The lifecontingencies Package. A Package to Perform Financial and Actuarial Mathematics Calculations in R

Giorgio Alfredo Spedicato, Ph.D

Abstract

lifecontingencies R package performs financial and actuarial mathematics calculations to model life contingencies insurances. Its functions permit to determine both the expected value and the stochastic distribution of insured benefits. Therefore they can be used both to price life insurance coverage as long as to assess portfolios' risk based capital requirements.

This paper briefly summarizes the theory regarding life contingencies, that is grounded on financial mathematics and demography concepts. Then, with the aid of applied examples, it shows how **lifecontingencies** package is a useful tool to perform routinary deterministic or stochastic calculations for life contingencies actuarial mathematics.

Keywords: life tables, financial mathematics, actuarial mathematics, life insurance.

1. Introduction

As of December 2012, **lifecontingencies** package, (Spedicato 2012), appears as the first R package that deals with life contingent actuarial mathematics. R statistical programming environment, (R Development Core Team 2012), has become the reference software for academics. In addition, in business context R is now considered a valid alternative to affirmed proprietary packages for statistics and data analysis, like as SAS, (SAS Institute Inc. 2011), MATLAB, (MATLAB 2010), and SPSS, (IBM Corp 2012). Regarding actuarial applications, some packages have already been developed within R. However, most of them mainly focus on non-life insurance. In fact, non - life insurance modeling involves more data analysis and applied statistical modelling than life insurance applications do. Functions to fit loss distributions and to perform credibility analysis are provided within the package **actuar**, (Christophe Dutang, Vincent Goulet, and Mathieu Pigeon 2008). Package **actuar** represents the computational side of the classical actuarial textbook Loss Distribution, (Klugman, Panjer, Willmot, and Venter 2009). The package **ChainLadder**, (Gesmann and Zhang 2011), provides functions to estimate loss reserves for non - life insurances. Generalized Linear Models (GLMs), widely used in non - life insurance rate making, can be fit by functions bundled in the base R distribution. More advanced predictive models used by actuaries, that are Generalized Additive Models for Location, Shape and Scale (GAMLSS) and Tweedie Regression for example, are handled by specifically developed packages as **gamlss**, (Rigby and Stasinopoulos 2005), and **cplm**, (Zhang 2011).
Life insurance actuarial works mainly deal with demographic and financial data. The Fi-

nance view on CRAN site lists packages specifically tailored for financial analysis. Packages **YieldCurve**, (Guirreri 2010), and **termstrc**, (Ferstl and Hayden 2010), can be used to perform financial modeling on interest rates. Among the few packages that handle demographic data **demography**, (Rob J Hyndman, Heather Booth, Leonie Tickle, and John Maindonald 2011), and **LifeTables**, (Riffe 2011) can be used to manage life table and demographic projections.

Numerous commercial software specifically tailored to life insurance actuarial analysis are available, on the other hand. "Moses" and "Prophet" are currently the leading actuarial softwares for life insurance modelling. **lifecontingencies** package aims to represent the R computational companion of the theoretical concepts exposed in textbooks like the classical Bowers, Jones, Gerber, Nesbitt, and Hickman (1997) and Dickson, Hardy, and Waters (2009) for actuarial mathematics and Broverman (2008) for financial mathematics. All along the paper, examples have been taken from (Chris Ruckman and Joe Francis 2006) and (Finan 2012), freely available financial and actuarial mathematic textbooks. The paper has been structured as follows: Section 2 outlines the statistical and financial mathematics theory regarding life contingencies, Section 3 overviews the structure of the **lifecontingencies** package, Section 4 gives a wide choice of applied **lifecontingencies** examples, finally Section 5 discusses package actual and future development as well as known limitations.

2. Life contingencies statistical and financial foundations

Life contingent insurances actuarial pricing and reserving involves the calculation of statistics regarding occurrences and amounts of future cash flows. I.e., the insurance pure premium (also known as benefit premium) can be thought as the expected value of the prospective benefits cash flow distribution, currently valued given an interest rates structure. Prospective benefits cash flows probabilities are based on the occurrence of policyholder's life events (life contingencies). In addition, the theory of interest is used to present value such amounts that will occur in the future. Therefore, life insurance actuarial mathematics bases itself on concepts derived from demography and theory of interest.

A life table (also called a mortality table or actuarial table) is a table that shows how mortality affects subjects of a cohort across different ages. It reports for each age x, the number of l_x individuals living at the beginning of age x. It consists in a sequence of $l_0, l_1, \ldots, l_\omega$, where ω, the terminal age, represents the farthest age until which a subject of the cohort can survive. Life table are typically distinguished according to gender, year of birth and nationality. Life tables are also commonly developed by line of business, assurance vs annuity for example.

From a statistical point of view, a life table allows the probability distribution of the future lifetime for a policyholder aged x, to be deduced. In particular, a life table allows to derive two key probability distributions: $\tilde{T}_x$, the complete future lifetime for a policyholder aged x and its curtate form, $\tilde{K}_x$, that is the number of complete future years completed before death. Therefore, many demographic statistics can be derived from the life table, of which a non exhaustive list follows:

- ${}_tp_x = \frac{l_{x+t}}{l_x}$, the probability that a policyholder alive at age x will reach age $x+t$.

- ${}_tq_x$, the complementary probability of ${}_tp_x$.
- ${}_td_x$, the number of deaths between age x and $x+t$.
- ${}_tL_x = \int_0^t l_{x+y}dy$, the expected number of years lived by the cohort between ages x and $x+t$.
- ${}_tm_x = \frac{{}_td_x}{{}_tL_x}$, the central mortality rate between ages x and $x+t$.
- e_x, the curtate expectation of life for a subject aged x, $e_x = E\left[\tilde{K}_x\right] = \sum_{k=1}^{\infty} {}_kp_x$.

The Keyfitz textbook, Keyfitz and Caswell (2005), provides an exhaustive coverage about life table theory and practice. Life table are usually published by institutions that have access to large amount of reliable historical data, like social security bureaus. It is a common practice for actuaries to start from these life tables and to adapt them to the insurer's portfolio actual experience.

Classical financial mathematics deals with monetary amounts that could be available in different times. The present value of a series of cash flows, expressed by Equation 1, is probably the most important concept. The present value can be considered as the value in current money of a series of financial cash flows, CF_t, that are available in different periods of time.

$$PV = \sum_{t \in T} \mathrm{CF}_t (1+i_t)^{-t} \tag{1}$$

The interest rate, i, represents the measure of the price of money available in future times. Parallel to the interest rate, the time value of the money can be expressed by means of discount rates, $d = \frac{i}{1+i}$. This paper will use the i symbol to express the effective compound interest, when money is invested once per period. In case money is invested more frequenty, say m times per period, each fractional period is named the interest conversion period. During each interest conversion period, the real interest rate $\frac{i^{(m)}}{m}$ is earned, where the $i^{(m)}$ expression defines the convertible (also known as "nominal") rate of interest payable m times per period.

Equation 2 combines the various notations for interest and discount rates, both on effective and convertible basis, to express how an amount of \$1 growths until time t.

$$A(t) = (1+i)^t = (1-d)^{-t} = v^{-t} = \left(1 + \frac{i^m}{m}\right)^{t*m} = \left(1 - \frac{d^m}{m}\right)^{-t*m} \tag{2}$$

All financial mathematics functions (such annuities, $a_{\overline{n}|}$, or accumulated values, $s_{\overline{n}|}$) can be rewritten as particular expressions of Equation 1. (See the classical Broverman 2008, textbook for further discussions on the topic).

Actuaries use the probabilities inherent the life tables to price life contingencies insurances. Life contingencies are themselves stochastic variables, in fact. A life contingent insurance can be represented by a sequence of one or more payments whose occurrence and timing, and therefore the present value of the whole sequence, are not certain. In fact their eventual occurrence and its timing depend by events regarding the life of the policyholder (that is the reason they are named "life contingencies" for). Since the actuary focuses on the present value of such uncertain payments, life contingencies insurances future payments need to be discounted using interest rates that may be also considered stochastic. **lifecontingencies** package provides functions to model most of standard life contingen random variables, $\tilde{Z}$, and in particular their expected value, the Actuarial Present Value (APV). APV is certainly the most important statistic on $\tilde{Z}$ variables that actuaries use. In fact, it represents the average cost of the benefits the insurer guarantees to policyholders. In a non - life insurance context it would be also named pure premium. The benefit premiums plus a loading for expense, profits and taxes sum up to the gross premium, G, that the policyholder pays. Life contingencies can be either continuous or discrete, as cited actuarial mathematics textbooks detail. **lifecontingencies** package directly models only discrete life contingencies with non - stochastic interest rate. Nevertheless, most continuous time life contingent insurances can be easily derived from their discrete form under broad assumptions that can be found in cited textbooks.

Few examples of life contingency insurances follow:

1. A n-year term life insurance provides a payment of $\$b$, if the insured dies within n years from issue. If the payment is performed at the end of year of death, $\tilde{Z}$ can be written as $\tilde{Z} = \begin{cases} v^{K+1}, \tilde{K}_x = 0, 1, \ldots, n-1 \\ 0, \tilde{K}_x \geq n \end{cases}$ Its APV expression is $A^{1}_{x:\overline{n}|}$.

2. A life annuity consists in a sequence of benefits paid as long as the insured life survives. In particular, a temporary life annuity due pays a benefit at the beginning of each period so long as the annuitant aged x survives, for up to a total of n years, or n payments. $\tilde{Z}$ can be written as $\tilde{Z} = \begin{cases} \ddot{a}_{\overline{K+1}|}, \tilde{K}_x < n \\ \ddot{a}_{\overline{n}|}, \tilde{K}_x \geq n \end{cases}$. Its APV expression is $\ddot{a}_{x:\overline{n}|}$.

3. A n-year pure endowment insurance grants a benefit payable at the end of n years, if the insured survives at least n years from issue. $\tilde{Z}$ can be written as $\tilde{Z} = \begin{cases} 0, \tilde{K}_x < n \\ v^n, \tilde{K}_x \geq n \end{cases}$. Its APV expression is $A_{x:\overset{1}{\overline{n}|}}$ (or ${}_nE_x$).

4. A n-year endowment insurance will pay a benefit either at the earlier of the year of death or the end of the n-th year, whichever occurs earlier. $\tilde{Z}$ can be written as $\tilde{Z} = \begin{cases} v^{K+1}, \tilde{K}_x = 0, 1, \ldots, n-1 \\ v^n, \tilde{K}_x \geq n \end{cases}$. Its APV expression is $A_{x:\overline{n}|}$.

