# Package ‘lifecontingencies’

June 11, 2022

**Type** Package  
**Title** Financial and Actuarial Mathematics for Life Contingencies  
**Version** 1.3.9  
**Maintainer** Giorgio Alfredo Spedicato <spedicato_giorgio@yahoo.it>  
**Description** Classes and methods that allow the user to manage life table, actuarial tables (also multiple decrements tables). Moreover, functions to easily perform demographic, financial and actuarial mathematics on life contingencies insurances calculations are contained therein. See Spedicato (2013) <doi:10.18637/jss.v055.i10>.  
**Depends** R (>= 4.0), methods  
**Imports** parallel, utils, markovchain, Rcpp (>= 0.12.18), stats  
**Suggests** demography, forecast, testthat, knitr, formatR, StMoMo, markdown  
**License** MIT + file LICENSE  
**Encoding** UTF-8  
**LazyLoad** yes  
**LazyData** true  
**BugReports** https://github.com/spedygiorgio/lifecontingencies/issues  
**BuildVignettes** yes  
**VignetteBuilder** utils, knitr  
**URL** https://github.com/spedygiorgio/lifecontingencies  
**LinkingTo** Rcpp  
**RoxygenNote** 7.2.0  
**NeedsCompilation** yes  
**Author** Giorgio Alfredo Spedicato [aut, cre] (<https://orcid.org/0000-0002-0315-8888>), Christophe Dutang [ctb] (<https://orcid.org/0000-0001-6732-1501>), Reinhold Kainhofer [ctb] (<https://orcid.org/0000-0002-7895-1311>), Kevin J Owens [ctb], Ernesto Schirmacher [ctb], Gian Paolo Clemente [ctb] (<https://orcid.org/0000-0001-6795-4595>), Ivan Williams [ctb]
## R topics documented:

<table>
<thead>
<tr>
<th>Package</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>lifecontingencies-package</td>
<td>3</td>
</tr>
<tr>
<td>accumulatedValue</td>
<td>5</td>
</tr>
<tr>
<td>actuarialtable-class</td>
<td>6</td>
</tr>
<tr>
<td>AExn</td>
<td>7</td>
</tr>
<tr>
<td>annuity</td>
<td>9</td>
</tr>
<tr>
<td>Axn</td>
<td>10</td>
</tr>
<tr>
<td>axn</td>
<td>12</td>
</tr>
<tr>
<td>Axn.mdt</td>
<td>13</td>
</tr>
<tr>
<td>axyn</td>
<td>14</td>
</tr>
<tr>
<td>Axyzn</td>
<td>16</td>
</tr>
<tr>
<td>DAxn</td>
<td>17</td>
</tr>
<tr>
<td>decreasingAnnuity</td>
<td>19</td>
</tr>
<tr>
<td>demoCanada</td>
<td>20</td>
</tr>
<tr>
<td>demoChina</td>
<td>21</td>
</tr>
<tr>
<td>demoFrance</td>
<td>22</td>
</tr>
<tr>
<td>demoGermany</td>
<td>22</td>
</tr>
<tr>
<td>demoIta</td>
<td>23</td>
</tr>
<tr>
<td>demoJapan</td>
<td>25</td>
</tr>
<tr>
<td>demoUk</td>
<td>26</td>
</tr>
<tr>
<td>demoUsa</td>
<td>27</td>
</tr>
<tr>
<td>de_angelis_di_falco</td>
<td>27</td>
</tr>
<tr>
<td>duration</td>
<td>28</td>
</tr>
<tr>
<td>Exn</td>
<td>29</td>
</tr>
<tr>
<td>exn</td>
<td>30</td>
</tr>
<tr>
<td>getDecrements</td>
<td>31</td>
</tr>
<tr>
<td>getLifecontingencyPv</td>
<td>32</td>
</tr>
<tr>
<td>getOmega</td>
<td>33</td>
</tr>
<tr>
<td>IAxn</td>
<td>34</td>
</tr>
<tr>
<td>Iaxn</td>
<td>36</td>
</tr>
<tr>
<td>increasingAnnuity</td>
<td>37</td>
</tr>
<tr>
<td>intensity2Interest</td>
<td>38</td>
</tr>
<tr>
<td>interest2Discount</td>
<td>39</td>
</tr>
<tr>
<td>Isn</td>
<td>40</td>
</tr>
<tr>
<td>lifetable-class</td>
<td>41</td>
</tr>
<tr>
<td>Lxt</td>
<td>43</td>
</tr>
<tr>
<td>mdt-class</td>
<td>44</td>
</tr>
<tr>
<td>multiple life probabilities</td>
<td>45</td>
</tr>
<tr>
<td>mx2qx</td>
<td>46</td>
</tr>
<tr>
<td>mxt</td>
<td>47</td>
</tr>
<tr>
<td>nominal2Real</td>
<td>48</td>
</tr>
<tr>
<td>presentValue</td>
<td>49</td>
</tr>
<tr>
<td>probs2lifetable</td>
<td>50</td>
</tr>
</tbody>
</table>
The lifecontingencies package performs standard financial, demographic and actuarial mathematics calculation. The main purpose of the package is to provide a comprehensive set of tools to perform risk assessment of life contingent insurances.

Details

Package: lifecontingencies
Type: Package
Version: 1.3.4
Date: 2018-04-01
License: GPL-2.0
LazyLoad: yes

Warning

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

Note

Work in progress.
Author(s)

Giorgio Alfredo Spedicato with contributions from Reinhold Kainhofer and Kevin J. Owens Maintainer: <spedicato_giorgio@yahoo.it>

References


See Also

accumulatedValue, annuity

Examples

```r
## financial mathematics example

# calculates monthly installment of a loan of 100,000,
# interest rate 0.05

i=0.05
monthlyInt=(1+i)^(1/12)-1
Capital=100000

R=1/12*Capital/annuity(i=i, n=10, k=12, type = "immediate")

balance=numeric(10*12+1)
capitals=numeric(10*12+1)
interests=numeric(10*12+1)
balance[1]=Capital
interests[1]=0
capitals[1]=0

for(i in (2:121)) {
  balance[i]=balance[i-1]*(1+monthlyInt)-R
  interests[i]=balance[i-1]*monthlyInt
  capitals[i]=R-interests[i]
}

loanSummary=data.frame(rate=c(0, rep(R,10*12)),
                        balance, interests, capitals)

head(loanSummary)
tail(loanSummary)

## actuarial mathematics example

# APV of an annuity
```
data(soaLt)
soa08Act=with(soaLt, new("actuarialtable",interest=0.06,
    x=x, lx=lx, name="SOA2008"))
#evaluate and life-long annuity for an aged 65
axn(soa08Act, x=65)

**accumulatedValue**  
*Function to evaluate the accumulated value.*

**Description**

This function returns the value at time \( n \) of a series of equally spaced payments of 1.

**Usage**

```r
accumulatedValue(i, n, m=0, k, type = "immediate")
```

**Arguments**

- **i**: Effective interest rate expressed in decimal form. E.g. 0.03 means 3%.
- **n**: Number of terms of payment.
- **m**: Deferring period, whose default value is zero.
- **k**: Frequency of payment.
- **type**: The Payment type, either "advance" for the annuity due (default) or "arrears" for the annuity immediate. Alternatively, one can use "due" or "immediate" respectively (can be abbreviated).

**Details**

The accumulated value is the future value of the terms of an annuity. Its mathematical expression is

\[
s_{\bar{a}} = (1 + i)^n \ a_{\bar{n}}
\]

**Value**

A numeric value representing the calculated accumulated value.

**Warning**

The function is provided as is, without any guarantee regarding the accuracy of calculation. We disclaim any liability for eventual losses arising from direct or indirect use of this software.

**Note**

Accumulated value are derived from annuities by the following basic equation

\[
s_{\bar{a}} = (1 + i)^n = a_{\bar{n}}.
\]
Author(s)
Giorgio A. Spedicato

References

See Also
annuity

Examples
# A man wants to save 100,000 to pay for his sons
# education in 10 years time. An education fund requires the investors to
# deposit equal installments annually at the end of each year. If interest of
# 0.075 is paid, how much does the man need to save each year in order to
# meet his target?
R = 100000/accumulatedValue(i = 0.075, n = 10)

Objects of class "actuarialtable" inherit the structure of class "lifetable" adding just the slot
for interest rate, interest.

Objects from the Class
Objects can be created by calls of the form new("actuarialtable", ...). Creation is the same
as lifetable objects creation, the slot for interest must be added too.

Slots
interest: Object of class "numeric" slot for interest rate, e.g. 0.03
x: Object of class "numeric" age slot
1x: Object of class "numeric" subjects at risk at age x
name: Object of class "character" name of the actuarial table

Extends
Class "lifetable", directly.
Methods

coerce signature(from = "actuarialtable", to = "data.frame"): moves from actuarialtable to data.frame

coerce signature(from = "actuarialtable", to = "numeric"): coerce from actuarialtable to a numeric

getOmega signature(object = "actuarialtable"): as for lifetable

print signature(x = "actuarialtable"): tabulates the actuarial commutation functions

show signature(object = "actuarialtable"): show method

summary signature(object = "actuarialtable"): prints brief summary

Warning

The function is provided as is, without any warranty regarding the accuracy of calculations. The author disclaims any liability for eventual losses arising from direct or indirect use of this software.

