Loss distributions modeling features of **actuar**

Christophe Dutang
ISFA, Université Claude Bernard Lyon 1

Vincent Goulet
École d'actuariat, Université Laval

Mathieu Pigeon
École d'actuariat, Université Laval

1 Introduction

One important task of actuaries is the modeling of claim amount distributions for ratemaking, loss reserving or other risk evaluation purposes. Package **actuar** (Dutang et al., 2008) offers many functions for loss distributions modeling. The following subsections detail 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 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 `dfoo`, `pfoo`, `qfoo` and `rfoo`, respectively.

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.

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).

Table 1 lists the supported distributions as named in Klugman et al. (2004) along with the root names of the R functions. For reference, Appendix A also gives for every distribution the pdf, the cdf and the name of the argument 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.

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

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

theoretical limited moments

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

and the moment generating function

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

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 and the inverse Gaussian distribution of 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 5 for their empirical counterparts. The `mgf` functions come in handy to compute the adjustment coefficient in ruin theory; see the `"risk"` vignette.

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

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

be the transition rates matrix (or intensity matrix) of such a process and let $(\boldsymbol{\pi}, \pi_{m+1})$ be the initial probability vector. Here, $\mathbf{T}$ is an $m \times m$ non-singular matrix with $t_{ii} < 0$ for $i = 1, \ldots, m$ and $t_{ij} \geq 0$ for $i \neq j$, $\mathbf{t} = -\mathbf{T}\mathbf{e}$ and $\mathbf{e}$ 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 $\mathbf{T}$ is

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

where

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

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

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

The package provides `d`, `p`, `r`, `m` and `mgf` functions for phase-type distributions. The root is `phtype` and parameters $\boldsymbol{\pi}$ and $\mathbf{T}$ are named `prob` and `rates`, respectively. For the package, function `pphtype` is central to the evaluation of the probability of ruin; see `?ruin` and the `"risk"` vignette.

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).

3 Grouped data

Grouped data is data represented in an interval-frequency manner. Typically, a grouped data set will 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

```
> 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

```
> class(x)

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

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

```
> 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):

```
> x[, 1]                                    # group boundaries

[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]                                   # group frequencies

  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,]                                   # first 3 groups

      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                          # frequency replacement

       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          # frequency replacement

       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                 # boundary replacement

       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.

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:

```
> mean(x)

Line.1 Line.2
 188.0  108.2
```

Higher empirical moments can be computed with `emm`; see Section 5.

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):

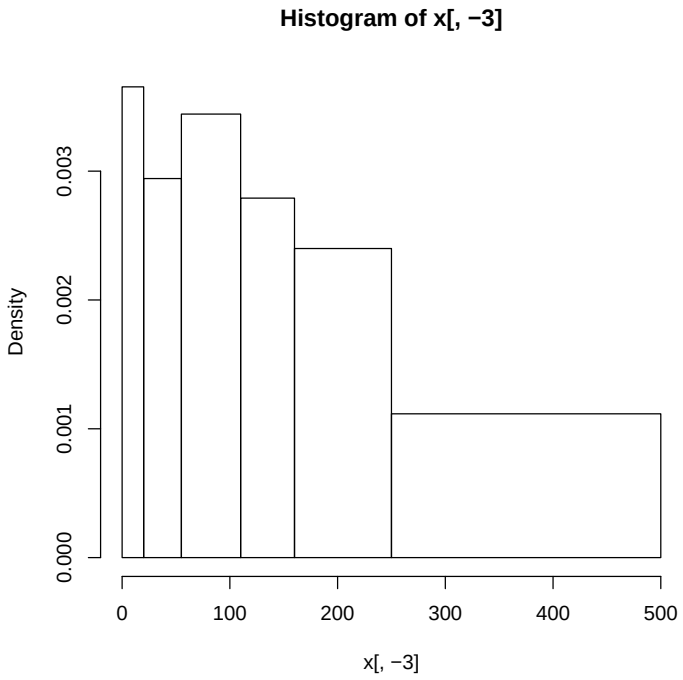

Figure 1: Histogram of a grouped data object

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

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

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

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 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:

