An introduction to lifecontingencies package

Giorgio A. Spedicato

August 17, 2011

1 Overview

I've decided to submit to the cran package to perform life contingencies calculation in order to fill a current lack within the CRAN archive. I will fill this vignette further, nevertheless I anticipate the structure:

1. Section 2 provides an overview of R usage within actuarial fields and describes the package structure.
2. Section 3 gives a wide choice of lifecontingencies packages example.
3. Finally section will provide a discussion of results and further potential developments.

Please note these vignette are a work in progress.

2 Lifecontingencies package description

2.1 Current R actuarial packages

R [**?**] represents a powerful enviromnent for statistical analysis and simulation. Thus many packages dedicated to P&C actuarial software have been available from some years. Among those we shall cite:

- The package **actuar** [**?**] provides functions to fit relevant loss distribution and perform credibility analysis. It represents the computational side of the classical book [**?**].
- The package **chainladder** [**?**] provides functions to estimate non-life loss reserve.

The choice of statistical functions to perform rate - making is more wide as R provides a wide range of statistical function to perform classification and predictive modelling task (e.g. GLMs, data - mining techniques) performed by pricing actuaries.

Life actuaries works more with demographic and financial data. While R has a dedicated view to packages dedicated to financial analysis and few packages exists to perform demographic analysis (see for examples [**?**]) as of August 2011 no package exists to perform life contingencies calculation.

This is the reason I dediced to write functions to perform life contingencies calculation, even if my professional histories lies in the non-life side by far.

2.2 The structure of the package

The package contains R function, classes and methods to perform classical financial mathematics calculations, working with lifetable objects and classical life contingencies calculations.
Demos and vignettes (like this document) are also available.

The package is load within the R working environment as follows:

```
> library(lifecontingencies)
```

3 Examples

3.1 Classical financial mathematics example

Two examples will show classical financial mathematics applications of package lifecontingencies. The amortization of a loan and a savings account projection.

3.1.1 Loan amortization

```
> capital = 1e+05
> interest = 0.05
> payments_per_year = 2
> effectiveRate = (1 + interest)^(1/payments_per_year) - 1
> years = 10
> installment = capital/annuity(i = effectiveRate, periods = years *
+     payments_per_year)
> installment

[1] 6396.251

> balance_due = numeric(years * payments_per_year)
> balance_due[1] = capital * (1 + effectiveRate) - installment
> for (i in 2:length(balance_due)) {
+     balance_due[i] = balance_due[i - 1] * (1 + effectiveRate) -
+         installment
+     cat("Payment ", i, " balance due:", round(balance_due[i]),
+         "\n")
+ }

Payment  2  balance due: 92050
Payment  3  balance due: 87926
Payment  4  balance due: 83702
Payment  5  balance due: 79372
Payment  6  balance due: 74936
Payment  7  balance due: 70390
Payment  8  balance due: 65733
Payment  9  balance due: 60960
Payment  10  balance due: 56069
Payment  11  balance due: 51057
Payment  12  balance due: 45922
Payment  13  balance due: 40659
Payment  14  balance due: 35267
Payment  15  balance due: 29742
Payment  16  balance due: 24080
Payment  17  balance due: 18279
Payment  18  balance due: 12334
Payment  19  balance due: 6242
Payment  20  balance due: 0
```

3.1.2 Saving account projection

```
> cumulatedSavings <- function(amount, rate, periods) {
+     service_charge = 1
+     service_fee = (0.01 * min(100, amount) + 0.005 * max(0, min(50,
+         amount - 100)))
+     invested_amount = amount - service_charge - service_fee
+     out = invested_amount * accumulatedValue(interestRates = rate,
+         periods = periods)
+     return(out)
+ }
> savings_sequence = seq(from = 50, to = 300, by = 10)
> periods = 30 * 12
> yearly_rate = 0.025
> montly_effective_rate = (1 + yearly_rate)^(1/12) - 1
> cumulated_value = sapply(savings_sequence, cumulatedSavings,
+     montly_effective_rate, periods)
```

3.2 Working with lifetable and actuarial table objects

Lifetable objects represent the basic class designed to handle life table calculations needed to evaluate life contingencies. Actuarialtable class inherits from lifetable class.
Both have been designed using the S4 class framework. To build a lifetable class object three items are needed:

1. The years sequence, that is an integer sequence $0, 1, \ldots, \omega - 1$. It shall starts from zero and going to the $\omega - 1$ age (the age x as $p_x = 0$).
2. The l_x vector, that is the number of subjects living at the beginning of age x.
3. The name of the life table.

```
> x_example = seq(from = 0, to = 9, by = 1)
> lx_example = c(1000, 950, 850, 700, 680, 600, 550, 400, 200,
+     50)
> fakeLt = new("lifetable", x = x_example, lx = lx_example, name = "fake lifetable")
```

After a lifecontingencies table has been created, basic probability calculations may be performed.

```
> pxt(fakeLt, 2, 1)

[1] 0.8235294

> qxt(fakeLt, 3, 2)

[1] 0.1428571

> exn(fakeLt, 5, 2)

[1] 1.583333
```

A print (or show - equivalent) method is also available, reporting the x, lx, px and ex in tabular form.

```
> print(fakeLt)

Life table fake lifetable

   x   lx        px       ex
1  0 1000 0.9500000 5.980000
2  1  950 0.8947368 5.242105
3  2  850 0.8235294 4.741176
4  3  700 0.9714286 4.542857
5  4  680 0.8823529 3.647059
6  5  600 0.9166667 3.000000
7  6  550 0.7272727 2.181818
8  7  400 0.5000000 1.625000
9  8  200 0.2500000 1.250000
10 9   50 0.0000000 1.000000
```

An actuarialtable class inherits from the lifecontingencies class, but contains and additional slot: the interest rate slot.

```
> irate = 0.03
> fakeAct = new("actuarialtable", x = fakeLt@x, lx = fakeLt@lx,
+     interest = irate, name = "fake actuarialtable")
```

3.3 Classical actuarial mathematics examples

3.4 A thorough examples

4 Discussion

4.1 Analysis of software capabilities

4.2 Prospective developments