Note

The interest slot will handle time-varying interest rates in the future.

Author(s)

Giorgio A. Spedicato

References


See Also

axn, lifetable

Examples

showClass("actuarialtable")

AExn

*Function to evaluate the n-year endowment insurance*

Description

This function evaluates the n-year endowment insurance.

Usage

AExn(actuarialtable, x, n, i=actuarialtable@interest, k = 1, type = "EV", power=1)
Arguments

actuarialtable An actuarial table object.

x Insured age.

n Length of the insurance.

i Rate of interest. When missing the one included in the actuarialtable object is used.

k Frequency of benefit payment.

type A string, either "EV" for expected value of the actuarial present value (default) or "ST" for one stochastic realization of the underlying present value of benefits. Alternatively, one can use "expected" or "stochastic" respectively (can be abbreviated).

power The power of the APV. Default is 1 (mean)

Details

The n-year endowment insurance provides a payment either in the year of death or at the end of the insured period.

Value

A numeric value.

Note

When type="EV" the function calls both Axn and Exn.

Author(s)

Giorgio A. Spedicato

References


See Also

Axn, Exn

Examples

#Actuarial Mathematics book example
#check the actuarial equality on the expected values Exn+Axn=AExn
data(soa08Act)
AExn(soa08Act, x=35,n=30,i=0.06)
Exn(soa08Act, x=35,n=30,i=0.06)+Axn(soa08Act, x=35,n=30,i=0.06)
**Description**

Function to calculate present value of annuities-certain.

**Usage**

```r
annuity(i, n, m=0, k=1, type = "immediate")
```

**Arguments**

- `i`: Effective interest rate expressed in decimal form. E.g. 0.03 means 3%. It can be a vector of interest rates of the same length of periods.
- `n`: Periods for payments. If \( n = \text{infinity} \) then `annuity` returns the value of a perpetuity (either immediate or due).
- `m`: Deferring period, whose default value is zero.
- `k`: Yearly payments frequency. A payment of \( k-1 \) is supposed to be performed at the end of each year.
- `type`: The Payment type, either "advance" for the annuity due (default) or "arrears" for the annuity immediate. Alternatively, one can use "due" or "immediate" respectively (can be abbreviated).

**Details**

This function calculates the present value of a stream of fixed payments separated by equal interval of time. Annuity immediate has the first payment at time \( t=0 \), while an annuity due has the first payment at time \( t=1 \).

**Value**

A string, either "immediate" or "due".

**Note**

The value returned by `annuity` function derives from direct calculation of the discounted cash flow and not from formulas, like \( a^{(m)} 
\frac{n}{\tau(m)} = \frac{1-v^n}{\tau(m)} \). When \( m \) is greater than 1, the payment per period is assumed to be \( \frac{1}{m} \).

**Author(s)**

Giorgio A. Spedicato

**References**

See Also

accumulatedValue

Examples

# The present value of 5 payments of 1000 at one year interval that begins
# now when the interest rate is 2.5% is
1000*annuity(i=0.05, n=5, type = "due")
# A man borrows a loan of 20,000 to purchase a car at
# a nominal annual rate of interest of 0.06. He will pay back the loan through monthly
# installments over 5 years, with the first installment to be made one month
# after the release of the loan. What is the monthly installment he needs to pay?
R=20000/annuity(i=0.06/12, n=5*12)

Axn

Function to evaluate life insurance.

Description

This function evaluates n - years term and whole life insurance.

Usage

Axn(actuarialtable, x, n, i=actuarialtable@interest,
m, k=1, type = "EV", power=1, ...)

Arguments

actuarialtable An actuarial table object.
x Age of the insured. (can be a vector).
n Coverage period, if missing the insurance is considered whole life \( n = \omega - x - m \). (can be a vector).
i Interest rate (overrides the interest rate slot in actuarialtable). (should be a scalar).
m Deferring period, even fractional, if missing assumed to be 0. (can be a vector).
k Number of periods per year at the end of which the capital is payable in case of
insured event, default=1 (capital payable at the end of death year). (should be a scalar).
type A string, either "EV" for expected value of the actuarial present value (default)
or "ST" for one stochastic realization of the underlying present value of benefits.
Alternatively, one can use "expected" or "stochastic" respectively (can be abbreviated).
power The power of the APV. Default is 1 (mean)
... Arguments to be passed to pxt().
Details

The variance calculation has not been implemented yet.

Value

A numeric value representing either the actuarial value of the coverage (when type="EV") or a number drawn from the underlying distribution of Axn.

Warning

The function is provided as is, without any guarantee regarding the accuracy of calculation. We disclaim any liability for eventual losses arising from direct or indirect use of this software.

Note

It is possible that value returned by stochastic simulation are biased. Successive releases of this software will analyze the issue with detail.

Author(s)

Giorgio A. Spedicato

References


See Also

axn, Exn

Examples

#assume SOA example life table to be load
data(soaLt)
soa08Act=with(soaLt, new("actuarialtable",interest=0.06,
x=x, lx=lx, name="SOA2008"))

#evaluate the value of a 40 years term life insurance for an aged 25
Axn(actuarialtable=soa08Act, x=25, n=40)

#check an relevant life contingencies relationship
k=12
i=0.06
j=real2Nominal(i,k)
Axn(soa08Act, 30, k=12)
i/j=Axn(soa08Act, 30, k=1)
Description

This function calculates actuarial value of annuities, given an actuarial table. Fractional and deferred annuities can be evaluated. Moreover it can be used to simulate the stochastic distribution of the annuity value.

Usage

```r
axn(actuarialtable, x, n, i = actuarialtable@interest, m, k = 1, type = "EV", power=1,payment = "advance", ...)```

Arguments

- `actuarialtable`: An actuarial table object.
- `x`: Age of the annuitant. (can be a vector).
- `n`: Number of terms of the annuity, if missing annuity is intended to be paid until death. (can be a vector).
- `i`: Interest rate (default value the interest of the life table). (should be a scalar).
- `m`: Deferring period. Assumed to be 1 whether missing. (can be a vector).
- `k`: Number of fractional payments per period. Assumed to be 1 whether missing. (should be a scalar).
- `type`: A string, either "EV" for expected value of the actuarial present value (default) or "ST" for one stochastic realization of the underlying present value of benefits. Alternatively, one can use "expected" or "stochastic" respectively (can be abbreviated).
- `power`: The power of the APV. Default is 1 (mean)
- `payment`: The Payment type, either "advance" for the annuity due (default) or "arrears" for the annuity immediate. Alternatively, one can use "due" or "immediate" respectively (can be abbreviated).
- `...`: Arguments to be passed to `pxt()`.

Details

When "ST" has been selected a stochastic value representing a number drawn from the domain of

$$a^n_x$$

is drawn. "EV" calculates the classical APV.

Value

A numeric value.
Warning

The function is provided as is, without any warranty regarding the accuracy of calculations. The author disclaims any liability for eventual losses arising from direct or indirect use of this software.

Note

When either $x = \omega$ or $n = 0$ zero is returned.

Author(s)

Giorgio A. Spedicato

References


See Also

annuity, Exn

Examples

#assume SOA example life table to be load
data(soaLt)
soa08Act=with(soaLt, new("actuarialtable",interest=0.06,
x=x,lx=lx,name="SOA2008"))
#evaluate and life-long annuity for an aged 65
axn(soa08Act, x=65)

Description

Function to evaluate multiple decrement insurances

Usage

`Axn.mdt(object, x, n, i, decrement)`

Arguments

- `object`: an mdt or actuarialtable object
- `x`: policyholder's age
- `n`: contract duration
- `i`: interest rate
- `decrement`: decrement category
Value

The scalar representing APV of the insurance

Warning

The function is experimental and very basic. Testing is still needed. Use at own risk!

Examples

# creates a temporary mdt
myTable<-data.frame(x=41:43,lx=c(800,776,752),d1=rep(8,3),d2=rep(16,3))
myMdt<-new("mdt",table=myTable,name="ciao")
Axn.mdt(myMdt, x=41,n=2,i=.05,decrement="d2")

axyn  Functions to evaluate life insurance and annuities on two heads.

Description

These functions evaluates life insurances and annuities on two heads.

Usage

axyn(tablex, tabley, x, y, n, i, m, k = 1, status = "joint", type = "EV",
 payment="advance")
Axyn(tablex, x, tabley, y, n, i, m, k = 1, status = "joint", type = "EV")

Arguments

tablex  Life X lifetable object.
tabley  Life Y lifetable object.
x       Age of life X.
y       Age of life Y.
n       Insured duration. Infinity if missing.
i       Interest rate. Default value is those implied in actuarialtable.
m       Deferring period. Default value is zero.
k       Fractional payments or periods where insurance is payable.
status  Either "joint" for the joint-life status model or "last" for the last-survivor status model (can be abbreviated).
type    A string, either "EV" for expected value of the actuarial present value (default) or "ST" for one stochastic realization of the underlying present value of benefits. Alternatively, one can use "expected" or "stochastic" respectively (can be abbreviated).
payment The Payment type, either "advance" for the annuity due (default) or "arrears" for the annuity immediate. Alternatively, one can use "due" or "immediate" respectively (can be abbreviated).
Details

Actuarial mathematics book formulas has been implemented.

Value

A numeric value returning APV of chosen insurance form.

Warning

The function is provided as is, without any warranty regarding the accuracy of calculations. The author disclaims any liability for eventual losses arising from direct or indirect use of this software.

Note

Deprecated functions. Use Axyzn and axyzn instead.

Author(s)

Giorgio A. Spedicato

References


See Also

pxyt

Examples

