Package 'MortalityTables'

April 3, 2018

Type Package

Version 1.0

Date 2018-03-30

Title A Framework for Various Types of Mortality / Life Tables

Author Reinhold Kainhofer [aut, cre]

Maintainer Reinhold Kainhofer <reinhold@kainhofer.com>

URL https://gitlab.open-tools.net/R/r-mortality-tables

Depends ggplot2, methods, scales, utils

Suggests lifecontingencies, knitr, rmarkdown

Description Classes to implement and plot cohort life tables
for actuarial calculations. In particular, birth-year dependent mortality
tables using a yearly trend to extrapolate from a base year are implemented,
as well as period life table, cohort life tables using an age shift, and
merged life tables.

License GPL (>= 2)

RoxygenNote 6.0.1

Collate 'mortalityTable.R' 'mortalityTable.period.R'
'mortalityTable.ageShift.R' 'ageShift.R'
'mortalityTable.observed.R' 'mortalityTable.joined.R'
'mortalityTable.mixed.R' 'ages.R' 'baseTable.R' 'baseYear.R'
'fillAges.R' 'pensionTable.R' 'commutationNumbers.R'
'mortalityTable.improvementFactors.R'
'mortalityTable.trendProjection.R' 'deathProbabilities.R'
'getCohortTable.R' 'getOmega.R' 'getPeriodTable.R'
'lifeTable.R' 'makeQxDataFrame.R' 'mortalityComparisonTable.R'
'mortalityImprovement.R' 'mortalityTable.MakehamGompertz.R'
'mortalityTable.Weibull.R' 'mortalityTable.deMoivre.R'
'periodDeathProbabilities.R' 'mortalityTable.jointLives.R'
'mortalityTables.list.R' 'mortalityTables.load.R'
'plot.mortalityTable.R' 'plotMortalityTableComparisons.R'
'plotMortalityTables.R' 'plotMortalityTrend.R' 'setLoading.R'
'setModification.R' 'undampenTrend.R'
'whittaker.mortalityTable.R'

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2018-04-03 20:07:24 UTC

R topics documented:

`MortalityTables-package` *Provide life table classes for life insurance purposes*

Description

Provide life table classes for life insurance purposes

Author(s)

Maintainer: Reinhold Kainhofer <reinhold@kainhofer.com>

See Also

Useful links:

- https://gitlab.open-tools.net/R/r-mortality-tables

`ages` *Return the defined ages of the life table*

Description

Return the defined ages of the life table

Usage

```
ages(object, ...)

## S4 method for signature 'mortalityTable.period'
ages(object, ...)

## S4 method for signature 'mortalityTable.mixed'
ages(object, ...)

## S4 method for signature 'mortalityTable.jointLives'
ages(object, ...)
```

Arguments

object	A life table object (instance of a mortalityTable class)
...	Currently unused

Methods (by class)

- mortalityTable.period: Return the defined ages of the period life table
- mortalityTable.mixed: Return the defined ages of the mixed life table
- mortalityTable.jointLives: Return the defined ages of the joint lives mortality table (returns the ages of the first table used for joint lives)

Examples

```
mortalityTables.load("Austria_*")
ages(AVOe2005R.male)
ages(AVOe1996R.male)
ages(mort.AT.census.2011.male)
```

---

ageShift *Return the age shift of the age-shifted life table given the birth year*

Description

Return the age shift of the age-shifted life table given the birth year

Usage

```
ageShift(object, YOB = 1975, ...)

## S4 method for signature 'mortalityTable'
ageShift(object, YOB = 1975, ...)

## S4 method for signature 'mortalityTable.ageShift'
ageShift(object, YOB = 1975, ...)
```

Arguments

object	The life table object (class inherited from mortalityTable)
YOB	The birth year for which the age shift should be determined.
...	Other parameters (currently unused)

Methods (by class)

- mortalityTable: Age shifts apply only to mortalityTagle.ageShift, so all other tables return NA.
- mortalityTable.ageShift: Return the age shift of the age-shifted life table given the birth year