Interested readers could see cited references for formulas regarding other life contingent insurances as $(DA)^{1}_{x:\overline{n}|}$, the decreasing term life insurance, $(IA)^{1}_{x:\overline{n}|}$, the increasing term life insurance, and common variations on payment form arrangements like deferment and fractional payments. Similarly, it is possible to define insurances and annuities depending on the survival status of two or more lives. For example, A_{xy} and $a_{\overline{xy}}$ represent respectively the two lives joint-live insurance and the two lives last-survivor annuity immediate APV symbols.

The **lifecontingencies** package provides functions that allow the actuary to perform classical financial and actuarial mathematics calculations. In addition to standard deterministic modeling, a peculiar feature of **lifecontingencies** is that it allows to generates variates from the stochastic distribution of the present value of future benefits, $\tilde{Z}$, for most life contingent insurances. This feature permits a deeper assessment of the insurance liabilities variability.

3. The structure of the package

Package **lifecontingencies** contains classes and methods to handle life-tables and actuarial tables conveniently.

The package is loaded within the R command line as it follows:

```
R> library("lifecontingencies")
```

Two main S4 classes, (Chambers 2008), have been defined within the **lifecontingencies** package: the `lifetable` class and the `actuarialtable` class. The lifetable class is defined as follows

```
R> showClass("lifetable")
```

```
Class "lifetable" [package "lifecontingencies"]

Slots:

Name:          x        lx      name
Class:   numeric   numeric character

Known Subclasses: "actuarialtable"
```

Class `actuarialtable` inherits from `lifetable` class being different from `lifetable` class by one more slot accounting for the interest rate.

```
R> showClass("actuarialtable")
```

```
Class "actuarialtable" [package "lifecontingencies"]

Slots:

Name:   interest         x        lx      name
Class:   numeric   numeric   numeric character

Extends: "lifetable"
```

A list of methods have been defined for `lifetable` and `actuarialtable` classes.

```
R> showMethods(classes=c("actuarialtable","lifetable"))
```

```
Function: coerce (package methods)
from="actuarialtable", to="data.frame"
from="data.frame", to="lifetable"
from="lifetable", to="data.frame"


Function "flat":
 <not an S4 generic function>
Function: getOmega (package lifecontingencies)
object="lifetable"

Function: head (package utils)
x="lifetable"

Function: initialize (package methods)
.Object="lifetable"
    (inherited from: .Object="ANY")

Function: plot (package graphics)
x="lifetable", y="ANY"

Function: show (package methods)
object="actuarialtable"
object="lifetable"

Function: summary (package base)
object="actuarialtable"
object="lifetable"

Function: tail (package utils)
x="lifetable"
```

The computation of financial, demographic and actuarial quantities is based on dedicated functions that use `lifetable` and `actuarialtable` objects if they are required. Table 1 shows the naming convention for common input parameters used within the package. The sections that follow briefly present such functions by the aid of examples.

Parameter	Significance
`x`	the policyholder's age.
`n`	the coverage duration or payment duration.
`i`	interest rate, that could be varying.
`k`	the frequency of payments.

Table 1: **lifecontingencies** functions parameters naming conventions.

Finally, **lifecontingencies** package depends on **methods** package, to define its classes, and **parallel** package that is used to speed up computations. As detailed in later future plans paragraph, C or C++ code is expected to be used in future version of the package to shorten computational times.

4. Code and examples

The example secton of this paper is structured as follows: Section 4.1 shows classical financial mathematics examples, Section 4.2 deals with life tables and actuarial tables management, Section 4.3 shows classical actuarial mathematics examples while Section 4.4 presents the **lifecontingencies** packages functions to perform simulation analysis.

4.1. Classical financial mathematics example

Function	Purpose
`presentValue`	present value for a series of cash flows.
`annuity`	present value of a annuity - certain, $a_{\overline{n}\rceil}$.
`accumulatedValue`	future value of a series of cash flows, $s_{\overline{n}\rceil}$.
`increasingAnnuity`	present value of an increasing annuity - certain, IA_n.
`decreasingAnnuity`	present value of a decreasing annuity - certain, $DA_{\overline{n}\rceil}$.
`convertible2Effective`	conversion from convertible to effective interest (discount) rates.
`effective2Convertible`	`convertible2Effective` inverse.
`intensity2Interest`	conversion from intensity of interest to the interest rate.
`interest2Intensity`	`intensity2Interest` inverse.
`duration`	dollar / Macaulay duration of a series of cash flows.
`convexity`	convexity of a series of cash flows.

Table 2: **lifecontingencies** functions for financial mathematics.

The **lifecontingencies** package provides functions to perform classical financial mathematics calculations as Table 2 lists.
Some of these implement closed form formulas and their inverses as shown in financial mathematics textbooks. A broader discussion, however, shall be dedicated to `presentValue` function. In fact, `presentValue` function is internally called by most other financial and actuarial functions within **lifecontingencies** package. `presentValue` function calculates present value or APVs by calculating $PV = \sum_{i=1}^{n} c_i * v^{t_i} * p_i$ sum, being the terms in the sum the cash flows vector, c_i, the corresponding discount factors vector, v^{t_i} and the occurrence probabilities vector, p_i elements. Many **lifecontingencies** package functions, like `axn` of `annuity`, work by defining the cash flows, interest rate and probabilities (in case of actuarial functions) patterns vectors, that are passed as arguments to `presentValue` function.

Examples that follow show how to handle interest and discount rates with different compounding frequencies, how to perform present value, annuities and future values analysis calculations as long as loans amortization and bond pricing.

Interest rate functions

Interest rates represent the time - value of the money. Different types of rates can be found in literature. As a remark, Equation 3 displays the relationship between effective interest rate, convertible interest rate, discount factor, force of interest, effective discount rate and convertible discount rate.

$$(1+i)^t = \left(1+\frac{i^{(m)}}{m}\right)^t = v^{-t} = \exp(\delta t) = (1-d)^{-t} = \left(1-\frac{d^{(m)}}{m}\right)^{-t} \tag{3}$$

Functions `interest2Discount`, `discount2Interest`, `convertible2Effective`, `effective2Convertible`, `interest2Intensity`, `intensity2Interest` have been based on Equation 3 and inverse formulas implied therein. Throughout the paper the interest rate used is deemed effective interest rate, unless otherwise stated.

As examples, functions `interest2Discount` and `discount2Interest` represent a convenient way to switch from interest to discount rates and conversely.

```
R> interest2Discount(0.03)

[1] 0.02912621

R> discount2Interest(interest2Discount(0.03))

[1] 0.03
```

Function `convertible2Effective` allows to find what is the effective interest rate implied in a consumer - credit loan that offers 10% convertible (nominal) interest rate with quarterly compounding.

```
R> convertible2Effective(i=0.10,k=4)

[1] 0.1038129
```

Present value and internal rate of return analysis

Performing a project appraisal means evaluating the net present value (NPV) of all projected cash flows. Code below shows an example of NPV analysis.

```
R> capitals <- c(-1000,200,500,700)
R> times <- c(0,1,2,5)
R> presentValue(cashFlows=capitals, timeIds=times, interestRates=0.03)

[1] 269.2989
```

When both interest rates vary and cash flows are uncertain, the `probabilities` parameter can be properly set as following code shows.

```
R> presentValue(cashFlows=capitals, timeIds=times,
+  interestRates=c( 0.04, 0.02, 0.03, 0.05),
+  probabilities=c(1,1,1,0.5))

[1] -58.38946
```

The internal rate of return (IRR) is defined as the interest rate that makes the NPV zero. It is a NPV alternative to rank financial investment projects by the timing and amount of their cash flows. The following example displays how to compute IRR arranging **lifecontingencies** and base R functions.

```
R> getIrr <- function(p) (presentValue(cashFlows=capitals, timeIds=times,
+  interestRates=p) - 0)^2
R> nlm(f=getIrr, p=0.1)$estimate

[1] 0.1105091
```

Annuities and future values

An annuity (certain) is a sequence of payments with specified amount that is present valued. Otherwise, when it is valued at the end of the term of payment is is called future value (or accumulated value). Code below shows examples of annuities, $a_{\overline{n}|}$, and accumulated values, $s_{\overline{n}|}$, evaluations.
The PV of an annuity immediate \$100 payable at the end of next 5 years at 3% interest rate is

```
R> 100 * annuity(i=0.03, n=5)

[1] 457.9707
```

while the corresponding future value is

```
R> 100 * accumulatedValue(i=0.03, n=5)

[1] 530.9136
```

Annuities and future values payable k-thly (where fractional payments of $\frac{1}{k}$ are received for each k-th of period) can be evaluated properly setting the functions' parameters.

```
R> ann1 <- annuity(i=0.03, n=5, k=1, type="immediate")
R> ann2 <- annuity(i=0.03, n=5, k=12, type="immediate")
R> c(ann1,ann2)

[1] 4.579707 4.642342
```

`increasingAnnuity` and `decreasingAnnuity` functions handle increasing and decreasing annuities, whose symbols are IA_x, DA_x respectively. Assuming a ten years term and a 3% interest rate, examples of increasing and decreasing annuities follow.

```
R> incrAnn <- increasingAnnuity(i=0.03, n=10, type="due")
R> decrAnn <- decreasingAnnuity(i=0.03, n=10, type="immediate")
R> c(incrAnn, decrAnn)

[1] 46.18416 48.99324
```

The last example within this section displays the calculation of the present value of a geometrically increasing annuity. If amounts increase by 3% and the interest rate is 4% and its term is 10 years, the implied present value is

```
R> annuity(i=((1+0.04)/(1+0.03)-1), n=10)

[1] 9.48612
```

Loan amortization

lifecontingencies financial mathematics functions allow to define the repayments schedule of any loan arrangement, as this section exemplifies. Let C denote the loaned capital (principal), then assuming an interest rate i, the amount due to the lender at each installment is $R = \frac{C}{a_{\overline{n|}}}$. Therefore the R amount repays $I_t = C_{t-1} * i$ as interest and $C_t = R - I_t$ as capital at each installment. The loan repayment periodic installment, R, is calculated as follows

```
R> capital <- 100000
R> interest <- 0.05
R> payments_per_year <- 2
R> rate_per_period <- (1+interest)^(1/payments_per_year)-1
R> years <- 30
R> R <- 1/payments_per_year *
+  capital/annuity(i=interest, n=years,
+                  k=payments_per_year)
R> R

[1] 3212.9
```

then the balance due at end of period (EoP) is calculated as it follows

```
R> balanceDue <- numeric(years * payments_per_year)
R> balanceDue[1] <- capital * (1+rate_per_period) - R
R> for(i in 2:length(balanceDue)) balanceDue[i]<-
+    balanceDue[i-1] * (1+rate_per_period) - R
```