```r
## Not run:
data(soa08Act)
#last survival status annuity
axyn(tablex=soa08Act, tabley=soa08Act, x=65, y=70, n=5, status = "last", type = "EV")
#first survival status annuity
Axyn(tablex=soa08Act, tabley=soa08Act, x=65, y=70, status = "last", type = "EV")

## End(Not run)
```
Multiple lifes insurances and annuities

Description

Function to evaluate the multiple lives insurances and annuities.

Usage

Axyzn(tablesList, x, n, i, m, k = 1, status = "joint", type = "EV", power=1)
axyzn(tablesList, x, n, i, m, k = 1, status = "joint", type = "EV", power=1, payment="advance")

Arguments

- `tablesList`: A list whose elements are either lifetable or actuarialtable class objects.
- `x`: A vector of the same size of tableList that contains the initial ages.
- `n`: Length of the insurance.
- `i`: Interest rate.
- `m`: Deferring period.
- `k`: Fractional payment frequency.
- `status`: Either "joint" for the joint-life status model or "last" for the last-survivor status model (can be abbreviated).
- `type`: A string, either "EV" for expected value of the actuarial present value (default) or "ST" for one stochastic realization of the underlying present value of benefits. Alternatively, one can use "expected" or "stochastic" respectively (can be abbreviated).
- `power`: The power of the APV. Default is 1 (mean).
- `payment`: The Payment type, either "advance" for the annuity due (default) or "arrears" for the annuity immediate. Alternatively, one can use "due" or "immediate" respectively (can be abbreviated).

Details

In theory, these functions apply the same concept of life insurances on one head on multiple heads.

Value

The insurance value is returned.

Note

These functions are the more general version of `axyn` and `Axyn`.
**Author(s)**
Giorgio Alfredo Spedicato, Kevin J. Owens.

**References**

**See Also**
axyn, Axyn.

**Examples**
```r
data(soaLt)
soa08Act=with(soaLt, new("actuarialtable", interest=0.06,
x=x, lx=Ix, name="SOA2008"))
# evaluate and life-long annuity for an aged 65
listOfTables=list(soa08Act, soa08Act)
# Check actuarial equality
axyzn(listOfTables, x=c(60,70), status="last")
axn(listOfTables[[1]],60)+axn(listOfTables[[2]],70)-axyzn(listOfTables, x=c(60,70), status="joint")
```

---

**DAxn**  
*Decreasing life insurance*

**Description**
This function evaluates the n-year term decreasing life insurance. Both actuarial value and stochastic random sample can be returned.

**Usage**
```
DAxn(actuarialtable, x, n, 
i=actuarialtable@interest,m = 0,k=1, 
type = "EV", power=1)
```

**Arguments**
- **actuarialtable**: An actuarial table object.
- **x**: Age of the insured.
- **n**: Length of the insurance period.
- **i**: Interest rate, when present it overrides the interest rate of the actuarial table object.
- **m**: Deferring period, even fractional, assumed 1 whether missing.
Number of fractional payments per period. Assumed to be 1 whether missing.

A string, either "EV" for expected value of the actuarial present value (default) or "ST" for one stochastic realization of the underlying present value of benefits. Alternatively, one can use "expected" or "stochastic" respectively (can be abbreviated).

The power of the APV. Default is 1 (mean)

Details
Formulas of Bowes book have been implemented.

Value
A numeric value representing the expected value or the simulated value.

Warning
The function is provided as is, without any guarantee regarding the accuracy of calculation. We disclaim any liability for eventual losses arising from direct or indirect use of this software.

Note
Neither fractional payments nor stochastic calculations have been implemented yet.

Author(s)
Giorgio A. Spedicato

References

See Also
Axn, IAxn

Examples
# using SOA illustrative life tables
data(soaLt)
soa08Act=with(soaLt, new("actuarialtable", interest=0.06,
x=x, lx=lx, name="SOA2008"))
# evaluate the value of a 10 years decreasing term life insurance for an aged 25
DAxn(actuarialtable=soa08Act, x=25, n=10)
decreasingAnnuity  

Function to evaluate decreasing annuities.

Description

This function return present values for decreasing annuities - certain.

Usage

decreasingAnnuity(i, n,type="immediate")

Arguments

i  
A numeric value representing the interest rate.

n  
The number of periods.

type  
The Payment type, either "advance" for the annuity due (default) or "arrears" for the annuity immediate. Alternatively, one can use "due" or "immediate" respectively (can be abbreviated).

Details

A decreasing annuity has the following flows of payments: n, n-1, n-2, ..., 1, 0.

Value

A numeric value reporting the present value of the decreasing cash flows.

Warning

The function is provided as is, without any guarantee regarding the accuracy of calculation. The author disclaims any liability for eventual losses arising from direct or indirect use of this software.

Note

This function calls presentValue function internally.

Author(s)

Giorgio A. Spedicato

References


See Also

annuity,increasingAnnuity,DAxn
Examples

# the present value of 10, 9, 8, ..., 0 payable at the end of the period
# for 10 years is
# decreasingAnnuity(i=0.03, n=10)
# assuming a 3% interest rate
# should be
sum((10:1)/(1+.03)^(1:10))

demoCanada

Description

UP94 life tables underlying mortality rates

Usage

data(demoCanada)

Format

A data frame with 120 observations on the following 7 variables.

x  age
up94M  UP 94, males
up94F  UP 94, females
up942015M  UP 94 projected to 2015, males
up942015F  UP 94 projected to 2015, females
up942020M  UP 94 projected to 2020, males
up942020F  UP 94 projected to 2020, females

Details

Mortality rates are provided.

Source

Courtesy of Andrew Botros

References

Courtesy of Andrew Botros
Examples

```r
data(demoCanada)
head(demoCanada)
# create the up94M life table
up94MLt <- probs2lifetable(probs = demoCanada$up94M, radix = 100000, "qx", name = "UP94")
# create the up94M actuarial table
up94MAct <- new("actuarialtable", lx = up94MLt@lx, x = up94MLt@x, interest = 0.02)
```

demoChina  

China Mortality Rates for life table construction

Description

Seven yearly mortality rates for each age

Usage

```r
data(demoChina)
```

Format

A data frame with 106 observations on the following 8 variables.

- `age`: Attained age
- `CL1`: CL1 rates
- `CL2`: CL2 rates
- `CL3`: CL3 rates
- `CL4`: CL4 rates
- `CL5`: CL5 rates
- `CL6`: CL6 rates
- `CL90-93`: CL 90-93 rates

Details

See the source link for details.

Source

Society of Actuaries

References

[https://mort.soa.org/](https://mort.soa.org/)

Examples

```r
data(demoChina)
tableChinaCL1 <- probs2lifetable(probs = demoChina$CL1, radix = 1000, type = "qx", name = "CHINA CL1")
```
**demoFrance**

*French population life tables*

**Description**

Illustrative life tables from French population.

**Usage**

```r
data(demoFrance)
```

**Format**

A data frame with 113 observations on the following 5 variables.

- **age**  Attained age
- **TH00_02**  Male 2000 life table
- **TF00_02**  Female 2000 life table
- **TD88_90**  1988 1990 life table
- **TV88_90**  1988 1990 life table

**Details**

These tables are real French population life tables. They regard 88 - 90 and 00 - 02 experience.

**Source**

Actuaris - Winter Associes

**Examples**

```r
data(demoFrance)
head(demoFrance)
```

---

**demoGermany**

*German population life tables*

**Description**

Dataset containing mortality rates for German population, male and females.

**Usage**