Examples

```
mortalityTables.load("Austria_Annuities")
ageShift(AVOe2005R.male.av, YOB=1910)
ageShift(AVOe2005R.male.av, YOB=1955)
ageShift(AVOe2005R.male.av, YOB=2010)
# A table with trend does NOT have any age shift, so NA is returned:
ageShift(AVOe2005R.male, YOB=1910)
```

---

baseTable *Return the base table of the life table*

Description

Return the base table of the life table

Usage

```
baseTable(object, ...)

## S4 method for signature 'mortalityTable'
baseTable(object, ...)

## S4 method for signature 'mortalityTable.period'
baseTable(object, ...)

## S4 method for signature 'mortalityTable.jointLives'
baseTable(object, ...)
```

Arguments

object	The life table object (class inherited from mortalityTable)
...	Other parameters (currently unused)

Methods (by class)

- mortalityTable: Return the base table of the life table
- mortalityTable.period: Return the base table of the life table
- mortalityTable.jointLives: Return the base table of the joint lives mortality table (returns the base table of the first table used for joint lives)

Examples

```
mortalityTables.load("Austria_Annuities")
baseTable(AVOe2005R.male)
```

baseYear *Return the base year of the life table*

Description

Return the base year of the life table

Usage

```
baseYear(object, ...)

## S4 method for signature 'mortalityTable'
baseYear(object, ...)

## S4 method for signature 'mortalityTable.mixed'
baseYear(object, ...)

## S4 method for signature 'mortalityTable.jointLives'
baseYear(object, ...)
```

Arguments

object	The life table object (class inherited from mortalityTable)
...	Other parameters (currently unused)

Methods (by class)

- mortalityTable: Return the base year of the life table
- mortalityTable.mixed: Return the base year of the life table
- mortalityTable.jointLives: Return the base year of the life table

Examples

```
mortalityTables.load("Austria_Annuities")
baseYear(AVOe2005R.male)
```

`calculateImprovements` *Calculate the improvement factors for the given birth-year and the* `mortalityTable.improvementFactors` *object*

Description

Calculate the improvement factors for the given birth-year and the `mortalityTable.improvementFactors` object

Usage

```
calculateImprovements(object, ...)

## S4 method for signature 'mortalityTable.improvementFactors'
calculateImprovements(object, ...,
  Period = NULL, YOB = 1982)
```

Arguments

`object`	A pension table object (instance of a `mortalityTable.improvementFactors` class)
`...`	Currently unused
`Period`	Observation period (either `Period` or `YOB` should be given)
`YOB`	Year of birth (either `Period` or `YOB` should be given)

Methods (by class)

- `mortalityTable.improvementFactors`: Calculate the total mortality improvement factors relative to the base year for the given birth-year and the `mortalityTable.improvementFactors` object

Examples

```
pensionTables.load("USA_PensionPlan_RP2014")
calculateImprovements(RP2014.male@qx, YOB = 2017)
```

`commutationNumbers` *Calculate the commutation numbers for the given parameters, using the mortality table and an interest rate*

Description

Calculate the commutation numbers for the given parameters, using the mortality table and an interest rate

Usage

```
commutationNumbers(object, ..., ages = NULL, i = 0.03)

## S4 method for signature 'mortalityTable'
commutationNumbers(object, ..., ages = NULL,
  i = 0.03)

## S4 method for signature 'numeric'
commutationNumbers(object, ages, i = 0.03)

## S4 method for signature 'pensionTable'
commutationNumbers(object, ..., ages = NULL,
  i = 0.03)
```

Arguments

`object`	The life table object (class inherited from mortalityTable)
`...`	Other parameters to be passed to the deathProbabilities call (e.g. YOB)
`ages`	Vector of ages for which the probabilities should be extracted and commutation numbers calculates
`i`	Interest rate used for the calculation of the commutation numbers

Methods (by class)