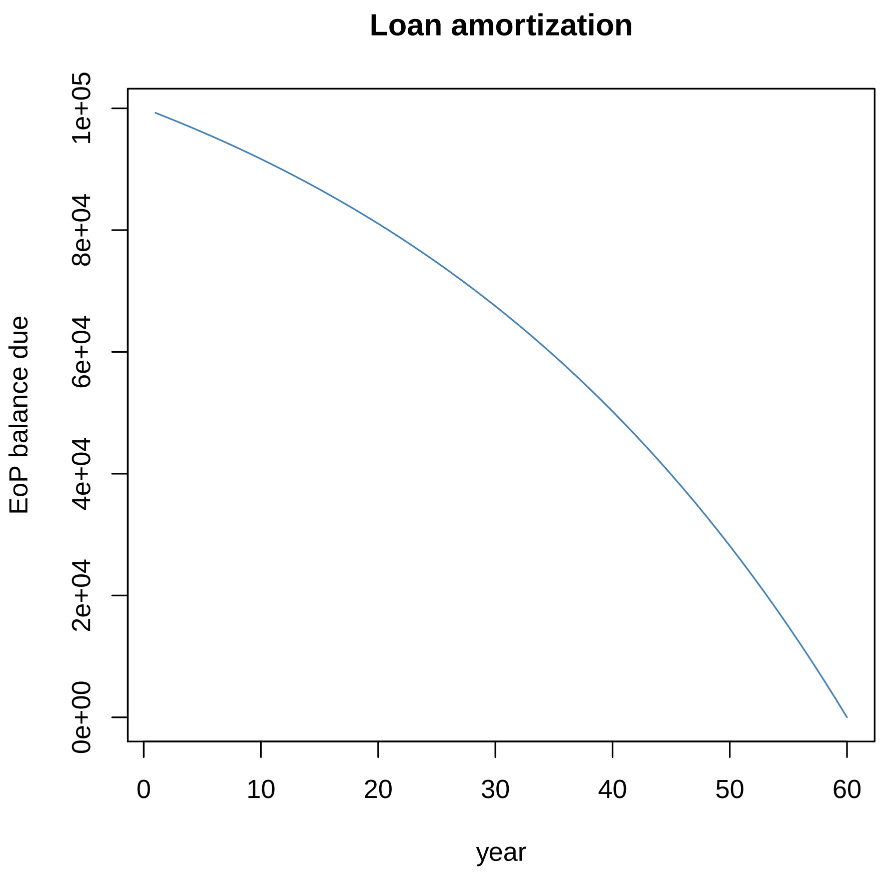

Figure 1: Loan amortization: EoP balance due.

Figure 1 shows the EoP balance due for a 30 years duration loan, assuming a 5% interest rate on a principal of \$ 100,000.

Bond pricing

Bond pricing represents another application of present value. A standard bond whose face value is C and term is T consists in a sequence of equal coupons c paid at regular intervals. The final payment at time T is $C_T + c$. Equation 4 expresses the present value of a bond if n remaining coupons are left.

$$B_t = c * a_{\overline{n}|} + Cv^T \tag{4}$$

Perpetuities are financial contracts that offer an indefinite sequence of payments either at the end (perpetuity-immediate) or at the beginning of each period (perpetuity-due).

Following examples show how **lifecontingencies** package elementary functions can be combined to price bond and perpetuities.

```
R> bond<-function(faceValue, couponRate, couponsPerYear, yield,maturity)
+  {
+          out <- numeric(1)
+          numberOfCF <- maturity * couponsPerYear
+          CFs <- numeric(numberOfCF)
+          payments <- couponRate * faceValue / couponsPerYear
+          cf <- payments * rep(1,numberOfCF)
+          cf[numberOfCF] <- faceValue + payments
+          times <- seq.int(from=1/couponsPerYear, to=maturity,
+                  by=maturity/numberOfCF)
+          out <- presentValue(cashFlows=cf, interestRates=yield,
+                     timeIds=times)
+          return(out)
+  }
R> perpetuity<-function(yield, immediate=TRUE)
+  {
+          out <- numeric(1)
+          out <- 1 / yield
+          out <- ifelse(immediate==TRUE, out, out*(1+yield))
+          return(out)
+  }
R>
```

`bond` and `perpetuity` functions defined above can be used to price any bond, given face value, coupon rate and term, as code below displays.

```
R> bndEx1 <-bond(1000, 0.06, 2, 0.05, 3)
R> bndEx2 <-bond(1000, 0.06, 2, 0.06, 3)
R> ppTy1 <-perpetuity(0.1)
R> c(bndEx1, bndEx2, ppTy1)
```

```
[1] 1029.250 1002.371   10.000
```

Duration and ALM

Duration and convexity formulas as defined within the package are reported in Equation 5 and Equation 6 respectively. Their typical application lies within porfolios' asset - liability management (ALM). (The interested reader could find details in Chris Ruckman and Joe Francis 2006; Broverman 2008, textbooks). However, the following example shows how Macaulay duration (`ex1`), modified duration (`ex2`) and convexity (`ex3`) of any series of cash flows can be calculate by **lifecontingencies** package functions.

$$D = \sum_{t}^{T} \frac{t * \mathrm{CF}_t \left(1 + \frac{i}{m}\right)^{-t*m}}{P} \tag{5}$$

$$C = \sum_{t}^{T} t * \left(t + \frac{1}{m}\right) * \mathrm{CF}_t \left(1 + \frac{y}{m}\right)^{-m*t-2} \tag{6}$$

```
R> cashFlows <- c(100,100,100,600,500,700)
R> timeVector <- seq(1:6)
R> interestRate <- 0.03
R> dur1 <-duration(cashFlows = cashFlows, timeIds = timeVector,
+                    i = interestRate, k = 1, macaulay = TRUE)
R> dur2 <-duration(cashFlows = cashFlows, timeIds = timeVector,
+                    i = interestRate, k = 1, macaulay = FALSE)
R> cvx1 <-convexity(cashFlows = cashFlows, timeIds = timeVector,
+                    i = interestRate, k = 1)
R> c(dur1, dur2, cvx1)
```

```
[1]  4.430218  4.563124 25.746469
```

The last example works out a small ALM problem. Suppose an insurance company has sold a guarantee term certificate (GTC) of face value \$ 10,000, that will mature in 7 years at a 5% interest rate. Its final value would be:

```
R> GTCFin<- 10000 * (1 + 0.05)^7
R> GTCFin
```

```
[1] 14071
```

Imagine the company can hedge its liability with two available investment instruments

1. A five year bond, with face value of 100 and 3% coupon rate yearly coupons.
2. A perpetuity-immediate. As a remark, the formulas for the PV, duration and convexity of a perpetuity immediate are $PV_{pp} = \frac{1}{y}$, $D_{pp} = \frac{1+y}{y}$, $C_{pp} = \frac{2}{y^2}$ respectively, if yield rate is y.

Assume the issuing company wants to hedge its liability with an investment portfolio that is not adverserly affected by changes in the investment yield. In order to solve the ALM problem the composition of assets within the porfolio shall be properly chosen. Moreover assume that the yield rate prevailing now in the marked is 4%. Following lines of code figure out some parameters that are used within the example.

```
R> yieldT0 <- 0.04
R> durLiab <- 7
R> pvLiab <- presentValue(cashFlows = GTCFin,timeIds = 7,
+                   interestRates = yieldT0)
R> convLiab <- convexity(cashFlows=GTCFin, timeIds = 7,
+                   i=yieldT0)
R> pvBond <- bond(100,0.03,1,yieldT0,5)
R> durBond <- duration(cashFlows=c(3,3,3,3,103),
+                   timeIds=seq(1,5), i = yieldT0)
R> convBond <- convexity(cashFlows=c(3,3,3,3,103),
+                   timeIds=seq(1,5), i = yieldT0)
R> pvPpty <- perpetuity(yieldT0)
R> durPpty <- (1+yieldT0)/yieldT0
R> covnPpty <- 2/(yieldT0^2)
```

Then the ALM problem is set out in a three steps problem, as Chris Ruckman and Joe Francis (2006) texbook remarks:

1. setting initial the present value of cash inflows (assets) to be equal to the present value of cash outflows (liabilities).
2. setting the interest rate sensitivity (i.e., the duration) of assets to be equal to the interest rate sensitivity of liabilities. This is done by solving the system of equations shown in Equation 7. w_i and D_i parameters stand for hegding assets weigths and durations values respectively.

$$\begin{cases} w_{\text{bnd}} D_{\text{bnd}} + w_{\text{ppt}} D_{\text{ppt}} = D_{\text{GTC}} \\ w_{\text{bnd}} + w_{\text{ppt}} = 1 \end{cases} \tag{7}$$

3. setting the convexity of asset to be greater than the convexity of liabilities. In other word, this means verifying that assets decline (growth) to be slower (faster) than liability decline in case of changing interest rate.

Following lines of code calculate the asset weights vector by linear algebra functions bundled in R base.

```
R> a <- matrix(c(durBond, durPpty,1,1), nrow=2,
+                   byrow=TRUE)
R> b <- as.vector(c(7,1))
R> weights <-solve(a,b)
R> weights

[1] 0.8848879 0.1151121
```

Vector `weights` displays the portfolio composition in terms of bonds and liabilities respectively. Therefore the number of bonds and perpetuities that can be purchased is determined by

```
R> bondNum <- weights[1] * pvLiab / pvBond
R> pptyNum <- weights[2] * pvLiab / pvPpty
R> bondNum

[1] 99.0279

R> pptyNum

[1] 49.23485
```

It can be verified that the assets convexity is greater than liabilities convexity.

```
R> convAsset <- weights[1] * convBond + weights[2] * covnPpty
R> convAsset>convLiab

[1] TRUE
```

The portfolio is immunized from yield rate variations since if interest rates suddently drops to 3% just after the hedging assets purchase, the present value of assets comes to be greater than the present value of liabilities. The same occurs in case of upward shift of interest rates toward 5%.

```
R> yieldT1low <- 0.03
R> immunizationTestLow <- (bondNum * bond(100,0.03,1,yieldT1low,5) +
+                           pptyNum * perpetuity(yieldT1low)>
+                           GTCFin / (1+yieldT1low)^7)
R> yieldT1high <- 0.05
R> immunizationTestHigh <- (bondNum * bond(100,0.03,1,yieldT1high,5) +
+                           pptyNum * perpetuity(yieldT1high)>
+                           GTCFin/(1+yieldT1high)^7)
R> immunizationTestLow

[1] TRUE

R> immunizationTestHigh

[1] TRUE
```

It is worth to remember that the assets allocation within the portfolio should be rebalanced with some frequence as time goes by, since portfolio's duration and convexity change as time flows.