```r
data(demoGermany)
```
**demoIta**

**Format**

A data frame with 113 observations on the following 5 variables.

- \( x \) Attained age
- \( qx\text{Male} \) Male mortality rate
- \( qx\text{Female} \) Female mortality rate

**Details**

Sterbetafel DAV 1994

**Source**

Private communication

**Examples**

data(demoGermany)
head(demoGermany)

**Description**

This dataset reports five pairs of Italian population life tables. These table can be used to create life table objects and actuarial tables object.

**Usage**

data(demoIta)

**Format**

A data frame with 121 observations on the following 9 variables.

- \( X \) a numeric vector, representing ages from 0 to \( \omega \).
- \( \text{SIM02} \) a numeric vector, 2002 cross section general population males life table
- \( \text{SIF02} \) a numeric vector, 2002 cross section general population females life table
- \( \text{SIM00} \) a numeric vector, 2000 cross section general population males life table
- \( \text{SIF00} \) a numeric vector, 2000 cross section general population females life table
- \( \text{SIM92} \) a numeric vector, 1992 cross section general population males life table
- \( \text{SIF92} \) a numeric vector, 1992 cross section general population females life table
- \( \text{SIM81} \) a numeric vector, 1981 cross sectional general population males life table
- \( \text{SIF81} \) a numeric vector, 1981 cross sectional general population females life table
SIM61 a numeric vector, 1961 cross sectional general population males life table
SIF61 a numeric vector, 1961 cross sectional general population females life table
RG48M a numeric vector, RG48 projected males life table
RG48F a numeric vector, RG48 projected females life table
IPS55M a numeric vector, IPS55 projected males life table
IPS55F a numeric vector, IPS55 projected females life table
SIM71 a numeric vector, 1971 cross sectional general population males life table
SIM51 a numeric vector, 1951 cross sectional general population males life table
SIM31 a numeric vector, 1931 cross sectional general population males life table

Details

These table contains the vectors of survival at the beginning of life years and are the building block of both lifetable and actuarialtable classes.

Source

These tables comes from Italian national statistical bureau (ISTAT) for SI series, government Ministry of Economics (Ragioneria Generale dello Stato) for RG48 or from Insurers’ industrial association IPS55. RG48 represents the projected survival table for the 1948 born cohort, while IPS55 represents the projected survival table for the 1955 born cohort.

References

ISTAT, IVASS, Ordine Nazionale Attuari

Examples

```r
#load and show
data(demoIta)
head(demoIta)
#create sim92 life and actuarial table
lxsim92<-demoIta$SIM92
lxsim92<-lxsim92[!is.na(lxsim92) & lxsim92!=0]
xsim92<-seq(0,length(lxsim92)-1,1)
#create the table
sim92lt=new("lifetable",x=xsim92,lx=lxsim92,name="SIM92")
plot(sim92lt)
```
**demoJapan**

*Japan Mortality Rates for life table construction*

---

**Description**

Two yearly mortality rates for each age

**Usage**

```r
data(demoJapan)
```

**Format**

A data frame with 110 observations on the following 3 variables.

- **JP8587M** Male life table
- **JP8587F** Female life table
- **age** Attained age

**Details**

See the references link for details.

**Source**

Society of Actuaries mortality web site

**References**

[https://mort.soa.org/](https://mort.soa.org/)

**Examples**

```r
data(demoJapan)
head(demoJapan)
```
**UK life tables**

**Description**

AM and AF one year mortality rate. Series of 1992

**Usage**

```r
data(demoUk)
```

**Format**

A data frame with 74 observations on the following 3 variables:

- **Age**  Annuitant age
- **AM92** One year mortality rate (males)
- **AF92** One year mortality rate (males)

**Details**

This data set shows the one year survival rates for males and females of the 1992 series. It has been taken from the Institute of Actuaries. The series cannot be directly used to create a life table since neither rates are not provided for ages below 16 nor for ages over 90. Various approach can be used to complete the series.

**Source**

Institute of Actuaries

**References**


**Examples**

```r
data(demoUk)
head(demoUk)
```
demoUsa

*United States Social Security life tables*

**Description**

This data set contains period life tables for years 1990, 2000 and 2007. Both males and females life tables are reported.

**Usage**

demoUsa

**Format**

A `data.frame` containing people surviving at the beginning of "age" at 2007, 2000, and 1990 split by gender

**Details**

Reported age is truncated at the last age with lx>0.

**Source**

See [http://www.ssa.gov/oact/NOTES/as120/LifeTables_Body.html](http://www.ssa.gov/oact/NOTES/as120/LifeTables_Body.html)

**Examples**

data(demoUsa)
head(demoUsa)

de_angelis_di_falco

*Italian Health Insurance Data*

**Description**

A list of data.frames containing transition probabilities by age (row) and year of projections. Transitions are split by males and females, and show probabilities of survival, death and transitions from Healthy to Disabled.

**Usage**

de_angelis_di_falco

**Format**

a list containing eleven items (data.frames), and an mdt data object (HealthyMaleTable2013)
duration

Source
Paolo De Angelis, Luigi di Falco (a cura di). Assicurazioni sulla salute: caratteristiche, modelli attuariali e basi tecniche

Description
These functions evaluate the duration or the convexity of a series of cash flows

Usage

duration(cashFlows, timeIds, i, k = 1, macaulay = TRUE)
convexity(cashFlows, timeIds, i, k = 1)

Arguments

cashFlows A vector representing the cash flows amounts.
timeIds Cash flows times
i APR interest, i.e. nominal interest rate compounded m-thly.
k Compounding frequency for the nominal interest rate i.
macaulay Is the macaulay duration (default value) or the effective duration to be evaluated?

Details
The Macaulay duration is defined as \( \sum_t \frac{t \cdot CF_t \left(1 + \frac{i}{k}\right)^{-k \cdot t}}{P} \), while \( \sum_t t \cdot \left( t + \frac{1}{k} \right) \cdot CF_t \left(1 + \frac{i}{k}\right)^{-k \cdot t-2} \)

Value
A numeric value representing either the duration or the convexity of the cash flow series

Note
Vectorial interest rate are not handled yet.

Author(s)
Giorgio A. Spedicato

References
**Exn**

Function to evaluate the pure endowment

**Description**

Function to evaluate the pure endowment

**Usage**

```
Exn(actuarialtable, x, n, i = actuarialtable@interest, type = "EV", power = 1)
```

**Arguments**

- `actuarialtable`: An actuarial table object.
- `x`: Age of the insured.
- `n`: Length of the contract.
- `i`: Interest rate (it overwrites the `actuarialtable` one).
- `type`: A string, either "EV" (default value), "ST" (stochastic realization) or "VR" if the value of the variance is needed.
- `power`: The power of the APV. Default is 1 (mean).

**Value**

The APV of the contract

**Author(s)**

Giorgio A. Spedicato

**References**

See Also

  axn, Axn

Examples

# assumes SOA example life table to be load
data(soaLt)
soa08Act=with(soaLt, new("actuarialtable",interest=0.06, x=x, lx=Ix, name="SOA2008"))
# evaluate the pure endowment for a man aged 30 for a time span of 35
Exn(soa08Act, x=30, n=35)

<table>
<thead>
<tr>
<th>exn</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Expected residual life.</strong></td>
</tr>
</tbody>
</table>

Description

Expected residual life.

Usage

exn(object, x, n, type = "curtate")

Arguments

- **object**: A lifetable/actuarialtable object.
- **x**: Attained age
- **n**: Time until which the expected life should be calculated. Assumed omega - x whether missing.
- **type**: Either "Tx", "complete" or "continuous" for continuous future lifetime, "Kx" or "curtate" for curtate future lifetime (can be abbreviated).

Value

A numeric value representing the expected life span.

Author(s)

Giorgio Alfredo Spedicato

References


See Also

lifetable
getDecrements

Examples

# loads and show
data(soa08Act)
exn(object=soa08Act, x=0)
exn(object=soa08Act, x=0, type="complete")

getDecrements

Function to return the decrements defined in the mdt class

Description

This function lists the character decrements of the mdf class

Usage

getDecrements(object)

Arguments

object A mdt class object

Details

A character vector is returned

Value

A character vector listing the decrements defined in the class

Note

To be updated

Author(s)

Giorgio Spedicato

References

Marcel Finan A Reading of the Theory of Life Contingency Models: A Preparation for Exam MLC/3L

See Also

getOmega
getLifecontingencyPv

Examples

# create a new table
tableDecr=data.frame(d1=c(150,160,160),d2=c(50,75,85))
newMdt<-new("mdt",name="testMDT",table=tableDecr)
getDecrements(newMdt)

getLifecontingencyPv  Functions to obtain the present value of a life contingency given the time to death

Description

It returns the present value of a life contingency, specified by its APV symbol, known the time to death of the subjects

Usage

getLifecontingencyPv(deathsTimeX, lifecontingency, object, x, t, i = object@interest, m = 0, k = 1, payment = "advance")
getLifecontingencyPvXyz(deathsTimeXyz, lifecontingency, tablesList, x, t, i, m = 0, k = 1, status = "joint", payment = "advance")

Arguments

deadsTimeX  Time to death
lifecontingency  lifecontingency symbol
object  life table(s)
x  age(s) of the policyholder(s)
t  term of the contract
i  interest rate
m  deferrement
k  fractional payments

payment  The Payment type, either "advance" for the annuity due (default) or "arrears" for the annuity immediate. Alternatively, one can use "due" or "immediate" respectively (can be abbreviated).

status  Either "joint" for the joint-life status model or "last" for the last-survivor status model (can be abbreviated).

Details

This function is a wrapper to the many internal functions that give the PV known the age of death.
getOmega

Function to return the terminal age of a life table.

Description

This function returns the ω value of a life table object, that is, the last attainable age within a life table.

Usage

getOmega(object)
Arguments

object A life table object.

Value

A numeric value representing the $\omega$ value of a life table object

Warning

The function is provided as is, without any guarantee regarding the accuracy of calculation. We disclaim any liability for eventual losses arising from direct or indirect use of this software.

Author(s)

Giorgio A. Spedicato

References


See Also

actuarialtable

Examples

#assumes SOA example life table to be load
data(soaLt)
soa08=with(soaLt, new("lifetable",
x=x, lx=lx, name="SOA2008"))
#the last attainable age under SOA life table is
getOmega(soa08)

IAXn

Increasing life insurance

Description

This function evaluates the APV of an increasing life insurance. The amount payable at the end of year of death are: 1, 2, ..., n − 1, n. N can be set as $\omega - x - 1$.

Usage

IAXn(actuarialtable, x, n,i=actuarialtable@interest, m = 0, k=1, type = "EV", power=1)
Arguments

- **actuarialtable**: The actuarial table used to perform life-contingencies calculations.
- **x**: The age of the insured.
- **n**: The term of life insurance. If missing, \( n = \omega - x - m - 1 \).
- **i**: Interest rate (overrides the interest rate of the actuarialtable object).
- **m**: The deferring period. If missing, \( m \) is set as 0.
- **k**: Number of fractional payments per period. Assumed to be 1 whether missing.
- **type**: A string, either "EV" for expected value of the actuarial present value (default) or "ST" for one stochastic realization of the underlying present value of benefits. Alternatively, one can use "expected" or "stochastic" respectively (can be abbreviated).
- **power**: The power of the APV. Default is 1 (mean).

Details

The stochastic value feature has not been implemented yet.

Value

A numeric value.

Warning

The function is provided as is, without any guarantee regarding the accuracy of calculation. The author disclaims any liability for eventual losses arising from direct or indirect use of this software.

Author(s)

Giorgio A. Spedicato

References


See Also

- DAxn

Examples