- `mortalityTable`: Calculate the commutation numbers for the given parameters, using the mortality table and an interest rate
- `numeric`: Calculate the commutation numbers for the given death probabilities (passed as a numeric vector with argument name "object"), ages and an interest rate Return value is a list of data frames
- `pensionTable`: Calculate the commutation numbers for the given parameters, using the pension table and an interest rate Return value is a list of data frames

Examples

```
mortalityTables.load("Austria_Annuities")
commutationNumbers(AVOe2005R.male, i = 0.03, YOB = 1975)
```

deathProbabilities *Return the (cohort) death probabilities of the life table given the birth year (if needed)*

Description

Return the (cohort) death probabilities of the life table given the birth year (if needed)

Usage

```
deathProbabilities(object, ..., ages = NULL, YOB = 1975)

## S4 method for signature 'mortalityTable.period'
deathProbabilities(object, ..., ages = NULL,
  YOB = 1975)

## S4 method for signature 'mortalityTable.ageShift'
deathProbabilities(object, ...,
  ages = NULL, YOB = 1975)

## S4 method for signature 'mortalityTable.trendProjection'
deathProbabilities(object, ...,
  ages = NULL, YOB = 1975)

## S4 method for signature 'mortalityTable.improvementFactors'
deathProbabilities(object, ...,
  ages = NULL, YOB = 1975)

## S4 method for signature 'mortalityTable.mixed'
deathProbabilities(object, ..., ages = NULL,
  YOB = 1975)

## S4 method for signature 'mortalityTable.jointLives'
deathProbabilities(object, ...,
  ageDifferences = c(), ages = NULL, YOB = 1975)
```

Arguments

object	The life table object (class inherited from mortalityTable)
...	Other parameters (currently unused)
ages	Desired age range (if NULL, the probabilities of the age range provided by the table will be returned), missing ages will be filled with NA
YOB	The birth year for which the death probabilities should be calculated
ageDifferences	A vector of age differences of all joint lives.

Methods (by class)

- `mortalityTable.period`: Return the (cohort) death probabilities of the life table given the birth year (if needed)
- `mortalityTable.ageShift`: Return the (cohort) death probabilities of the life table given the birth year (if needed)
- `mortalityTable.trendProjection`: Return the (cohort) death probabilities of the life table given the birth year (if needed)
- `mortalityTable.improvementFactors`: Return the (cohort) death probabilities of the life table given the birth year (if needed)
- `mortalityTable.mixed`: Return the (cohort) death probabilities of the life table given the birth year (if needed)
- `mortalityTable.jointLives`: Return the (cohort) death probabilities of the life table given the birth year (if needed)

Examples

```
mortalityTables.load("Austria_Annuities")
deathProbabilities(AVOe2005R.male, YOB = 1975)
deathProbabilities(AVOe2005R.male, YOB = 2017)

mortalityTables.load("Germany_Census")
table.JL = mortalityTable.jointLives(
    name = "ADSt 24/26 auf verbundene Leben",
    table = mort.DE.census.1924.26.male
)
deathProbabilities(table.JL, YOB = 1977, ageDifferences = c(1, 5, -5, 16))
deathProbabilities(table.JL, YOB = 1977, ageDifferences = c(0))
deathProbabilities(table.JL, YOB = 1977, ageDifferences = c(1, 5, 16))
```

`deathProbabilitiesIndividual` *Return a matrix of the persons' individual death probabilities of a joint-life table (instance of `mortalityTable.jointLives`)*

Description

Return a matrix of the persons' individual death probabilities of a joint-life table (instance of `mortalityTable.jointLives`)

Usage

```
deathProbabilitiesIndividual(tables, YOB, ageDifferences)
```

Arguments

tables	List of life table objects (object inherited from mortalityTable)
YOB	The birth year for the first person
ageDifferences	The age differences to the first person

Examples

```
mortalityTables.load("Germany_Census")
deathProbabilitiesIndividual(list(mort.DE.census.1924.26.male), 1977, c(0, 0))
deathProbabilitiesIndividual(list(mort.DE.census.1924.26.male), 1977, c(0, -5, 13))
```

---

fillAges *Fill the given probabilities with NA to match the desired age range.*

Description

Fill the given probabilities with NA to match the desired age range.