Function	Purpose
`dxt`	deaths between age x and $x+t$, ${}_td_x$.
`pxt`	survival probability between age x and $x+t$, ${}_tp_x$.
`pxyzt`	survival probability for two (or more) lives, ${}_tp_{xy}$.
`qxt`	death probability between age x and $x+t$, ${}_tq_x$.
`qxyzt`	death probability for two (or more) lives, ${}_tq_{xy}$.
`Txt`	number of person-years lived after exact age x, ${}_tT_x$.
`mxt`	central death rate, ${}_tm_x$.
`exn`	expected lifetime between age x and age $x+n$, ${}_ne_x$.
`rLife`	sample from the time until death distribution underlying a life table.
`rLifexyz`	sample from the time until death distribution underlying two or more life.
`exyz`	n-year curtate lifetime of the joint-life status.
`probs2lifetable`	life table l_x from raw one - year survival / death probabilities.

Table 3: **lifecontingencies** functions for demographic analysis.

4.2. Life tables and actuarial tables analysis

`lifetable` and `actuarialtable` classes are designed to handle demographic and actuarial mathematics calculations. An `actuarialtable` class inherits from `lifetable` class and it adds one more slot to allow for the rate of interest. Both classes have been designed using the `S4` R classes framework.
Table 3 lists the functions that have been developed to perform demographich analysis within **lifecontingencies** package, that this section briefly exemplifies.

Creating lifetable and actuarialtable objects

Life table objects can be created by raw R commands or using existing `data.frame` objects. However, to build a `lifetable` class object three components are needed:

1. The years sequence, that is an integer sequence $0, 1, \ldots, \omega$. It shall start from zero and going to ω, the terminal age (the age x for which $p_x = 0$).
2. The l_x vector, that is the number of subjects living at the beginning of age x, in other words, the number of subjects at risk to die between year x and $x+1$.
3. The name of the life table.

There are three main approaches to create a `lifetable` object:

1. directly from the x and l_x vector.
2. importing x and l_x from an existing `data.frame` object.
3. from raw survival probabilities.

Creating a `lifetable` object directly can be done as shown by code below

```
R> x_example <- seq(from=0,to=9, by=1)
R> lx_example <- c(1000,950,850,700,680,600,550,400,200,50)
```

```
R> exampleLt <- new("lifetable", x=x_example, lx=lx_example,
+                   name="example lifetable")
```

while `print` and `show` methods tabulate the x, l_x, ${}_tp_x$ and e_x values for a given life table.

```
R> print(exampleLt)

Life table example lifetable

  x   lx        px       ex
1 0 1000 0.9500000 4.980000
2 1  950 0.8947368 4.242105
3 2  850 0.8235294 3.741176
4 3  700 0.9714286 3.542857
5 4  680 0.8823529 2.647059
6 5  600 0.9166667 2.000000
7 6  550 0.7272727 1.181818
8 7  400 0.5000000 0.625000
9 8  200 0.2500000 0.250000
```

`head` and `tail` methods for `data.frame` S3 classes have also been implemented on `lifetable` classes

```
R> head(exampleLt)

  x   lx
1 0 1000
2 1  950
3 2  850
4 3  700
5 4  680
6 5  600
```

Nevertheless the easiest way to create a `lifetable` object is to start from a suitable existing `data.frame`. This will be probably the most practical approach for practicing actuaries. Some life or mortality rates table have been bundled within **lifecontingencies** package, as Table 4 displays.

The following example shows how the US Social Security life tables are loaded from the existing `demoUsa` data set bundled in the **lifecontingencies** package.

```
R> data("demoUsa")
R> data("demoIta")
R> usaMale07 <- demoUsa[,c("age", "USSS2007M")]
R> usaMale00 <- demoUsa[,c("age", "USSS2000M")]
R> names(usaMale07) <- c("x","lx")
R> names(usaMale00) <- c("x","lx")
```

Data set	Description
`AF92Lt`	UK AF92 life table.
`AM92Lt`	UK AF92 life table.
`demoChina`	China mortality rates from SOA website.
`demoIta`	Various Italian life tables including RG48 and IPS55 projected tables.
`demoJapan`	Japan mortality rates from SOA website.
`demoUsa`	US Social Security life tables.
`demoFrance`	1990 and 2002 French life tables.
`soa08`	SOA illustrative life table.
`soa08Act`	SOA illustrative actuarial table at 6%.

Table 4: Life tables and other data objects bundled within **lifecontingencies**.

```
R> usaMale07Lt <-as(usaMale07,"lifetable")
R> usaMale07Lt@name <- "USA MALES 2007"
R> usaMale00Lt <-as(usaMale00,"lifetable")
R> usaMale00Lt@name <- "USA MALES 2000"
```

The same operation can be performed on IPS55 tables bundled in the `demoIta` data set. The purpose of following example is to stress that it is important a clean l_x series to be given in input to the coerce method. A ”clean” l_x series means that neither 0 nor missing values are present anywhere and the l_x series to be decreasing.

```
R> lxIPS55M <- with(demoIta, IPS55M)
R> pos2Remove <- which(lxIPS55M %in% c(0,NA))
R> lxIPS55M <-lxIPS55M[-pos2Remove]
R> xIPS55M <-seq(0,length(lxIPS55M)-1,1)
R> ips55M <- new("lifetable",x=xIPS55M, lx=lxIPS55M,
+                    name="IPS 55 Males")
R> lxIPS55F <- with(demoIta, IPS55F)
R> pos2Remove <- which(lxIPS55F %in% c(0,NA))
R> lxIPS55F <- lxIPS55F[-pos2Remove]
R> xIPS55F <- seq(0,length(lxIPS55F)-1,1)
R> ips55F <- new("lifetable",x=xIPS55F, lx=lxIPS55F,
+                    name="IPS 55 Females")
```

The last way a `lifetable` object can be created is from one year survival or death probabilities combining the `probs2lifetable` function and `as.data.frame` coerce methods. Two potential benefits arise from this function. A first benefit lies in the use of a mortality projection method results. (See the Lee - Carter method Lee and Carter 1992, for example). Lee - Carter method allows to vary mortality table by cohort of birth. It makes therefore possible to project demographic quantities, like the expected lifetime, e_0, as a function of year of birth. A second one lies in the creation of ”cut-down” mortality tables. This latter application is exemplified in the code line that follows, where a `itaM2002reduced` life table is obtained cutting down the one - year mortality rates of Italian males aged between 20 and 60 to 20% of its original value.

```
R> data("demoIta")
R> itaM2002 <- demoIta[,c("X","SIM92")]
R> names(itaM2002) <- c("x","lx")
R> itaM2002Lt <- as(itaM2002,"lifetable")
```

```
removing NA and 0s
```

```
R> itaM2002Lt@name <- "IT 2002 Males"
R> itaM2002 <- as(itaM2002Lt,"data.frame")
R> itaM2002$qx <- 1-itaM2002$px
R> for(i in 20:60) itaM2002$qx[itaM2002$x==i] = 0.2 * itaM2002$qx[itaM2002$x==i]
R> itaM2002reduced <- probs2lifetable(probs=itaM2002[,"qx"], radix=100000,
+                     type="qx",name="IT 2002 Males reduced")
```

An `actuarialtable` can be easily created from a `lifetable` existing object.

```
R> exampleAct <- new("actuarialtable",x=exampleLt@x, lx=exampleLt@lx,
+  interest=0.03, name="example actuarialtable")
```

Method `getOmega` returns the terminal age, ω when applied either on `actuarialtable` or `lifetable` classes.

```
R> getOmega(exampleAct)
```

```
[1] 9
```

Method `print` behaves differently between `lifetable` objects and `actuarialtable` objects. In fact, when `print` method is applied on a `lifetable` object, it tabulates one year survival probability and complete expected remaining life until death. Conversely, classical commutation functions (D_x, N_x, C_x, M_x, R_x), discussed further, are printed out when `print` method is applied on a `lifetable` object.

```
R> print(exampleLt)
```

```
Life table example lifetable

  x   lx        px       ex
1 0 1000 0.9500000 4.980000
2 1  950 0.8947368 4.242105
3 2  850 0.8235294 3.741176
4 3  700 0.9714286 3.542857
5 4  680 0.8823529 2.647059
6 5  600 0.9166667 2.000000
7 6  550 0.7272727 1.181818
8 7  400 0.5000000 0.625000
9 8  200 0.2500000 0.250000
```

```
R> print(exampleAct)

Actuarial table  example actuarialtable interest rate  3 %

   x   lx         Dx         Nx        Cx       Mx        Rx
1  0 1000 1000.00000 5467.92787  48.54369 840.7400 4839.7548
2  1  950  922.33010 4467.92787  94.25959 792.1963 3999.0148
3  2  850  801.20652 3545.59778 137.27125 697.9367 3206.8185
4  3  700  640.59916 2744.39125  17.76974 560.6654 2508.8819
5  4  680  604.17119 2103.79209  69.00870 542.8957 1948.2164
6  5  600  517.56527 1499.62090  41.87421 473.8870 1405.3207
7  6  550  460.61634  982.05563 121.96373 432.0128  931.4337
8  7  400  325.23660  521.43929 157.88185 310.0491  499.4210
9  8  200  157.88185  196.20268 114.96251 152.1672  189.3719
10 9   50   38.32084   38.32084  37.20470  37.2047   37.2047
```

It is possible to convert the `actuarialtable` object into a `data.frame` object, as shown below.

```
R> exampleActDf <- as(exampleAct, "data.frame")
```

Finally a `plot` method can be applied to `lifetable` or `actuarialtable` objects. The underlying survival function (that is the plot of x vs l_x) is displayed in both cases. Figure 2 shows the `plot` methods applied on the Society of Actuaries (SoA) actuarial table at 6%, bundled within the **lifecontingencies** package as `soa08Act` object.

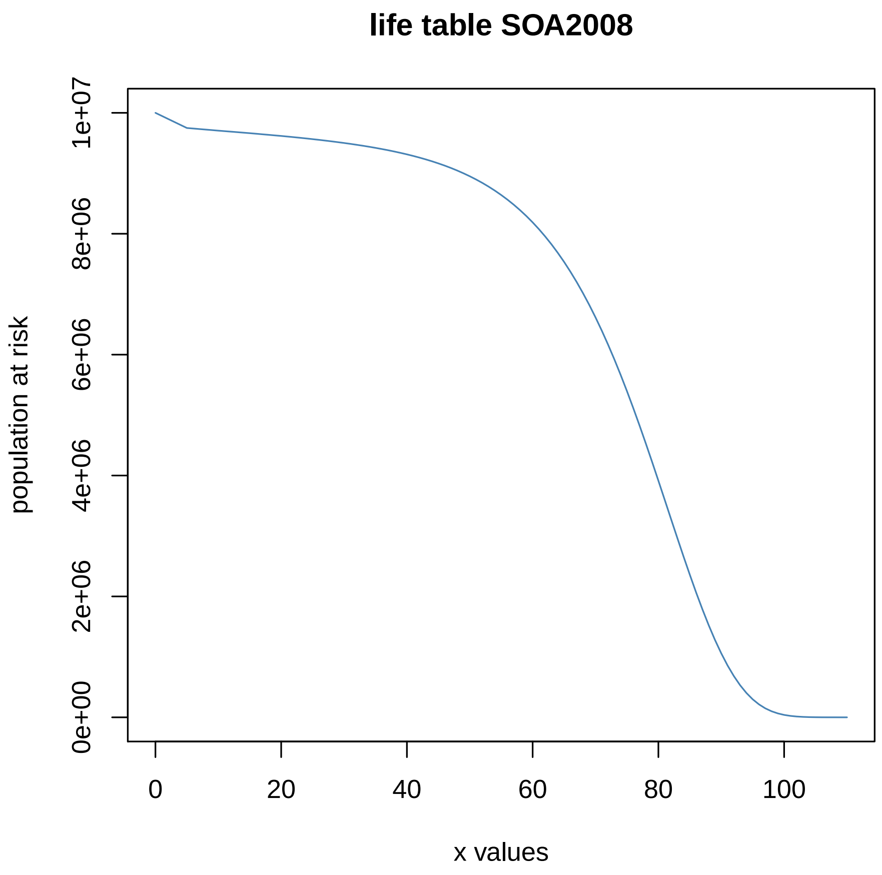

Figure 2: SoA illustrative life table underlying survival function.