```r
# assumes SOA example life table to be load
data(soaLt)
soa08Act=with(soaLt, new("actuarialtable",interest=0.06,
x=x, lx=Ix, name="SOA2008"))
# evaluate the value of a 10 years increasing term life insurance for an aged 25
```
Iaxn

**Description**

This function evaluates increasing annuities

**Usage**

```r
Iaxn(actuarialtable=soa08Act, x=25, n=10)
```

**Arguments**

- `actuarialtable` An actuarialtable object.
- `x` The age of the insured head.
- `n` The duration of the insurance
- `i` The interest rate that overrides the one in the actuarialtable object.
- `m` The deferring period.
- `type` Yet only "EV" is implemented.
- `power` The power of the APV. Default is 1 (mean)

**Details**

This actuarial mathematics is generally exoteric. I have seen no valid example of it.

**Value**

The APV of the insurance

**Warning**

The function is provided as is, without any guarantee regarding the accuracy of calculation. We disclaim any liability for eventual losses arising from direct or indirect use of this software.

**Note**

The function is provided as is, without any guarantee regarding the accuracy of calculation. We disclaim any liability for eventual losses arising from direct or indirect use of this software.

**Author(s)**

Giorgio A. Spedicato
increasingAnnuity

References


See Also

axn, IAxn

Examples

#using SOA illustrative life tables
data(soaLt)
soa08Act=with(soaLt, new("actuarialtable", interest=0.06,
x=x, lx=lx, name="SOA2008"))
#evaluate the value of a lifetime increasing annuity for a subject aged 80
Iaxn(actuarialtable=soa08Act, x=80, n=10)

increasingAnnuity  Increasing annuity.

Description

This function evaluates non-stochastic increasing annuities.

Usage

increasingAnnuity(i, n, type = "immediate")

Arguments

i  A numeric value representing the interest rate.

n  The number of periods.

type  The Payment type, either "advance" for the annuity due (default) or "arrears" for the annuity immediate. Alternatively, one can use "due" or "immediate" respectively (can be abbreviated).

Details

An increasing annuity shows the following flow of payments: 1, 2, \ldots, n - 1, n

Value

The value of the annuity.

Warning

The function is provided as is, without any guarantee regarding the accuracy of calculation. We disclaim any liability for eventual losses arising from direct or indirect use of this software.
Note
This function calls internally presentValue function.

Author(s)
Giorgio A. Spedicato

References

See Also
decreasingAnnuity, IAxn

Examples
# the present value of 1, 2, ..., n-1, n sequence of payments, # payable at the end of the period # for 10 periods is increasingAnnuity(i=0.03, n=10) # assuming a 3% interest rate

intensity2Interest  Functions to switch from interest to intensity and vice versa.

Description
There functions switch from interest to intensity and vice versa.

Usage
intensity2Interest(intensity)

interest2Intensity(i)

Arguments

intensity  Intensity rate
i  interest rate

Details
Simple financial mathematics formulas are applied.

Value
A numeric value.
interest2Discount

**Author(s)**
Giorgio A. Spedicato

**References**

**See Also**
real2Nominal, nominal2Real

**Examples**

```
# a force of interest of 0.02 corresponds to an APR of
interest2Intensity(intensity=0.02)
# an interest rate equal to 0.02 corresponds to a force of interest of
interest2Intensity(i=0.02)
```

---

**interest2Discount**  
*Functions to switch from interest to discount rates*

**Description**

These functions switch from interest to discount rates and vice-versa.

**Usage**

```
interest2Discount(i)
discunt2Interest(d)
```

**Arguments**

- `i`: Interest rate
- `d`: Discount rate

**Details**

The following formula (and its inverse) rules the relationships:

\[
\frac{i}{1+i} = d
\]

**Value**

A numeric value
Author(s)

Giorgio Alfredo Spedicato

References


See Also

intensity2Interest, nominal2Real

discount2Interest(d=0.04)

Isn

Function to calculated accumulated increasing annuity future value.

Description

This function evaluates non-stochastic increasing annuities future values.

Usage

Isn(i, n, type = "immediate")

Arguments

i
Interest rate.

n
Terms.

type
Either "due" for annuity due or "immediate" for annuity immediate.

Details

It calls increasingAnnuity after having capitalized by \((1 + i)^n\)

Value

A numeric value

Warning

The function is provided as is, without any guarantee regarding the accuracy of calculation. We disclaim any liability for eventual losses arising from direct or indirect use of this software.
Note
This function calls internally increasingAnnuity function.

Author(s)
Giorgio A. Spedicato

References

See Also
accumulatedValue

Examples
Isn(n=10,i=0.03)

Description
lifetable objects allow to define and use life tables with the aim to evaluate survival probabilities and mortality rates easily. Such values represent the building blocks used to estimate life insurances actuarial mathematics.

Objects from the Class
Objects can be created by calls of the form new("lifetable", ...). Two vectors are needed. The age vector and the population at risk vector.

Slots
x: Object of class "numeric", representing the sequence 0,1,...,\( \omega \)
1x: Object of class "numeric", representing the number of lives at the beginning of age \( x \). It is a non increasing sequence. The last element of vector x is supposed to be \( > 0 \).
name: Object of class "character", reporting the name of the table
Methods

- `coerce` signature(from = "lifetable", to = "data.frame"): method to create a data-frame from a lifetable object
- `coerce` signature(from = "lifetable", to = "markovchainList"): coerce method from lifetable to markovchainList
- `coerce` signature(from = "lifetable", to = "numeric"): brings to numeric
- `coerce` signature(from = "data.frame", to = "lifetable"): brings to life table
- `getOmega` signature(object = "lifetable"): returns the maximum attainable life age
- `plot` signature(x = "lifetable", y = "ANY"): plot method
- `head` signature(x = "lifetable"): head method
- `print` signature(x = "lifetable"): method to print the survival probability implied in the table
- `show` signature(object = "lifetable"): identical to plot method
- `summary` signature(object = "lifetable"): it returns summary information about the object

Warning

The function is provided as is, without any warranty regarding the accuracy of calculations. The author disclaims any liability for eventual losses arising from direct or indirect use of this software.

Note

t may be missing in pxt, qxt, ext. It assumes value equal to 1 in such case.

Author(s)

Giorgio A. Spedicato

References


See Also

actuarialtable

Examples