Usage

```
fillAges(probs = c(), givenAges = c(), neededAges = NULL,
  fill = NA_real_)
```

Arguments

probs	Numeric vector
givenAges	ages assigned to the given vector
neededAges	desired age range for output
fill	If set, missing values will be replaced with this value. Default is to fill with NA.

Examples

```
# Ages 20-70 have linearly increasing death probabilities. Fill with 0 for the whole age range 0-120
fillAges(probs = c(0:50/50), givenAges = 20:70, neededAges = 0:120, fill = 0)
```

generateAgeShift *Generate data.frame containing age shifts for each birth year*

Description

Generate a dataframe suitable to be passed to the mortalityTable.ageShift class.

Usage

```
generateAgeShift(initial = 0, YOBs = c(1900, 2100), step = -1)
```

Arguments

initial	Age shift for the first birth year given in the YOBs vector
YOBs	Vector of birth years in which the age shift changes by step. The last entry gives the first birth year that does not have any shift defined any more.
step	How much the age shift changes in each year given in the YOBs vector

Examples

```
generateAgeShift(initial = 1, YOBs = c(1922, 1944, 1958, 1973, 1989, 2006, 2023, 2041, 2056))
```

getCohortTable *Return the cohort life table as a* mortalityTable.period *object*

Description

Return the cohort life table as a mortalityTable.period object

Usage

```
getCohortTable(object, YOB, ...)

## S4 method for signature 'mortalityTable'
getCohortTable(object, YOB, ...)
```

Arguments

object	The life table object (class inherited from mortalityTable)
YOB	The birth year for which the life table should be calculated
...	Other parameters (currently unused)

Methods (by class)

- `mortalityTable`: Return the cohort life table as a `mortalityTable.period` object

Examples

```
mortalityTables.load("Austria_Annuities")
tb75 = getCohortTable(AVOe2005R.male, YOB = 1975)
# The tb75 is a fixed table with no trend any more
plot(AVOe2005R.male, tb75, Period = 2017)
```

---

getOmega *Return the maximum age of the life table*

Description

Return the maximum age of the life table

Usage

```
getOmega(object)

## S4 method for signature 'mortalityTable.period'
getOmega(object)

## S4 method for signature 'mortalityTable.mixed'
getOmega(object)

## S4 method for signature 'mortalityTable.jointLives'
getOmega(object)
```

Arguments

`object`	A life table object (instance of a `mortalityTable` class)

Methods (by class)

- `mortalityTable.period`: Return the maximum age of the period life table
- `mortalityTable.mixed`: Return the maximum age of the mixed life table
- `mortalityTable.jointLives`: Return the maximum age of the joint lives mortality table (returns the maximum age of the first table used for joint lives, as the ages of the joint lives are now known to the function)

Examples

```
mortalityTables.load("Austria_Annuities")
getOmega(AVOe2005R.male)
getOmega(mortalityTable.deMoivre(omega = 100))
```

getPeriodTable *Return the period life table as a* mortalityTable.period *object*

Description

Return the period life table as a mortalityTable.period object

Usage

```
getPeriodTable(object, Period, ...)

## S4 method for signature 'mortalityTable'
getPeriodTable(object, Period, ...)
```

Arguments

object	The life table object (class inherited from mortalityTable)
Period	The observation year, for which the death probabilities should be determined
...	Other parameters (currently unused)

Methods (by class)

- mortalityTable: Return the period life table as a mortalityTable.period object

Examples

```
mortalityTables.load("Austria_Annuities")
tb17 = getPeriodTable(AVOe2005R.male, Period = 2017)
# The tb17 is a fixed table with no trend any more
plot(AVOe2005R.male, tb17, YOB = 1975)
```

lifeTable *Return the lifetable object (package lifecontingencies) for the cohort life table*

Description

Return the lifetable object (package lifecontingencies) for the cohort life table