Basic demographic analysis

Basic demographic calculations can be performed on valid `lifetable` or `actuariatable` objects. All functions discussed in this sections access to the `lifetable` object slots, calculating proper ratios or sums on l_x or d_x values following demographic formulas definitions.

Code below shows how ${}_1p_{20}$, ${}_2q_{30}$ and $\mathring{e}_{50:\overline{20}|}$ respectively are calculated on the IPS55 male population table

```
R> demoEx1<-pxt(ips55M,20,1)
R> demoEx2<-qxt(ips55M,30,2)
R> demoEx3<-exn(ips55M, 50,20,"complete")
R> c(demoEx1,demoEx2,demoEx3)

[1]  0.999595096  0.001332031 19.472765230
```

The package allows the calculation of fractional survival probabilities using the linear interpolation, constant force of mortality and hyperbolic Balducci's assumptions as code below shows.

```
R> data("soa08Act")
R> pxtLin <- pxt(soa08Act,80,0.5,"linear")
R> pxtCnst <- pxt(soa08Act,80,0.5,"constant force")
R> pxtHyph <- pxt(soa08Act,80,0.5,"hyperbolic")
R> c(pxtLin,pxtCnst,pxtHyph)

[1] 0.9598496 0.9590094 0.9581701
```

Survival probabilities calculations on two (or more) lives can be performed also. As a remark, two different life statuses are defined within multiple lives survival analysis: "joint" survival status and "last" survival status. The "joint" survival status exists until all the members of the pool are alive, while the "last" survival status exists until the last member of the pool death. All calculations assume that multiple lives are independent. Equation 8 mathematically expresses the remaining future lifetime on a couple xy, under the joint and last survival status respectively.

$$\begin{aligned} \tilde{T}_{xy} &= \min\left(T_x, T_y\right) \\ \tilde{T}_{\bar{xy}} &= \max\left(T_x, T_y\right) \end{aligned} \tag{8}$$

Following code lines show how joint survival probability, last survival probability and expected joint lifetime can be evaluated using **lifecontingencies** functions.

```
R> tablesList <- list(ips55M, ips55F)
R> jsp <- pxyzt(tablesList, x=c(65,63), t=2)
R> lsp <- pxyzt(tablesList, x=c(65,63), t=2, status="last")
R> jelt <- exyzt(tablesList, x=c(65,63), status="joint")
R> c(jsp,lsp,jelt)
```

```
[1]  0.9813187  0.9999275 19.1982972
```

4.3. Classical actuarial mathematics examples

Function	Purpose	APV symbol
`Axn`	one life insurance	$A^{1}_{x:\overline{n}\rceil}$.
`AExn`	the n-year endowment	$A_{x:\overset{1}{\overline{n}\rceil}}$.
`Axyzn`	two lives life insurances	$\bar{A}^{\,1}_{\overline{xy}:\overline{n}\rceil}$.
`axn`	one life annuity	$\ddot{a}_{x}$.
`axyzn`	two lives annuities	$\ddot{a}_{xy}$.
`Exn`	pure endowment	${}_{n}E_{x}$.
`Iaxn`	increasing annuity	Ia_{x}.
`IAxn`	increasing life insurance	$(IA)^{1}_{x:\overline{n}\rceil}$.
`DAxn`	decreasing life insurance	$(DA)^{1}_{x:\overline{n}\rceil}$.
`rLifeContingencies`	variates generation from the $\tilde{Z}$ distribution.	
`rLifeContingenciesXyz`	multiple life version of `rLifeContingencies`.	

Table 5: **lifecontingencies** functions for actuarial mathematics.

Table 5 lists the function contained in **lifecontingencies** example that allow the user to perform classical actuarial mathematics calculations. A selection of example follows, where the SOA illustrative life table at 6% interest rates will be used, unless otherwise stated.

Life insurance examples

The evaluation of the APV has traditionally followed three approaches: the use of commutation tables, the current payment technique and the expected value techniques.
Commutation tables extend life table by tabulating special functions of age and rate of interest, (as Anderson 1999, further considers). Ratios of commutation table functions allow the actuary to evaluate APV for standard insurances. However, commutation table usage has become less important in computer era. In fact they are not enough flexible and their usage is computationally inefficient. Therefore, commutation table approach has not been used within **lifecontingencies** package to evaluate APVs.
The current payment technique calculates the APV of a life contingency insurance, $\bar{Z}$, as the scalar product of three vectors: $\bar{Z} = \langle\langle\bar{c} \bullet \bar{v}\rangle \bullet \bar{p}\rangle$. The vector of all possible uncertain cash flows, $\bar{c}$, the vector of discount factors, $\bar{v}$ and the vector of cash flow probabilities, $\bar{p}$. **lifecontingencies** package implements the current payment technique in actuarial functions listed in Table 5 to evaluate APVs. Finally, the expected value approach models $\bar{Z}$ as the scalar product of two vector: $\bar{Z} = \langle\bar{pk} \bullet \bar{x}\rangle$. $\bar{pk}$ is $Pr\left[\tilde{K} = k\right]$, that is the probability that the future curtate lifetime to be exactly k years, $\bar{x}$ is the present value of benefits due under the policy term if $\tilde{K} = k$. `rLifeContingencies` and `rLifeContingenciesXyz` implement the expected value approach to generate $\tilde{Z}$ variates.

Consider an annuity due whose term is n years. Its APV, $\ddot{a}_{x:\overline{n}\rceil}$, using the commutation tables approach is reported in Equation 9, while Equation 10 reports the APV using the current payment technique. Finally, Equation 11 calculates the APV using the expected value approach.

$$\text{APV} = \frac{N_x - N_{x+n}}{D_x} \tag{9}$$

$$\text{APV} = \sum_{k=0}^{\min(\omega-x,n)} {}_kp_x * v^k \tag{10}$$

$$\text{APV} = \sum_{k=0}^{\omega-x} \Pr\left[\tilde{K}_x = k\right] * \ddot{a}_{\overline{\min(k,n)}|} \tag{11}$$

In order to understand how **lifepackage** implements current payment technique in its actuarial function, it is worth to look closer to `axn` function core. `axn` function takes following parameters as input: `n`, the term of the annuity, `k` the fractional payment frequency, `x` the annuitant age and `m`, the deferring period. Then, it defines:

1. The vector of possible payments, $\bar{c}$, by

```
payments = rep(1/k, n * k)
```

2. The vector timing of payments, by

```
times=m + seq(from=0, to=(n-1/k), by=1/k)
```

3. The vector of payment probability, $\bar{p}$, by

```
for(i in 1:length(times)) probs[i] = pxt(
actuarialtable, x,times[i])
```

4. Finally the three vectors are passed as input parameters to `presentValue` function as following line of code shows

```
presentValue(cashFlows=payments, timeIds=times,
interestRates = interest,
probabilities=probs)
```

Examples of premium and reserve calculation of life contingent insurances follows, that make use of SoA illustrative actuarial table unless otherwise stated.

The first example values a 40-year term insurance on a policyholder aged 25, with benefits payable at the end of the month of death. Equation 12 would determine the benefit premium using the commutation table approach.

$$U = \frac{M_{25} - M_{65}}{D_{65}} \frac{i}{i^{(12)}} \tag{12}$$

Following lines of code compute the benefit premium using both the commutation, `UComm`, and the current payment technique, `UCpt`.

```
R> data(soa08Act)
R> UComm <- Axn(actuarialtable=soa08Act, x=25, n=65-25, k=12)
R> UCpt <- ((soa08ActDf$Mx[26]-soa08ActDf$Mx[66])/soa08ActDf$Dx[26]) *
+                   0.06/real2Nominal(i=0.06,k=12)
R> c(UComm, UCpt)
```

```
[1] 0.04927622 0.04927622
```

If instead of being paid in a lump sum, the premium were paid by ten equal installments at the beginning of each year the policyholder is alive, then the yearly premium, P, would be determined as it follows.

```
R> P <- UCpt/axn(actuarialtable=soa08Act,x=25,n=10)
R> P
```

```
[1] 0.006351049
```

lifecontingencies package allows to evaluate APVs of endowment insurances as well as increasing and decreasing life insurances. The code lines that follow will computationally prove the actuarial equivalence expressed by Equation 13.

$$(n+1) * A^{1}_{x:\overline{n}|} = (DA)^{1}_{x:\overline{n}|} + (IA)^{1}_{x:\overline{n}|} \tag{13}$$

```
R> (10 + 1 ) * Axn(actuarialtable=soa08Act, x=25, n=10)
```

```
[1] 0.1194393
```

```
R> DAxn(actuarialtable = soa08Act, x=25, n=10) +
+  IAxn(actuarialtable = soa08Act, x=25, n=10)
```

```
[1] 0.1194393
```

Life annuities examples

Life contingent annuities consist in sequences of payments whose occurrence and duration depend on future policyholder's lifetime. Few examples follow, showing how **lifecontingencies** package can easily compute APV for the typical life contingent annuities insurances directly using bundled functions as well as using the classical commutation table approach.