```
showClass("lifetable")
data(soa08)
summary(soa08)
#the last attainable age under SOA life table is
getOmega(soa08)
#head and tail
data(soaLt)
tail(soaLt)
head(soaLt)
```
Various demographic functions

Description

Various demographic functions

Usage

Lxt(object, x, t = 1, fxt = 0.5)

Tx(object, x)

Arguments

- object: a lifetable or actuarialtable object
- x: age of the subject
- t: duration of the calculation
- fxt: correction constant, default 0.5

Details

Tx is the sum of years lived since age x by the population of the life table, it is the sum of Lx. The function is provided as is, without any warranty regarding the accuracy of calculations. Use at own risk.

Value

A numeric value

Author(s)

Giorgio Alfredo Spedicato.

References


Examples

data(soaLt)
soa08Act = with(soaLt, new("actuarialtable", interest=0.06, x=x, lx=Ix, name="SOA2008"))
Lxt(soa08Act, 67, 10)
# assumes SOA example life table to be load
data(soaLt)
soa08Act = with(soaLt, new("actuarialtable", interest=0.06, x=x, lx=Ix, name="SOA2008"))
Tx(soa08Act, 67)
mdt-class

Class "mdt"

Description

A class to store multiple decrement tables

Objects from the Class

Objects can be created by calls of the form new("mdt", name, table, ...). They store absolute decrements

Slots

name: The name of the table
table: A data frame containing at least the number of decrements

Methods

getDecrements signature(object = "mdt"): return the name of decrements
getOmega signature(object = "mdt"): maximum attainable age
initialize signature(.Object = "mdt"): method to initialize the class
print signature(x = "mdt"): tabulate absolute decrement rates
show signature(object = "mdt"): show rates of decrement
coerce signature(from = "mdt", to = "markovchainList"): coercing to markovchainList objects
coerce signature(from = "mdt", to = "data.frame"): coercing to markovchainList objects
summary signature(object = "mdt"): it returns summary information about the object

Note

Currently only decrements storage of the class is defined.

Author(s)

Giorgio Spedicato

References

Marcel Finan A Reading of the Theory of Life Contingency Models: A Preparation for Exam MLC/3L

See Also

lifetable
multiple life probabilities

Examples

```r
# shows the class definition
showClass("mdt")
# create a new table
tableDecr=data.frame(d1=c(150,160,160), d2=c(50,75,85))
newMdt<-new("mdt", name="testMDT", table=tableDecr)
```

multiple life probabilities

*Functions to deals with multiple life models*

Description

These functions evaluate multiple life survival probabilities, either for joint or last life status. Arbitrary life probabilities can be generated as well as random samples of lifes.

Usage

```r
exyzt(tablesList, x, t = Inf, status = "joint", type = "Kx", ...)
pxyz(tablesList, x, t, status = "joint",
fractional = rep("linear", length(tablesList)), ...)
qxyz(tablesList, x, t, status = "joint",
fractional = rep("linear", length(tablesList)), ...)
```

Arguments

- `tablesList` A list whose elements are either lifetable or actuarialtable class objects.
- `x` A vector of the same size of tableList that contains the initial ages.
- `t` The duration.
- `status` Either "joint" for the joint-life status model or "last" for the last-survivor status model (can be abbreviated).
- `type` Either "Tx" for continuous future lifetime, "Kx" for curtate future lifetime (can be abbreviated).
- `fractional` Assumptions for fractional age. One of "linear", "hyperbolic", "constant force" (can be abbreviated).
- `...` Options to be passed to `pxt`.

Details

These functions extends `pxyt` family to an arbitrary number of life contingencies.
**Value**

An estimate of survival / death probability or expected lifetime, or a matrix of ages.

**Note**

The procedure is experimental.

**Author(s)**

Giorgio Alfredo, Spedicato

**References**


**See Also**

pxt, exn

**Examples**

```r
# assessment of curtate expectation of future lifetime of the joint-life status
genrate a sample of lifes
data(soaLt)
soa08Act = with(soaLt, new("actuarialtable", interest = 0.06, x = x, lx = Ix, name = "SOA2008"))
tables = list(males = soa08Act, females = soa08Act)
xVec = c(60, 65)
test = rLifexyz(n = 50000, tablesList = tables, x = xVec, type = "Kx")
# check first survival status
t.test(x = apply(test, 1, "min"), mu = exyzt(tablesList = tables, x = xVec, status = "joint"))
# check last survival status
t.test(x = apply(test, 1, "max"), mu = exyzt(tablesList = tables, x = xVec, status = "last"))
```

---

**mx2qx**

*Mortality rates to Death probabilities*

**Description**

Function to convert mortality rates to probabilities of death

**Usage**

```r
mx2qx(mx, ax = 0.5)
```

**Arguments**

- `mx`: mortality rates vector
- `ax`: the average number of years lived between ages `x` and `x + 1` by individuals who die in that interval
Details
  Function to convert mortality rates to probabilities of death

Value
  A vector of death probabilities

See Also
  mxt, qxt, qx2mx

Examples
  # using some recursion
  qx2mx(mx2qx(.2))

---

mxt  

Central mortality rate

Description
  This function returns the central mortality rate demographic function.

Usage
  mxt(object, x, t)

Arguments
  object  a lifetable or actuarialtable object
  x       subject’s age
  t       period on which the rate is evaluated

Value
  A numeric value representing the central mortality rate between age \( x \) and \( x + t \).

References
Examples

```r
# assumes SOA example life table to be load
data(soaLt)
soa08Act = with(soaLt, new("actuarialtable", interest=0.06, x=x, lx=Ix, name="SOA2008"))
# compare mx and qx
mxt(soa08Act, 60, 10)
qxt(soa08Act, 60, 10)
```

### nominal2Real

Functions to switch from nominal / effective / convertible rates

#### Description

Functions to switch from nominal / effective / convertible rates

#### Usage

```r
nominal2Real(i, k = 1, type = "interest")
convertible2Effective(i, k = 1, type = "interest")
real2Nominal(i, k = 1, type = "interest")
effective2Convertible(i, k = 1, type = "interest")
```

#### Arguments

- `i`: The rate to be converted.
- `k`: The original / target compounting frequency.
- `type`: Either "interest" (default) or "nominal".

#### Details

effective2Convertible and convertible2Effective wrap the other two functions.

#### Value

A numeric value.

#### Note

Convertible rates are synonims of nominal rates

#### References

`presentValue`  

See Also  

`real2Nominal`  

Examples  

```r  
# a nominal rate of 0.12 equates an APR of  
nominal2Real(i=0.12, k = 12, "interest")  
```

## Description  
This function evaluates the present values of a series of cash flows, given occurrence time. Probabilities of occurrence can also be taken into account.

## Usage  
```r  
presentValue(cashFlows, timeIds, interestRates, probabilities, power = 1)  
```

## Arguments  
- `cashFlows`: Vector of cashFlow, must be coherent with `timeIds`
- `timeIds`: Vector of points of time where cashFlows are due.
- `interestRates`: A numeric value or a time-size vector of interest rate used to discount cash flows.
- `probabilities`: Optional vector of probabilities.
- `power`: Power to square discount and cash flows. Default is set to 1

## Details  
`probabilities` is optional, a sequence of 1 length of `timeIds` is assumed. Interest rate shall be a fixed number or a vector of the same size of `timeIds`. `power` parameters is generally useless beside life contingencies insurances evaluations.

## Value  
A numeric value representing the present value of `cashFlows` vector, or the actuarial present value if `probabilities`

## Note  
This simple function is the kernel working core of the package. Actuarial and financial mathematics ground on it.

## References  
Examples

# simple example
cf=c(10,10,10) #$10 of payments one per year for three years
t=c(1,2,3) #years
p=c(1,1,1) # assume payments certainty
# assume 3% of interest rate
presentValue(cashFlows=cf, timeIds=t, interestRates=0.03, probabilities=p)

probs2lifetable

Life table from probabilities

Description

This function returns a newly created lifetable object given either survival or death (one year) probabilities.

Usage

probs2lifetable(probs, radix = 10000, type = "px", name = "ungiven")

Arguments

probs A real valued vector representing either one year survival or death probabilities. The last value in the vector must be either 1 or 0, depending if it represents death or survival probabilities respectively.
radix The radix of the life table.
type Character value either "px" or "qx" indicating how probabilities must be interpreted.
name The character value to be put in the corresponding slot of returned object.

Details

The $\omega$ value is the length of the probs vector.

Value

A lifetable object.

Warning

The function is provided as is, without any guarantee regarding the accuracy of calculation. We disclaim any liability for eventual losses arising from direct or indirect use of this software.

Note

This function allows to use mortality projection given by other softwares with the lifecontingencies package.
Author(s)

Giorgio A. Spedicato

References


See Also

actuarialtable

Examples

fakeSurvivalProbs=seq(0.9,0,by=-0.1)
newTable=probs2lifetable(fakeSurvivalProbs,type="px",name="fake")
head(newTable)
tail(newTable)

pxt

Functions to evaluate survival, death probabilities and deaths.

Description

These functions evaluate raw survival and death probabilities between age x and x+t.

Usage

dxt(object, x, t, decrement)
pxt(object, x, t, fractional = "linear", decrement)
qxt(object, x, t, fractional = "linear", decrement)

Arguments

object A lifetable object.
x Age of life x. (can be a vector for pxt, qxt).
t Period until which the age shall be evaluated. Default value is 1. (can be a vector for pxt, qxt).
fractional Assumptions for fractional age. One of "linear", "hyperbolic", "constant force" (can be abbreviated).
decrement The reason of decrement (only for mdt class objects). Can be either an ordinal number or the name of decrement.
Details

Fractional assumptions are:

- **linear**: linear interpolation between consecutive ages, i.e. assume uniform distribution.
- **constant force of mortality**: constant force of mortality, also known as exponential interpolation.
- **hyperbolic**: Balducci assumption, also known as harmonic interpolation.

Note that fractional="uniform", "exponential", "harmonic" or "Balducci" is also authorized. See references for details.

Value

A numeric value representing requested probability.

Warning

The function is provided as is, without any warranty regarding the accuracy of calculations. The author disclaims any liability for eventual losses arising from direct or indirect use of this software.

Note

Function dxt accepts also fractional value of t. Linear interpolation is used in such case. These functions are called by many other functions.

Author(s)

Giorgio A. Spedicato

References


See Also

exn, lifetable

Examples