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

The package includes a function `ogive` that otherwise behaves exactly like

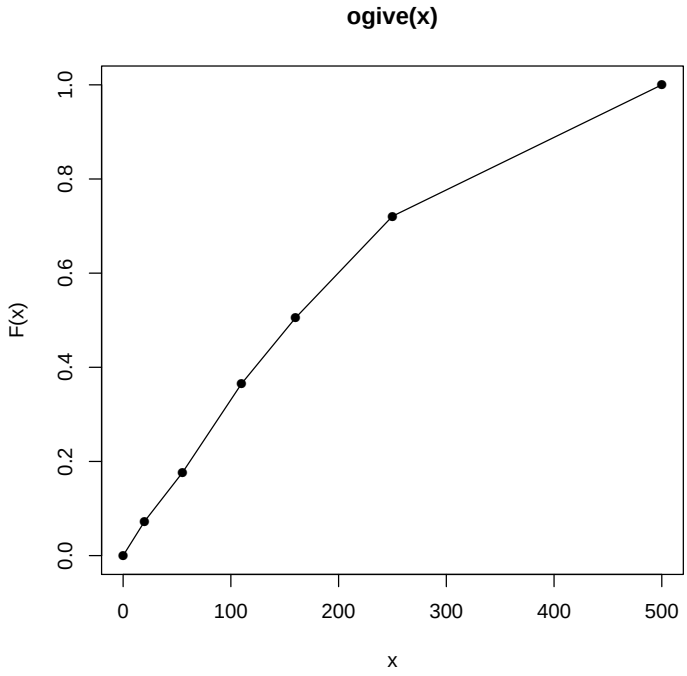

Figure 2: Ogive of a grouped data object

`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):

```
> Fnt <- ogive(x)
> knots(Fnt)                                  # group boundaries
[1]   0  20  55 110 160 250 500
> Fnt(knots(Fnt))                             # ogive at group boundaries
[1] 0.00000 0.07309 0.17608 0.36545 0.50498 0.72093 1.00000
> plot(Fnt)                                   # plot of the ogive
```

Finally, a method of function `quantile` for grouped data objects returns linearly smoothed quantiles, that is the inverse of the ogive evaluated at various points:

```
> quantile(x)
    0%    25%    50%    75%   100%
  0.00  76.47 158.21 276.04 500.00
> Fnt(quantile(x))
[1] 0.00 0.25 0.50 0.75 1.00
```

4 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
```

5 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:

```
> emm(dental, order = 1:3)                # first three moments

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

> emm(gdental, order = 1:3)               # idem

[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))                         # ELEV at data points

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

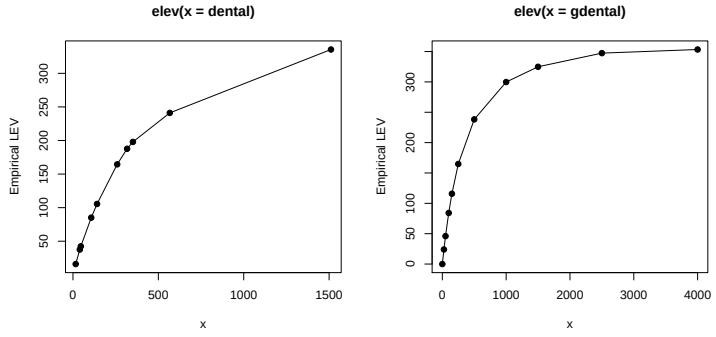

Figure 3: Empirical limited expected value function of an individual data object (left) and a grouped data object (right)

```
> plot(lev, type = "o", pch = 19)          # plot of the ELEV function
> lev <- elev(gdental)
> lev(knots(lev))                          # ELEV at data points

 [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)          # plot of the ELEV function
```

6 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:

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

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

The arguments of `mde` are a data set, a function to compute $F(x)$ or $E[X \wedge x]$, 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") # no convergence

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
```

This procedure may introduce additional bias in the estimators, though.

7 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

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 per-loss variable (Y^L) and per-payment variable (Y^P) as defined in Klugman et al. (2004).

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 $Y_1^P, \ldots, Y_n^P$ (or $Y_1^L, \ldots, Y_n^L$) from the random variable Y^P (Y^L) to fit a model on the unobservable random variable X. This requires expressing the pdf or cdf of Y^P (Y^L) 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:

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

function (x, shape, rate = 1, scale = 1/rate)
{
    Call <- match.call()
    Sd <- Call
    Sd$lower.tail <- FALSE
    Sd[[1L]] <- as.name("pgamma")
    names(Sd)[2L] <- "q"
    Sd[[2L]] <- 1
    Su <- Sd
    Su[[2L]] <- 10
    f <- Call
    f[[1L]] <- as.name("dgamma")
    res <- numeric(length(x))
    w <- which(0 < x & x < 9)
    f[[2L]] <- x[w] + 1
    res[w] <- eval.parent(f)/(p <- eval.parent(Sd))
    res[x == 9] <- eval.parent(Su)/p
    res
}
<environment: 0x7fd3b3aa8e68>

> 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
```

Note how function `f` is built specifically for the coverage modifications submitted and contains as little useless code as possible.