Equation 14 expresses the full premium for a ten-year deferred annuity due for a policyholder aged 75 by means of commutation functions.

$$U = {}_{10|}\ddot{a}_{75} = \frac{N_{85}}{D_{75}} \tag{14}$$

```
R> UCpt <- axn(actuarialtable=soa08Act, x=75, m=10)
R> UComm <- with(soa08ActDf,Nx[86]/Dx[76])
R> c(UCpt,UComm)

[1] 1.146484 1.146484
```

If the premium were paid by means of five annual payments, as long as the insured is alive, Equation 14 would be rewritten as Equation 15.

$$ {}_5\bar{P}({}_{10|}\ddot{a}_{75}) = \frac{{}_{10|}\ddot{a}_{75}}{\ddot{a}_{75:\overline{5|}}} = \frac{\frac{N_{85}}{D_{75}}}{\frac{N_{75}-N_{80}}{D_{75}}} \tag{15} $$

```
R> P=axn(actuarialtable=soa08Act, x=75, m=10) /
+                   axn(actuarialtable=soa08Act, x=75, n=5)
R> P

[1] 0.2854726

R> PComm <- with(soa08ActDf,(Nx[86]/Dx[76]) /
+                                   ((Nx[76]-Nx[81])/Dx[76]))
R> PComm

[1] 0.2854726
```

If amounts of $\frac{1}{m}$ were paid at the beginning of each month, the APV of the annuity would be $U = {}_{10|}\ddot{a}_{75}^{(12)}$.

```
R> U <- axn(actuarialtable=soa08Act, x=75, m=10, k=12)
R> P <- axn(actuarialtable=soa08Act, x=75, m=10, k=12) /
+                   axn(actuarialtable=soa08Act, x=75, n=5)
R> c(U,P)

[1] 1.0325685 0.2571079
```

Benefit reserves examples

The (prospective) benefit reserve consists in the difference between the APV of future benefit payments obligations due by the insurer and the APV of projected inflows due by the policyholder. It represents the outstanding net insurer's obligation arising from the underwritten insurance policy. An example will better exemplify this concept.
Code below evaluates the benefit reserve for a 25 years old 40 - year life insurance of \$ 100,000, with benefits payable at the end of year of death, assuming level benefit premium payable at the beginning of each year. The benefit premium and reserve equations for this life contingent insurance are displayed by Equation 16.

$$ \begin{aligned} &P\ddot{a}_{25:\overline{40|}} = 100000A^{\;1}_{25:\overline{40|}} \\ &{}_tV^{\;1}_{25+t:\overline{n-t|}} = 100000A^{\;1}_{25+t:\overline{40-t|}} - P\ddot{a}_{25+t:\overline{40-t|}} \end{aligned} \tag{16} $$

```
R> P=100000 * Axn(soa08Act,x=25,n=40)/axn(soa08Act,x=25,n=40)
R> reserveFun = function(t) return(100000*Axn(soa08Act,x=25+t,n=40-t)-P*
+                                 axn(soa08Act,x=25+t,n=40-t))
R> for(t in 0:40) {if(t%%5==0) cat("At time ",t,
+                               " benefit reserve is ",
+                               reserveFun(t),"\n")}

At time  0  benefit reserve is  0
At time  5  benefit reserve is  1109.885
At time  10  benefit reserve is  2401.368
At time  15  benefit reserve is  3825.879
At time  20  benefit reserve is  5256.249
At time  25  benefit reserve is  6421.796
At time  30  benefit reserve is  6789.186
At time  35  benefit reserve is  5328.029
At time  40  benefit reserve is  0
```

Another reserve calculation example shows the benefit reserve for a deferred annuity due on a policyholder aged 25 when the annuity is deferred until age 65. Code below shows the reserve calculation while Figure 3 plots the outstanding reserve at the end of each contract year.

```
R> yearlyRate <- 12000
R> irate <- 0.02
R> APV <- yearlyRate*axn(soa08Act, x=25, i=irate,m=65-25,k=12)
R> levelPremium=APV/axn(soa08Act, x=25,n=65-25,k=12)
R> annuityReserve<-function(t) {
+         out<-NULL
+         if(t<65-25) out <- yearlyRate*axn(soa08Act, x=25+t,
+     i=irate, m=65-(25+t),k=12)-levelPremium*axn(soa08Act,
+              x=25+t, n=65-(25+t),k=12) else {
+                out <- yearlyRate*axn(soa08Act, x=25+t, i=irate,k=12)
+         }
+         return(out)
+ }
R> years <- seq(from=0, to=getOmega(soa08Act)-25-1,by=1)
R> annuityRes <- numeric(length(years))
R> for(i in years) annuityRes[i+1] <- annuityReserve(i)
R> dataAnnuityRes <- data.frame(years=years, reserve=annuityRes)
```

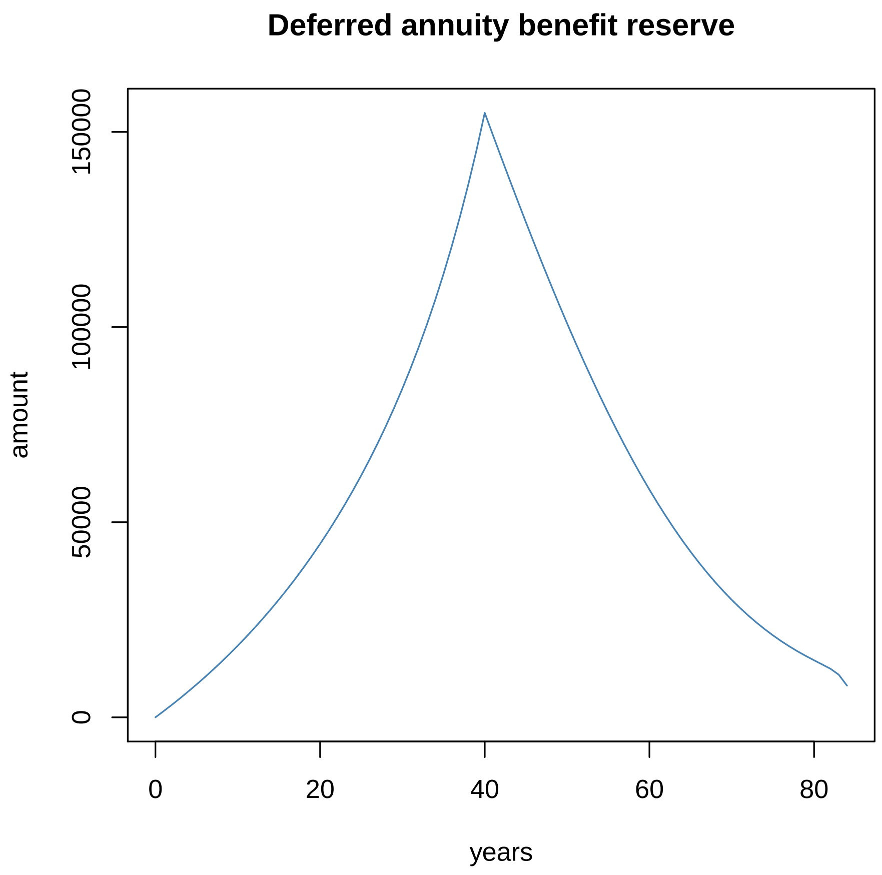

Figure 3: Benefit reserve profile for the exemplified annuity contract

Expenses considerations

The premium paid by the policyholder usually contains an allowance for expenses and profit loading. Expenses cover the policy servicing and the producers' commissions. The insurer profit load is explicitly taken into account in the benefit premium as a flat amount or as a percentage of final premium, in some cases. In other cases an implicit profit loading is generated by using demographic and financial assumptions more prudential than what it would be necessary. The equivalence principle can be extended to the gross premium, G, and expense augmented reserve, ${}_tV^E$, when expense allowances are taken into account by using Equation 17.

$$\begin{aligned} G &= \text{APV}\,(\text{Benefits}) + \text{APV}\,(\text{Expenses}) \\ {}_tV^E &= \text{APV}\,(\text{Benefits}) + \text{APV}\,(\text{Expenses}) - \text{APV}\,(\text{Gross Premium}) \end{aligned} \tag{17}$$

The following example shows how to a expense loaded premium G for a \$ 100,000 whole life insurance on a 35 year old insured $100,000A_{35}$ is calculated assuming the following: 10% of premium expense per year, 25 per year of policy expense, annual maintenance expense of 2.5 per 1,000 unit of capital.

The equation to be solved is $G * \ddot{a}_{35} = 100000 * A_{35} + (2.5 * 100000/1000 + 25 + 0.1G) * \ddot{a}_{35}$.

```
R> G <- (100000*Axn(soa08Act, x=35) + (2.5*100000/1000 + 25)*
+                          axn(soa08Act,x=35))/((1-.1)*axn(soa08Act,x=35))
R> G

[1] 1234.712
```

Insurances and annuities on two lives

The package provides functions designed to evaluate life insurance and annuities on two lives. Following example checks the actuarial mathematics identity on joint and last survival status annuities expressed by Equation 18.

$$a_{\overline{xy}} = a_x + a_y - a_{xy} \tag{18}$$

```
R> twoLifeTables <- list(maleTable=soa08Act, femaleTable=soa08Act)
R> axn(soa08Act, x=65,m=1)+axn(soa08Act, x=70,m=1)-
+  axyn(soa08Act,soa08Act,        x=65,y=70,status="joint",m=1)

[1] 10.35704

R> axyzn(tablesList=twoLifeTables, x=c(65,y=70), status="last",m=1)

[1] 10.35704
```

Finally, reversionary annuities (annuities payable to life y upon death of x) APVs, $a_{x|y} = a_y - a_{xy}$, can also be computed combining **lifecontingencies** functions as the code below shows.

```
R> axn(actuarialtable = soa08Act, x=60,m=1)-
+                     axyzn(tablesList = twoLifeTables,
+                                         x=c(65,60),status="joint",m=1)

[1] 2.695232
```

4.4. Stochastic analysis

This last section illustrates some stochastic analysis that can be performed by **lifecontingencies** package, both in demographic analysis (Section 4.4.1) and life insurance evaluation (Section 4.4.2).