```r
dxt example
data(soa08Act)
dxt(object=soa08Act, x=90, t=2)
#x example
qxt(object=soa08Act, x=90, t=2)
#p example
pxt(object=soa08Act, x=90, t=2, "constant force")
#add another example for MDT
```
Functions to evaluate joint survival probabilities.

Description

These functions evaluate survival and death probabilities for two heads.

Usage

\[
\begin{align*}
\text{exyt}(\text{objectx}, \text{objecty}, x, y, t, \text{status} = \text{"joint"}) \\
\text{pxyt}(\text{objectx}, \text{objecty}, x, y, t, \text{status} = \text{"joint"}) \\
\text{qxyt}(\text{objectx}, \text{objecty}, x, y, t, \text{status} = \text{"joint"})
\end{align*}
\]

Arguments

- \text{objectx} lifetable for life X.
- \text{objecty} lifetable for life Y.
- \text{x} Age of life X.
- \text{y} Age of life Y.
- \text{t} Time until survival has to be evaluated.
- \text{status} Either "joint" for the joint-life status model or "last" for the last-survivor status model (can be abbreviated).

Value

A numeric value representing joint survival probability.

Warning

The function is provided as is, without any warranty regarding the accuracy of calculations. The author disclaims any liability for eventual losses arising from direct or indirect use of this software. Also it is being Deprecated and asap removed from the package.

Note

These functions are used to evaluate two or more life contingencies.

Author(s)

Giorgio A. Spedicato, Kevin J. Owens.
References


See Also

exyt

Examples

## Not run:
data(soa08Act)
pxyt(soa08Act, soa08Act, 65, 70, 10)
pxyt(soa08Act, soa08Act, 65, 70, 10, "last")

## End(Not run)

qx2mxDeath Probabilities to Mortality Rates

Description

Function to convert death probabilities to mortality rates

Usage

qx2mx(qx, ax = 0.5)

Arguments

qx death probabilities
ax the average number of years lived between ages x and x +1 by individuals who die in that interval

Details

Function to convert death probabilities to mortality rates

Value

A vector of mortality rates

See Also

mxt, qxt, mx2qx
Examples

data(soa08Act)
soa08qx<-as(soa08Act,"numeric")
soa08mx<-qx2mx(qx=soa08qx)
soa08qx2<-mx2qx(soa08mx)

qxt.prime.fromMdt

Return Associated single decrement from absolute rate of decrement

Description

Return Associated single decrement from absolute rate of decrement

Usage

qxt.prime.fromMdt(object, x, t = 1, decrement)

qxt.fromQxprime(qx.prime, other.qx.prime, t = 1)

Arguments

object      a mdj object
x           age
 t          period (default 1)
 decrement   type (necessary)
 qx.prime    single ASDT decrement of which corresponding decrement is desired
 other.qx.prime ASDT decrements other than qx.prime

Value

a single value (AST)

Functions

• qxt.fromQxprime: Obtain decrement from single decrements

Examples

#Creating the valdez mdf
valdezDf<-data.frame(
x=c(50:54),
 lx=c(4832555,4821937,4810206,4797185,4782737),
 hearth=c(5168, 5363, 5618, 5929, 6277),
 accidents=c(1157, 1206, 1443, 1679,2152),
 other=c(4293,5162,5960,6840,7631))
valdezMdt<-new("mdt",name="ValdezExample",table=valdezDf)
qxt.prime.fromMdt(object=valdezMdt,x=53,decrement="other")

# Finan example 67.2
qxt.fromQxprime(qx.prime = 0.01,other.qx.prime = c(0.03,0.06))

---

**rLifeContingencies**  
*Function to generate samples from the life contingencies stochastic variables*

**Description**

Function to generate samples from the life contingencies stochastic variables

**Usage**

```r
rLifeContingencies(
  n,  
  lifecontingency,  
  object,  
  x,  
  t,  
  i = object@interest,  
  m = 0,  
  k = 1,  
  parallel = FALSE,  
  payment = "advance"
)
```

```r
rLifeContingenciesXyz(
  n,  
  lifecontingency,  
  tablesList,  
  x,  
  t,  
  i,  
  m = 0,  
  k = 1,  
  status = "joint",  
  parallel = FALSE,  
  payment = "advance"
)
```
Arguments

- **n**: Size of sample
- **lifecontingency**: A character string, either "Exn", "Axn", "axn", "IAxn" or "DAxn"
- **object**: An actuarialtable object.
- **x**: Policyholder’s age at issue time; for rLifeContingenciesXyz a numeric vector of the same length of object, containing the policyholders’ ages
- **t**: The length of the insurance. Must be specified according to the present value of benefits definition.
- **i**: The interest rate, whose default value is the actuarialtable interest rate slot value.
- **m**: Deferring period, default value is zero.
- **k**: Fractional payment, default value is 1.
- **parallel**: Uses the parallel computation facility.
- **payment**: The Payment type, either "advance" for the annuity due (default) or "arrears" for the annuity immediate. Alternatively, one can use "due" or "immediate" respectively (can be abbreviated).
- **tablesList**: A list of actuarialtable objects
- **status**: Either "joint" for the joint-life status model or "last" for the last-survivor status model (can be abbreviated).

Value

A numeric vector

Examples

```r
## Not run:
# assumes SOA example life table to be load
data(soaLt)
soa08Act=with(soaLt, new("actuarialtable",interest=0.06, x=x, lx=Ix, name="SOA2008"))
out<-rLifeContingencies(n=1000, lifecontingency="Axn", object=soa08Act, x=40,
t=getOmega(soa08Act)-40, m=0)
APV=Axn(soa08Act,x=40)
# check if out distribution is unbiased
t.test(x=out, mu=APV)$p.value>0.05

## End(Not run)
## Not run:
data(soa08Act)
n=10000
lifecontingency="Axyz"
tablesList=list(soa08Act,soa08Act)
x=c(60,60); i=0.06; m=0; status="joint"; t=30; k=1
APV=Axyzn(tablesList=tablesList,x=x,n=t,m=m,k=k,status=status,type="EV")
samples=rLifeContingenciesXyz(n=n,lifecontingency = lifecontingency,tablesList = tablesList,
x=x,t=t,m=m,k=k,status=status, parallel=FALSE)
```
**Description**

Function to generate random future lifetimes

**Usage**

```
rLife(n, object, x = 0, k = 1, type = "Tx")
rLifexyz(n, tablesList, x, k = 1, type = "Tx")
```

**Arguments**

- `n`: Number of variates to generate
- `object`: An object of class lifetable
- `x`: The attained age of subject x, default value is 0
- `k`: Number of periods within the year when it is possible death to happen, default value is 1
- `type`: Either "Tx" for continuous future lifetime, "Kx" for curtate future lifetime (can be abbreviated).
- `tablesList`: An list of lifetables

**Details**

Following relation holds for the future life time: \( T_x = K_x + 0.5 \)

**Value**

A numeric vector of n elements.

**Note**

The function is provided as is, without any warranty regarding the accuracy of calculations. The author disclaims any liability for eventual losses arising from direct or indirect use of this software.

**References**

Simulate from a multiple decrement table

rmdt

Description

Simulate from a multiple decrement table

Usage

rmdt(n = 1, object, x = 0, t = 1, t0 = "alive", include.t0 = TRUE)

Arguments

n
  Number of simulations.

object
  The mdt object to simulate from.

x
  the period to simulate from.

t
  the period until to simulate.

t0
  initial status (default is "alive").

include.t0
  should initial status to be included (default is TRUE)?

See Also

lifetable, exn

Examples

## Not run:
##get 20000 random future lifetimes for the Soa life table at birth
data(soa08Act)
lifes=rLife(n=20000,object=soa08Act, x=0, type="Tx")
check if the expected life at birth derived from the life table is statistically equal
to the expected value of the sample
t.test(x=lifes, mu=exn(soa08Act, x=0, type="continuous"))

## End(Not run)

## Not run:
##assessment of curtate expectation of future lifetime of the joint-life status
##generate a sample of lifes
data(soaLt)
soa08Act=with(soaLt, new("actuarialtable",interest=0.06,x=x,lx=Ix,name="SOA2008"))
tables=list(males=soa08Act, females=soa08Act)
xVec=c(60,65)
test=rLifexyz(n=50000, tablesList = tables, x=xVec, type="Kx")
#check first survival status
t.test(x=apply(test,1,"min"),mu=exyzt(tablesList=tables, x=xVec, status="joint"))
#check last survival status
t.test(x=apply(test,1,"max"),mu=exyzt(tablesList=tables, x=xVec, status="last"))

## End(Not run)
Value
A matrix with n columns (the length of simulation) and either t (if initial status is not included) or 
t+1 rows.

Details
The function uses `rmarkovchain` function from markovchain package to simulate the chain

Author(s)
Giorgio Spedicato

See Also
`rLifeContingenciesXyz`, `rLifeContingencies`

Examples
```r
mdtDf<-data.frame(x=c(0,1,2,3),death=c(100,50,30,10),lapse=c(150,20,2,0))
myMdt<-new("mdt",name="example Mdt",table=mdtDf)
ciao<-rmdt(n=5,object = myMdt,x = 0,t = 4,include.t0=FALSE,t0="alive")
```

---

**soa08**  
*Society of Actuaries Illustrative Life Table object.*

Description
This is the table that appears in the classical book Actuarial Mathematics in Appendix 2A and used 
throughout the book to illustrate life contingent calculations. The Society of Actuaries has been 
using this table when administering US actuarial professional MLC preliminary examinations.

Usage
```r
data(soa08)
```

Format
Formal class 'lifetable' [package "lifecontingencies"] with 3 slots ..@ x : int [1:141] 0 1 2 3 4 5 6 7 
8 9 ... ..@ lx : num [1:141] 100000 97958 97826 97707 97597 ... ..@ name: chr "SOA Illustrative 
Life Table"
Details

This table is a blend of Makeham’s mortality law for ages 13 and above and some ad hoc values for ages 0 to 12.

The parameters for Makeham’s mortality law are

\[ 1000 \times \mu(x) = 0.7 + 0.05 \times 10^{0.04 \times x} \]

where \( \mu(x) \) is the force of mortality.

The published Illustrative Life Table just shows ages 0 to 110 but in the computing exercises of chapter 3 the authors explain that the table’s age range is from 0 to 140.

Note

This table is based on US 1990 general population mortality.

References


Examples

