}{4+x}\right)^5$$

and $\mu = 1$. Then

$$\begin{aligned} H(x) &= \int_0^x \left(\frac{4}{4+y}\right)^5 dy \\ &= 1 - \left(\frac{4}{4+x}\right)^4, \end{aligned}$$

or else H is Pareto$(4, 4)$. Furthermore, we determine the premium rate c with the expected value premium principle and a safety loading of 20%, that is $c = 1.2\lambda\mu$. Thus, $p = 0.2/1.2 = 1/6$.

One can get functions to compute lower bounds and upper bounds for $F(u)$ with functions `discretize` and `aggregateDist` as follows:

```
> f.L <- discretize(ppareto(x, 4, 4), from = 0, to = 200,
+                   step = 1, method = "lower")
> f.U <- discretize(ppareto(x, 4, 4), from = 0, to = 200,
+                   step = 1, method = "upper")
> F.L <- aggregateDist(method = "recursive",
+                      model.freq = "geometric",
+                      model.sev = f.L, prob = 1/6)
> F.U <- aggregateDist(method = "recursive",
+                      model.freq = "geometric",
+                      model.sev = f.U, prob = 1/6)
```

Corresponding functions for the probability of ruin $\psi(u)$ lower and upper bounds are (see Figure 6 for the graphic):

```
> psi.L <- function(u) 1 - F.U(u)
> psi.U <- function(u) 1 - F.L(u)
> u <- seq(0, 50, by = 5)
> cbind(lower = psi.L(u), upper = psi.U(u))
          lower   upper
 [1,] 0.6719160 0.83333
 [2,] 0.2892792 0.51572
 [3,] 0.1361541 0.32938
 [4,] 0.0662486 0.21200
 [5,] 0.0329848 0.13700
 [6,] 0.0167551 0.08877
 [7,] 0.0086802 0.05764
 [8,] 0.0045911 0.03749
 [9,] 0.0024843 0.02443
[10,] 0.0013790 0.01595
[11,] 0.0007877 0.01043
> curve(psi.L, from = 0, to = 100, col = "blue")
> curve(psi.U, add = TRUE, col = "green")
```

One can make the bounds as close as one wishes by reducing the discretization step.

References

S. Asmussen and T. Rolski. Computational methods in risk theory: a matrix-algorithmic approach. *Insurance: Mathematics and Economics*, 10:259–274, 1991.

J. A. Beekman. Collective risk results. *Transactions of the Society of Actuaries*, 20:182–199, 1968.

M. d. L. Centeno. Measuring the effects of reinsurance by the adjustment coefficient in the sparre-anderson model. *Insurance: Mathematics and Economics*, 30:37–49, 2002.

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

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

C Dutang, V. Goulet, and M. Pigeon. **actuar**: An R package for actuarial science. *Journal of Statistical Software*, 25(7), 2008. URL http://www.actuar-project.org.

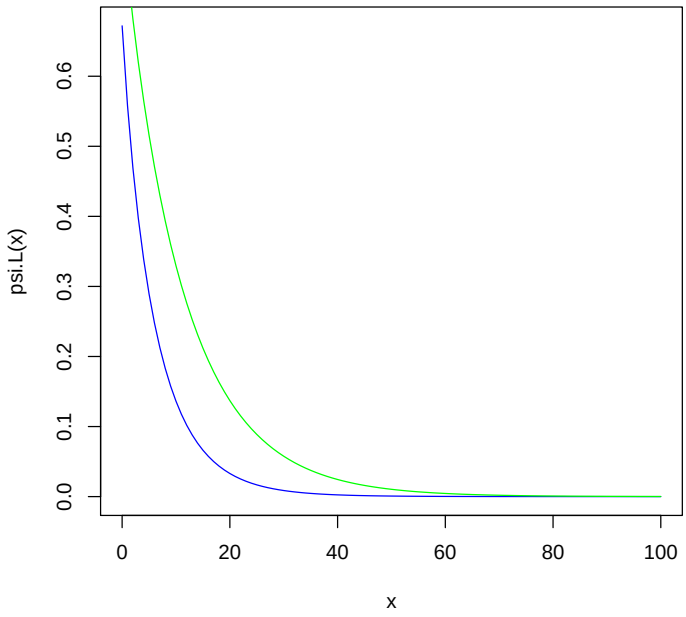

Figure 6: Lower and upper bounds for the probability of ruin as determined using Beekman's convolution formula.

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

R. Kaas, M. Goovaerts, J. Dhaene, and M. Denuit. *Modern Actuarial Risk Theory. Using R*. Springer, 2 edition, 2008. ISBN 978-3-54070992-3.

R. Kass. Beekman's convolution formula. In J. L. Teugels and B. Sundt, editors, *Encyclopedia of actuarial science*, volume 1. Wiley, 2004. ISBN 0-4708467-6-3.

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

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