Demographic examples

The age-until-death, both in the continuous, $\tilde{T}_x$, or curtate form, $\tilde{K}_x$, is a stochastic variable whose distribution is intrinsic in the deaths within a life table. Therefore a dedicated function, `rLife`, has been designed within **lifecontingencies** package to draw sample from $\tilde{K}_x$ or $\tilde{T}_x$. Drawing from K_x is quite simple: the distribution of curtate future lifetime is defined, $\Pr\left[\tilde{K}_x = t\right] = \frac{d_{x+t}}{\sum\limits_{j=0}^{\omega-x} l_{x+j}}$, and it is passed as `prob` parameter to base R `sample` function. For example, the code below shows how `rLife` function can be used to draw sample of size five from the curtate future lifetime of a policyholder aged 45 implicit in the SOA life table.

```
R> rLife(n = 5, object = soa08Act, x = 45, type = "Kx")

[1] 40 18 29 12 51
```

`rLifexyz` represents the multiple lives extension of `rLife` function. It returns a matrix of sampled expected future lifetimes of J policyholders given a list of J lifetables. The simulation approach is useful to evaluate demographical quantities when the analytical approach is not feasible. One example could be the expected years of widowhood, that Equation 19 defines. $\tilde{T}_x$ and $\tilde{T}_y$ in Equation 19 stand for complete future lifetimes for the husband and the wife respectively.

$$E\left[\tilde{W}_y\right] = \max\left(0, \tilde{T}_y - \tilde{T}_x\right) \tag{19}$$

Following code shows how this function could be used to evaluate the expected years of widowhood for a wife within a couple. The example makes use of the Italian projected life tables ips55M and ips55F, whose derivation was shown in Section 4.2.

```
R> futureLifetimes <- as.data.frame(rLifexyz(n=numSim,
+                                         tablesList=list(husband=ips55M,wife=ips55F),
+                                         x=c(68,65), type="Tx"))
R> names(futureLifetimes) <- c("husband","wife")
R> temp <- futureLifetimes$wife - futureLifetimes$husband
R> futureLifetimes$widowance  <-  sapply(temp, max,0)
R> mean(futureLifetimes$widowance)
```

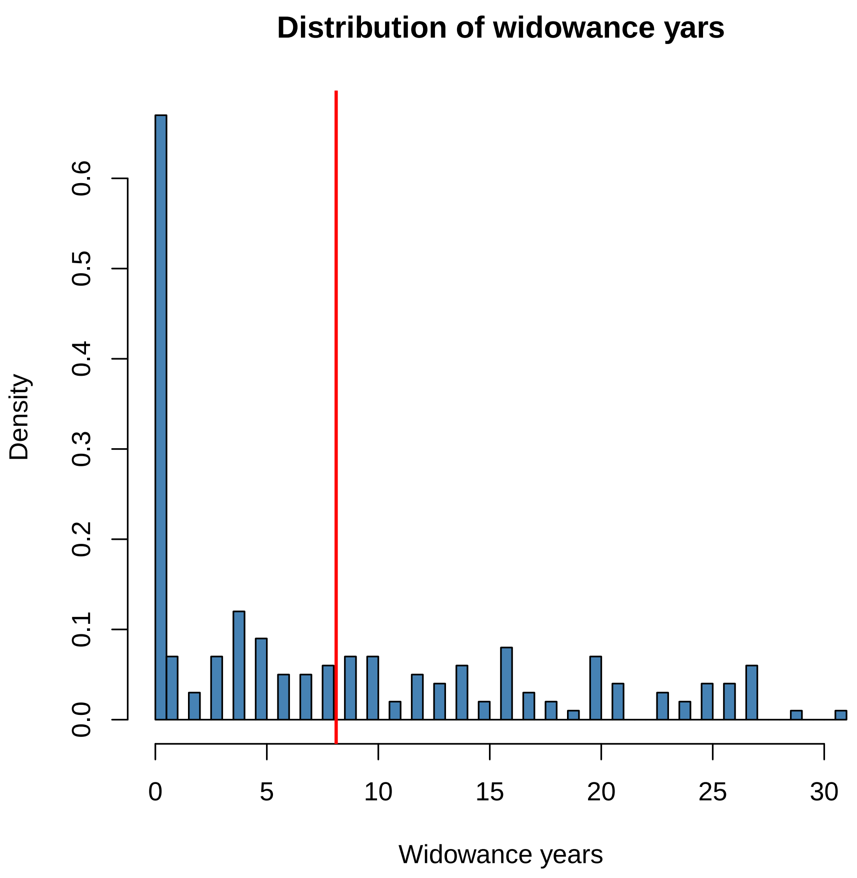

Figure 4: Years of widowance distribution, the red line represents the expected value.

```
[1] 8.11
```

Finally, Figure 4 shows the distribution of widowance years determined in previous example.

Actuarial mathematics examples

The present value of future benefits cash flows distribution, $\tilde{Z}$, is a random variable. It is a function of the interest rate and indicator variables depending by the life status of the insured. Both these quantities can be deemed as stochastic. However interest rates are considered deterministic within **lifecontingencies** package framework.
Generating n-size variates from $\tilde{Z}$ is performed by the following algorithm:

1. Define a function, PV, that returns the present value of the life contingent insurance benefits, given the age at death of the policyholder, T_0, $PV\left(T_0\right)$. Within **lifecontingencies** package, present value functions have been defined for most important life contingent insurances. Such functions are not visibly exported in package namespace.

2. Sample n variates from T_0.

3. Give T_0 variates as inputs to $PV\left(T_0\right)$ to get variates from $\tilde{Z}$.

Code below shows the internal function `.faxn` that returns the present value of an annuity life contingent insurance. `.faxn` is internally called by `rLifeContingencies` function, discussed below. `T`, `y`, `n`, `i`, `m`, `k` represent the age at death, the attained age, the term of the annuity, the interest rate, the deferring period as well as the fractional payment frequency respectively.

```
.faxn<-function(T,y,n, i, m, k=1)
{
        out <- numeric(1)
        K <- T-y
                if(K<m) {
                        out <- 0
                } else {
                  times <- seq(from=m, to=min(m+n-1/k,K),by=1/k)
                   out <- presentValue(cashFlows=rep(1/k, length(times)),
          timeIds=times, interestRates=i)
                }
        return(out)
}
```

Life contingencies insurance functions return the APV, that is $E\left[\tilde{Z}\right]$ as default value. Functions in Table 5 compute APVs by the current payment technique. Another possible approach to evalutate APVs, even if computationally inefficient, could be drawn a sample from the underlying $\tilde{Z}$ distribution and computing its sample mean.

Every function in Table 5 returns a sample of size one if the `type` parameter default value, "EV" (that stands for expected value), is overridden by the string "ST" (that stands for stochastic).
However, when samples of greater size are required, the most straightforward approach is the use of the `rLifeContingencies` function. Code below shows how to generate $\tilde{Z}$ variates from term life insurances, increasing term insurances, temporary annuity and endowment insurances respectively. For each example, the unbiaseness is verified by comparing the mean

of the sample with the theoretical APV using a classical t - test. All examples are referred to an individual aged 20 years old for an 40 years term insurance. Figure 5 shows the resulting $\tilde{Z}$ distributions.

```
R> APVAxn <- Axn(soa08Act,x=25,n=40,type="EV")
R> APVAxn

[1] 0.0479709

R> sampleAxn <- rLifeContingencies(n=numSim, lifecontingency="Axn",
+                   object=soa08Act,x=25,t=40,parallel=TRUE)
R> tt1<-t.test(x=sampleAxn,mu=APVAxn)$p.value
R> APVIAxn <- IAxn(soa08Act,x=25,n=40,type="EV")
R> APVIAxn

[1] 1.045507

R> sampleIAxn <- rLifeContingencies(n=numSim, lifecontingency="IAxn",
+                   object=soa08Act,x=25,t=40,parallel=TRUE)
R> tt2<-t.test(x=sampleIAxn,mu=APVIAxn)$p.value
R> APVaxn <- axn(soa08Act,x=25,n=40,type="EV")
R> APVaxn

[1] 15.46631

R> sampleaxn <- rLifeContingencies(n=numSim, lifecontingency="axn",
+                   object=soa08Act,x=25,t=40,parallel=TRUE)
R> tt3 <- t.test(x=sampleaxn,mu=APVaxn)$p.value
R> APVAExn <- AExn(soa08Act,x=25,n=40,type="EV")
R> APVAExn

[1] 0.1245488

R> sampleAExn <- rLifeContingencies(n=numSim, lifecontingency="AExn",
+                   object=soa08Act,x=25,t=40,parallel=TRUE)
R> tt4<-t.test(x=sampleAExn,mu=APVAExn)$p.value
R> c(tt1, tt2,tt3, tt4)

[1] 0.9902633 0.3826659 0.4360037 0.2228864
```

The full distribution of a life contingent insurance $\tilde{Z}$ variable can used to compute premiums using the percentile premium principle. Under this approach, the premium is set to ensure the insurer will suffer financial loss with sufficiently low probability (explicited by the percentile). An example will better exemplify the concept. For a 40 - year term insurance on a single policyholder aged 25, the actuarial present value of benefit, i.e. the expected value of discounted future benefits, would be

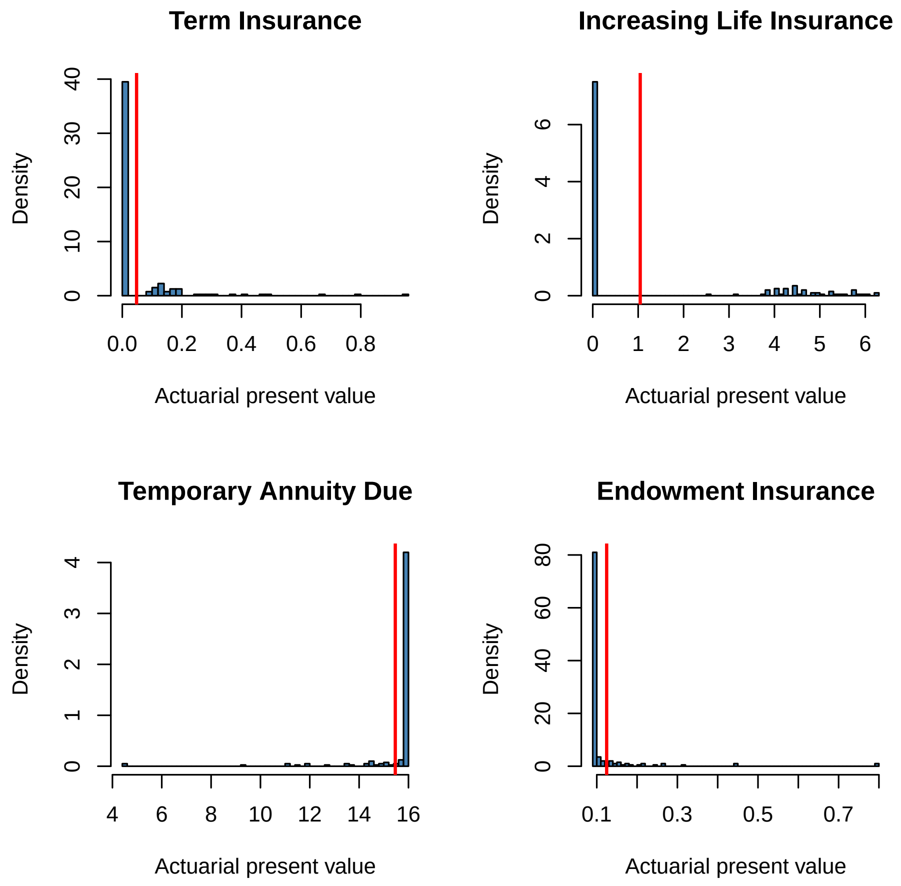

Figure 5: Life insurance stochastic variables distributions. Red vertical line represents APV.

```
R> APV <- Axn(actuarialtable = soa08Act, x=25, n=40)
R> APV

[1] 0.0479709
```

while the benefit premium at 90th percentile, that is the premium that would make the insurer to incurr in an underwriting loss with 10% of probability, would be

```
R> samples <- rLifeContingencies(n=numSim, lifecontingency = "Axn",
+                     object= soa08Act, x=25,t=40,paralle=TRUE)
R> pct90Pr <- quantile(samples,.90)
R> pct90Pr

      90%
0.1461862
```

Finally, if $N = 1000$ similar policyholders were insured, the law of large numbers would lead to a strong reduction in the premium charged on each policyholder, as computed below.

```
R> pct90Pr2 <- qnorm(p=0.90,mean=APV, sd=sd(samples)/sqrt(1000))
R> pct90Pr2

[1] 0.05250107
```

The final example of the paper shows how the stochastic functions bundled in **lifecontingencies** can be used to make an actuarial appraisal of embedded benefits as following example shows. Suppose a corporation grants its 100 employees a life insurance benefit equal to the annual salary, payable at the month of death. Suppose moreover that:

1. The expected value and the standard deviation of the salary are \$ 50,000 and \$ 15,000 respectively and salary distribution follows a log-normal distribution.
2. The employees distribution is uniform in the range 25 - 65. Assume 65 to be retirement age.
3. The SoA illustrative table represents an unbiased description of the population mortality.
4. Assume no lapse to hold.
5. The policy length is annual.