```r
data(soa08)
```
Details

This table is a blend of Makeham’s mortality law for ages 13 and above and some ad hoc values for ages 0 to 12.

The parameters for Makeham’s mortality law are

\[
1000 \times \mu(x) = 0.7 + 0.05 \times 10^{0.04 \times x}
\]

where \( \mu(x) \) is the force of mortality.

The published Illustrative Life Table just shows ages 0 to 110 but in the computing exercises of chapter 3 the authors explain that the table’s age range is from 0 to 140.

References


Examples

```r
## Not run:
data(soa08Act)

## End(Not run)
```

<table>
<thead>
<tr>
<th>SoAISTdata</th>
<th>SoA illustrative service table</th>
</tr>
</thead>
</table>

Description

Bowers’ book Illustrative Service Table

Usage

data(SoAISTdata)

Format

A data frame with 41 observations on the following 6 variables.

- \( x \) Attained age
- \( 1x \) Surviving subjects at the beginning of each age
- \( \text{death} \) Drop outs for death cause
- \( \text{withdrawal} \) Drop outs for withdrawal cause
- \( \text{inability} \) Drop outs for inability cause
- \( \text{retirement} \) Drop outs for retirement cause

Details

It is a data frame that can be used to create a multiple decrement table
soaLt

Source
Optical recognized characters from below source with some few adjustments

References

Examples
data(SoAISTdata)
head(SoAISTdata)

soaLt  Society of Actuaries life table

Description
This table has been used by the classical book Actuarial Mathematics and by the Society of Actuaries for US professional examinations.

Usage
data(soaLt)

Format
A data.frame with 111 obs on the following 2 variables:
x  a numeric vector
Ix  a numeric vector

Details
Early ages have been found elsewhere since miss in the original data sources; SOA did not provide population at risk data for certain spans of age (e.g. 1-5, 6-9, 11-14 and 16-19)

References

Examples
data(soaLt)
head(soaLt)
Description

Uk AM AF life tables

Usage

data(AF92Lt)

Format

The format is: Formal class 'lifetable' [package ".GlobalEnv"] with 3 slots ..@ x : int [1:111] 0 1 2 3 4 5 6 7 8 9 ... ..@ lx : num [1:111] 100000 99924 99847 99770 99692 ... ..@ name: chr "AF92"

Details

Probabilities for earliest (under 16) and lastest ages (over 92) have been derived using a Brass - Logit model fit on Society of Actuaries life table.

Source

See Uk life table.

References


Examples

data(AF92Lt)
exn(AF92Lt)
data(AM92Lt)
exn(AM92Lt)
Index

* classes
  actuarialtable-class, 6
  lifetable-class, 41
  mdt-class, 44

* datasets
  de_angelis_di_falco, 27
demoCanada, 20
demoChina, 21
demoFrance, 22
demoGermany, 22
demoIta, 23
demoJapan, 25
demoUK, 26
demoUsa, 27
soa08, 60
soa08Act, 61
SoAISTdata, 62
soaLt, 63
Uk life tables, 64

accumulatedValue, 4, 5, 10, 41
actuarialtable, 24, 34, 42, 51
actuarialtable-class, 6
AExn, 7
AF92Lt (Uk life tables), 64
AM92Lt (Uk life tables), 64
annuity, 4, 6, 9, 13, 19, 29
Axn, 8, 10, 18, 30
axn, 7, 11, 12, 30, 37
Axn.mdt, 13
Ayn, 16, 17
Ayn (axyn), 14
axyn, 14, 16, 17
Axyz, 16
axyzn (Axyz), 16

coeerce,actuarialtable.data.frame-method (actuarialtable-class), 6
coeerce,actuarialtable.numeric-method (actuarialtable-class), 6
coeerce,data.frame,lifetable-method (lifetable-class), 41
coeerce,lifetable.data.frame-method (lifetable-class), 41
coeerce,lifetable.markovchainList-method (lifetable-class), 41
coeerce,lifetable.numeric-method (lifetable-class), 41
coeerce,mdt.data.frame-method (mdt-class), 44
coerce,mdt.markovchainList-method (mdt-class), 44
convertible2Effective (nominal2Real), 48
convexity (duration), 28
DAxn, 17, 19, 35
de_angelis_di_falco, 27
decreasingAnnuity, 19, 38
demoCanada, 20
demoChina, 21
demoFrance, 22
demoGermany, 22
demoIta, 23
demoJapan, 25
demoUK, 26
demoUsa, 27
discount2Interest (interest2Discount), 39
duration, 28
dxt (pxt), 51
effective2Convertible (nominal2Real), 48
Exn, 8, 11, 13, 29
exn, 30, 46, 52, 59
exyt, 54
exyt (pxyt), 53
exyzt (multiple life probabilities), 45
getDecrements, 31
getDecrements,mdt-method (mdt-class), 44
getLifecontingencyPv, 32
getLifecontingencyPvXyz
  (getLifecontingencyPv), 32
getOmega, 31, 33
getOmega, actuarialtable-method
  (actuarialtable-class), 6
getOmega, lifetable-method
  (lifetable-class), 41
getOmega, mdt-method (mdt-class), 44
head, lifetable-method
  (lifetable-class), 41
IAxn, 18, 34, 37, 38
Iaxn, 36
increasingAnnuity, 19, 37, 40
initialize, actuarialtable-method
  (actuarialtable-class), 6
initialize, lifetable-method
  (lifetable-class), 41
initialize, mdt-method (mdt-class), 44
intensity2Interest, 38, 40
interest2Discount, 39
interest2Intensity
  (intensity2Interest), 38
Isn, 40
lifectingencies
  (lifecontingencies-package), 3
lifecontingencies-package, 3
lifetable, 6, 7, 24, 30, 44, 50, 52, 59
lifetable-class, 41
Lxt, 43
mdt-class, 44
multiple life probabilities, 45
mx2qx, 46
mxt, 47
nominal2Real, 39, 40, 48
plot, lifetable, ANY-method
  (lifetable-class), 41
presentValue, 49
print, actuarialtable-method
  (actuarialtable-class), 6
print, lifetable-method
  (lifetable-class), 41
print, mdt-method (mdt-class), 44
probs2lifetable, 50
pxt, 45, 46, 51
pxyt, 15, 45, 53
pxyzt (multiple life probabilities), 45
qx2mx, 54
qxt (pxt), 51
qxt.fromQxprime (qxt.prime.fromMdtt), 55
qxt.prime.fromMdtt, 55
qxyt (pxyt), 53
qxyzt (multiple life probabilities), 45
real2Nominal, 39, 49
real2Nominal (nominal2Real), 48
rlife (rlifes), 58
rlifeContingencies, 33, 56, 60
rlifeContingenciesXyz, 33, 60
rlifeContingenciesXyz
  (rlifeContingencies), 56
rlifes, 58
rlifexyz (rlifes), 58
rmdt, 59
show, actuarialtable-method
  (actuarialtable-class), 6
show, lifetable-method
  (lifetable-class), 41
show, mdt-method (mdt-class), 44
soa08, 60
soa08Act, 61
SoAISTdata, 62
soaLt, 63
summary, actuarialtable-method
  (actuarialtable-class), 6
summary, lifetable-method
  (lifetable-class), 41
summary, mdt-method (mdt-class), 44
tail, lifetable-method
  (lifetable-class), 41
Tx (Lxt), 43
Uk life tables, 64