Let object `y` contain a sample of claims amounts from policies with the above deductible and limit. Then one can fit a gamma distribution by maximum likelihood to the claim severity process as follows:

```
> library(MASS)
> fitdistr(y, f, start = list(shape = 2, rate = 0.5))

   shape     rate
  5.0399   1.0001
 (0.8117) (0.1669)
```

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

References

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

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.

R. V. Hogg and S. A. Klugman. *Loss Distributions*. Wiley, New York, 1984. ISBN 0-4718792-9-0.

S. A. Klugman, H. H. Panjer, and G. Willmot. *Loss Models: From Data to Decisions*. Wiley, New York, 1998. ISBN 0-4712388-4-8.

S. A. Klugman, H. H. Panjer, and G. Willmot. *Loss Models: From Data to Decisions*. Wiley, New York, second edition, 2004. ISBN 0-4712157-7-5.

M. F. Neuts. *Matrix-geometric solutions in stochastic models: an algorithmic approach*. Dover Publications, 1981. ISBN 978-0-4866834-2-3.

W. N. Venables and B. D. Ripley. *Modern applied statistics with S*. Springer, New York, 4 edition, 2002. ISBN 0-3879545-7-0.

B. Wheeler. *SuppDists: Supplementary distributions*, 2008. URL http://cran.r-project.org/package=SuppDists. R package version 1.1-2.

A Probability laws

This appendix gives the pdf and cdf of the probability laws appearing in Table 1 using the parametrization of Klugman et al. (2004) and Hogg and Klugman (1984).

In the following,

$$\Gamma(\alpha; x) = \frac{1}{\Gamma(\alpha)} \int_0^x t^{\alpha-1} e^{-t}\, dt, \quad \alpha > 0, x > 0$$

with

$$\Gamma(\alpha) = \int_0^\infty t^{\alpha-1} e^{-t}\, dt$$

is the incomplete gamma function, whereas

$$\beta(a, b; x) = \frac{1}{\beta(a, b)} \int_0^x t^{a-1}(1-t)^{b-1}\, dt, \quad a > 0, b > 0, 0 < x < 1$$

with

$$\begin{aligned}\beta(a,b) &= \int_0^1 t^{a-1}(1-t)^{b-1}\,dt \\ &= \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}\end{aligned}$$

is the (regularized) incomplete beta function.

Unless otherwise stated all parameters are strictly positive and the functions are defined for $x > 0$.

A.1 Transformed beta family

Transformed beta

Root: `trbeta`, `pearson6`
Parameters: `shape1` (α), `shape2` (γ), `shape3` (τ), `rate` ($\lambda = 1/\theta$), `scale` (θ)

$$\begin{aligned}f(x) &= \frac{\gamma u^{\tau}(1-u)^{\alpha}}{x\beta(\alpha,\tau)}, \qquad u = \frac{v}{1+v}, \qquad v = \left(\frac{x}{\theta}\right)^{\gamma} \\ F(x) &= \beta(\tau,\alpha;u)\end{aligned}$$

Burr

Root: `burr`
Parameters: `shape1` (α), `shape2` (γ), `rate` ($\lambda = 1/\theta$), `scale` (θ)

$$\begin{aligned}f(x) &= \frac{\alpha\gamma u^{\alpha}(1-u)}{x}, \qquad u = \frac{1}{1+v}, \qquad v = \left(\frac{x}{\theta}\right)^{\gamma} \\ F(x) &= 1-u^{\alpha}\end{aligned}$$

Loglogistic

Root: `llogis`
Parameters: `shape` (γ), `rate` ($\lambda = 1/\theta$), `scale` (θ)

$$\begin{aligned}f(x) &= \frac{\gamma u(1-u)}{x}, \qquad u = \frac{v}{1+v}, \qquad v = \left(\frac{x}{\theta}\right)^{\gamma} \\ F(x) &= u\end{aligned}$$

Paralogistic

Root: `paralogis`
Parameters: `shape` (α), `rate` ($\lambda = 1/\theta$), `scale` (θ)

$$f(x) = \frac{\alpha^2 u^\alpha (1-u)}{x}, \qquad u = \frac{1}{1+v}, \qquad v = \left(\frac{x}{\theta}\right)^\alpha$$
$$F(x) = 1 - u^\alpha$$

Generalized Pareto

Root: `genpareto`
Parameters: `shape1` (α), `shape2` (τ), `rate` ($\lambda = 1/\theta$), `scale` (θ)

$$f(x) = \frac{u^\tau (1-u)^\alpha}{x\beta(\alpha,\tau)}, \qquad u = \frac{v}{1+v}, \qquad v = \frac{x}{\theta}$$
$$F(x) = \beta(\tau, \alpha; u)$$