We evaluated the best estimate, that is the fair value of the insured benefits according to IFRS accounting standards (another word for benefit premium), and a risk margin measure. As risk margin measure, the difference between the 75th percentile and the best estimate will be used in this example. IFRS standards, Post, Grandl, Schmidl, and Dorfman (2007), define the fair value of an insurance liability as the sum of its best estimate plus its risk margin.

In the initial part of the example, we set out the parameter of the model and configure the parallel computation facility available by the package **parallel**. The code parallelization has been adapted from examples found in (McCallum and Weston 2011) textbook.

```
R> nsim <- 100
R> employees <- 100
R> salaryDistribution <- rlnorm(n=employees,m=10.77668944,s=0.086177696)
R> ageDistribution <- round(runif(n=employees,min=25, max=65))
R> policyLength <- sapply(65-ageDistribution, min, 1)
R> getEmployeeBenefit<-function(index,type="EV") {
+          out <- numeric(1)
+          out <- salaryDistribution[index]*Axn(actuarialtable=soa08Act,
+                          x=ageDistribution[index],n=policyLength[index],
+                          i=0.02,m=0,k=1, type=type)
+          return(out)
+  }
R> require(parallel)
R> cl <- makeCluster(detectCores())
R> worker.init <- function(packages) {
+          for (p in packages) {
+                  library(p, character.only=TRUE)
+          }
+          invisible(NULL)
+  }
R> clusterCall(cl,
+                  worker.init, c('lifecontingencies'))

[[1]]
NULL

[[2]]
NULL

R> clusterExport(cl, varlist=c("employees","getEmployeeBenefit",
+                                 "salaryDistribution","policyLength",
+                                 "ageDistribution","soa08Act"))
```

Then we perform best estimate and risk margin calculations.

```
R> employeeBenefits <- numeric(employees)
R> employeeBenefits <- parSapply(cl, 1:employees,getEmployeeBenefit, type="EV")
R> employeeBenefit <- sum(employeeBenefits)
R> benefitDistribution=numeric(nsim)
R> yearlyBenefitSimulate<-function(i)
+  {
+          out <- numeric(1)
+          expenseSimulation <- numeric(employees)
+          expenseSimulation <- sapply(1:employees, getEmployeeBenefit, type="ST")
+          out <- sum(expenseSimulation)
+          return(out)
+  }
```

```
R> benefitDistribution <- parSapply(cl, 1:nsim,yearlyBenefitSimulate )
R> stopCluster(cl)
R> riskMargin <- as.numeric(quantile(benefitDistribution,.75)-employeeBenefit)
R> totalBookedCost <- employeeBenefit+riskMargin
R> employeeBenefit

[1] 29011.67

R> riskMargin

[1] 18962.9

R> totalBookedCost

[1] 47974.57
```

5. Discussion

5.1. Advantages, limitations and future perspectives

The **lifecontingencies** package allows actuaries to perform demographic, financial and actuarial mathematics calculations within R software, in particular, pricing and reserving of life contingent insurance contracts. In addition, an peculiar feature of **lifecontingencies** is the ability to generate variates from future life time and life contingent insurances underlying stochastic distributions.

One of the most important limitations of **lifecontingencies** package, as of current release 0.9.7, is that only single decrements tables are currently handled. In addition, continuous - time life contingent models are currently not handled explicitely.

We expect to remove such limitations in the future. In addition, we expect to provide coerce methods toward packages specialized in demographic analysis, like **demography** and **LifeTables** packages as well as to make easier to share analyses with interest rates modelling packages like **termstrcR**.
Finally code optimization is continuously carried on. The extension of parallel computation features, memory usage profiling as well as the use of C code fragments in select parts of the code have been planned for the next future.

5.2. Accuracy

The accuracy of calculation have been verified by checking with numerical examples reported in Bowers *et al.* (1997) and in the lecture notes of Actuarial Mathematics the author attended years ago at Catholic University of Milan, Mazzoleni (2000). The numerical results are identical to those reported in the Bowers *et al.* (1997) textbook for most function, with the exception of fractional payments annuities where the accuracy leads only to the 5th decimal. The reason of such inaccuracy is due to the fact that the package calculates the APV by directly sum of fractional survival probabilities, while the results reported in Bowers *et al.* (1997) textbook are obtained by close formulas.

Finally, it is worth to remember that the package and functions herein are provided as is, without any guarantee regarding the accuracy of calculations. The author disclaims any liability arising by eventual losses due to direct or indirect use of this package.

Acknowledgments

The author wishes to thank all those whose suggestions contributed to the package enhancements, in particular to Christophe Dutang and Tim Riffle, for his suggestions and vignette revision and to the anonymous Journal of Statistical Software referees that have helped to improve the quality of the vignettes.

References

Anderson J (1999). *Commutation Functions*. Education and Examination Committee of the Society of Actuaries.

Bowers NL, Jones DA, Gerber HU, Nesbitt CJ, Hickman JC (1997). *Actuarial Mathematics, 2nd Edition*. SOA. ISBN 9780938959106.

Broverman S (2008). *Mathematics of Investment and Credit*. ACTEX Academic Series. ACTEX Publications. ISBN 9781566986571.

Chambers J (2008). *Software for Data Analysis: Programming with R*. Statistics and computing. Springer. ISBN 9780387759357.

Chris Ruckman, Joe Francis (2006). *Financial Mathematics: A Practical Guide for Actuaries and Other Business Professionals*. Warren Centre for Actuarial Studies and Research, second edition edition.

Christophe Dutang, Vincent Goulet, Mathieu Pigeon (2008). "**actuar**: An R Package for Actuarial Science." *Journal of Statistical Software*, **25**(7), 38. URL http://www.jstatsoft.org/v25/i07.

Dickson D, Hardy M, Waters H (2009). *Actuarial Mathematics for Life Contingent Risks*. International Series on Actuarial Science. Cambridge University Press. ISBN 9780521118255. URL http://books.google.it/books?id=maci14_9kAEC.

Ferstl R, Hayden J (2010). "Zero-Coupon Yield Curve Estimation with the Package **termstrc**." *Journal of Statistical Software*, **36**(1), 1–34. URL http://www.jstatsoft.org/v36/i01/.

Finan MA (2012). "A Reading of the Theory of Life Contingency Models: A Preparation for Exam MLC." `http://faculty.atu.edu/mfinan/actuarieshall/MLCbook2.pdf`. Accessed: 01/11/2012.

Gesmann M, Zhang Y (2011). ***ChainLadder**: Mack, Bootstrap, Munich and Multivariate-chain-ladder Methods.* R package version 0.1.4-3.4.

Guirreri S (2010). *Simulating the Term Structure of Interest Rates with Arbitrary Marginals.* Ph.D. thesis, University of Palermo - Department of Statistics and Mathematics "S. Vianelli", Palermo. URL `http://www.guirreri.host22.com`.

IBM Corp (2012). *SPSS, Release 21.0 Advanced Statistical Procedures Companion.* IBM Corp., Armonk, NY.

Keyfitz N, Caswell H (2005). *Applied Mathematical Demography.* Statistics for biology and health. Springer. ISBN 9780387225371.

Klugman S, Panjer H, Willmot G, Venter G (2009). *Loss Models: From Data to Decisions.* Third edition. Wiley New York.

Lee R, Carter L (1992). "Modeling and Forecasting U.S. Mortality." *Journal of the American Statistical Association*, **87**(419), 659–675. `doi:10.2307/2290201`.

MATLAB (2010). *Version 7.10.0 (R2010a).* The MathWorks Inc., Natick, Massachusetts.

Mazzoleni P (2000). *Appunti di Matematica Attuariale.* EDUCatt Università Cattolica. ISBN 9788883110825.

McCallum Q, Weston S (2011). *Parallel R.* O'Reilly Media. ISBN 9781449309923.

Post T, Grandl H, Schmidl L, Dorfman MS (2007). "Implications of IFRS for the European Insurance Industry Insights from Capital Market Theory." *Risk Management and Insurance Review*, **10**(2), 247–265. ISSN 1540-6296. `doi:10.1111/j.1540-6296.2007.00117.x`. URL `http://dx.doi.org/10.1111/j.1540-6296.2007.00117.x`.

R Development Core Team (2012). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL `http://www.R-project.org/`.

Riffe T (2011). *LifeTable: **LifeTable**, a Package with a Small Set of Useful Lifetable Functions.* R package version 1.0.1, URL `http://sites.google.com/site/timriffepersonal/r-code/lifeable`.

Rigby RA, Stasinopoulos DM (2005). "Generalized Additive Models for Location, Scale and Shape." *Applied Statistics*, **54**, 507–554.

Rob J Hyndman, Heather Booth, Leonie Tickle, John Maindonald (2011). ***demography**: Forecasting Mortality, Fertility, Migration and Population Data.* R package version 1.09-1, URL `http://CRAN.R-project.org/package=demography`.

SAS Institute Inc (2011). *SAS/STAT Software, Version 9.3.* Cary, NC. URL `http://www.sas.com/`.

Spedicato GA (2012). ***lifecontingencies**: An R Package to Perform Life Contingencies Calculations.* R package version 0.9.7, URL http://CRAN.R-project.org/package=lifecontingencies.

Zhang W (2011). ***cplm**: Monte Carlo EM Algorithms and Bayesian Methods for Fitting Tweedie Compound Poisson Linear Models.* R package version 0.2-1, URL http://CRAN.R-project.org/package=cplm.

Affiliation:

Giorgio Alfredo Spedicato
StatisticalAdvisor
Via Firenze 11 20037 Italy
Telephone: +39/334/6634384
E-mail: lifecontingencies@statisticaladvisor.com
URL: www.statisticaladvisor.com