Pareto

Root: `pareto`, `pareto2`
Parameters: `shape` (α), `scale` (θ)

$$f(x) = \frac{\alpha u^\alpha (1-u)}{x}, \qquad u = \frac{1}{1+v}, \qquad v = \frac{x}{\theta}$$
$$F(x) = 1 - u^\alpha$$

Inverse Burr

Root: `invburr`
Parameters: `shape1` (τ), `shape2` (γ), `rate` ($\lambda = 1/\theta$), `scale` (θ)

$$f(x) = \frac{\tau\gamma u^\tau (1-u)}{x}, \qquad u = \frac{v}{1+v}, \qquad v = \left(\frac{x}{\theta}\right)^\gamma$$
$$F(x) = u^\tau$$

Inverse Pareto

Root: `invpareto`
Parameters: `shape` (τ), `scale` (θ)

$$f(x) = \frac{\tau u^\tau (1-u)}{x}, \qquad u = \frac{v}{1+v}, \qquad v = \frac{x}{\theta}$$
$$F(x) = u^\tau$$

Inverse paralogictic

Root: `invparalogis`
Parameters: `shape` (τ), `rate` ($\lambda = 1/\theta$), `scale` (θ)

$$f(x) = \frac{\tau^2 u^\tau (1-u)}{x}, \qquad u = \frac{v}{1+v}, \qquad v = \left(\frac{x}{\theta}\right)^\tau$$
$$F(x) = u^\tau$$

A.2 Transformed gamma family

Transformed gamma

Root: `trgamma`
Parameters: `shape1` (α), `shape2` (τ), `rate` ($\lambda = 1/\theta$), `scale` (θ)

$$f(x) = \frac{\tau u^\alpha e^{-u}}{x\Gamma(\alpha)}, \qquad u = \left(\frac{x}{\theta}\right)^\tau$$
$$F(x) = \Gamma(\alpha; u)$$

Inverse transformed gamma

Root: `invtrgamma`
Parameters: `shape1` (α), `shape2` (τ), `rate` ($\lambda = 1/\theta$), `scale` (θ)

$$f(x) = \frac{\tau u^\alpha e^{-u}}{x\Gamma(\alpha)}, \qquad u = \left(\frac{\theta}{x}\right)^\tau$$
$$F(x) = 1 - \Gamma(\alpha; u)$$

Inverse gamma

Root: `invgamma`
Parameters: `shape` (α), `rate` ($\lambda = 1/\theta$), `scale` (θ)

$$f(x) = \frac{u^\alpha e^{-u}}{x\Gamma(\alpha)}, \qquad u = \frac{\theta}{x}$$
$$F(x) = 1 - \Gamma(\alpha; u)$$

Inverse Weibull

Root: `invweibull`, `lgompertz`
Parameters: `shape` (τ), `rate` ($\lambda = 1/\theta$), `scale` (θ)

$$f(x) = \frac{\tau u e^{-u}}{x}, \qquad u = \left(\frac{\theta}{x}\right)^{\tau}$$
$$F(x) = e^{-u}$$

Inverse exponential

Root: `invexp`
Parameters: `rate` ($\lambda = 1/\theta$), `scale` (θ)

$$f(x) = \frac{u e^{-u}}{x}, \qquad u = \frac{\theta}{x}$$
$$F(x) = e^{-u}$$

A.3 Other distributions

Loggamma

Root: `lgamma`
Parameters: `shapelog` (α), `ratelog` (λ)

$$f(x) = \frac{\lambda^{\alpha}(\ln x)^{\alpha-1}}{x^{\lambda+1}\Gamma(\alpha)}, \qquad x > 1$$
$$F(x) = \Gamma(\alpha; \lambda \ln x), \qquad x > 1$$

Single parameter Pareto

Root: `pareto1`
Parameters: `shape` (α), `min` (θ)

$$f(x) = \frac{\alpha\theta^{\alpha}}{x^{\alpha+1}}, \qquad x > \theta$$
$$F(x) = 1 - \left(\frac{\theta}{x}\right)^{\alpha}, \qquad x > \theta$$

Although there appears to be two parameters, only α is a true parameter. The value of θ is the minimum of the distribution and is usually set in advance.

Generalized beta

Root: `genbeta`
Parameters: `shape1` (α), `shape2` (β), `shape3` (τ), `rate` ($\lambda = 1/\theta$), `scale` (θ)

$$f(x) = \frac{\tau u^{\alpha}(1-u)^{\beta-1}}{x\beta(\alpha,\beta)}, \qquad u = \left(\frac{x}{\theta}\right)^{\tau}, \qquad 0 < x < \theta$$
$$F(x) = \beta(\alpha,\beta;u)$$