Usage

```
lifeTable(object, ...)

## S4 method for signature 'mortalityTable'
lifeTable(object, ...)
```

Arguments

object	The life table object (class inherited from mortalityTable)
...	Parameters to be passed to the deathProbabilities method of the life table

Methods (by class)

- mortalityTable: Return the lifetable object (package lifecontingencies) for the cohort life table

Examples

```
library("lifecontingencies")
mortalityTables.load("Austria_Annuities")
lifeTable(AVOe2005R.male, YOB = 2017)
axn(lifeTable(AVOe2005R.male, YOB = 1975), x = 65, i = 0.03)
axn(lifeTable(AVOe2005R.male, YOB = 2017), x = 65, i = 0.03)
```

---

makeQxDataFrame *Converts one or multiple mortality table objects to a data frame that can be plotted by* plotMortalityTables *or* plotMortalityTableComparisons

---

Description

It is not required to call this function manually, plotMortalityTables will automatically do it if object derived from class mortalityTable are passed.

Usage

```
makeQxDataFrame(..., YOB = 1972, Period = NA, reference = NULL)
```

Arguments

...	Life tables (objects of classes derived from mortalityTable)
YOB	desired year of birth to be plotted as cohort life table (default: 1972)
Period	desired observation year to be plotted (default: NA). If both YOB and Period are given, a period comparison is generated.
reference	Reference life table, used to show relative death probabilities (i.e. the q_x for all ages are divided by the corresponding probabilities of the reference table)

Examples

```
mortalityTables.load("Austria_Annuities")
makeQxDataFrame(AVOe2005R.male, AVOe2005R.female, YOB = 1975)
```

`mortalityComparisonTable` *Calculate relative mortalities for age bands and birth years*

Description

Calculate relative mortalities for age bands and birth years

Usage

```
mortalityComparisonTable(table1, table2, years, ages, binsize = 5, ...)
```

Arguments

`table1, table2`	The `mortalityTable` objects to compare (mortalities of `table1` relative to `table2`)
`years`	Vector of birth years to include in the comparisons.
`ages`	Vector of ages to include in the comparisons
`binsize`	How many ages to combine into one age band
`...`	Other parameters (currently unused)

Examples

```
mortalityTables.load("Austria_Annuities")
# Compare mortality of Austrian male and female annuitants born 1930 to 2030
mortalityComparisonTable(
    AVOe2005R.male, AVOe2005R.female,
    years = seq(1930, 2030, by = 10),
    ages = 0:119)

# Compare the two Austrian male annuity tables AVOe 2005-R and AVOe 1996-R,
# combining ages 10-19, 20-29, etc.
mortalityComparisonTable(
    AVOe2005R.male, AVOe1996R.male,
    years = seq(1930, 2030, by = 10),
    ages = 0:109, binsize=10)
```

`mortalityImprovement` *Return the mortality trend (yearly log-death-probability improvement) of the given period or the given generation.*

Description

Return the mortality trend (yearly log-death-probability improvement) of the given period or the given generation.

Usage

```
mortalityImprovement(object, ..., Period = NULL, YOB = 1975)

## S4 method for signature 'mortalityTable'
mortalityImprovement(object, ..., Period = NULL,
  YOB = 1975)
```

Arguments

`object`	The life table object (class inherited from mortalityTable)
`...`	Other parameters (currently unused)
`Period`	The observation year for which the mortality improvement should be calculated. If both YOB and Period are given, YOB is ignored.
`YOB`	The birth year for which the mortality improvement should be calculated

Methods (by class)

- `mortalityTable`: Return the yearly log-mortality improvement of the life table given the birth or observation year

Examples

```
mortalityTables.load("Austria_Annuities")
# AVOe 2005R includes a trend decline by default, compare the exact table
# with the table without decline:
mortalityImprovement(AVOe2005R.male, Period = 2017)
mortalityImprovement(AVOe2005R.male.nodamping, Period = 2017)
```

`mortalityTable-class` *Class mortalityTable*

Description

Class `mortalityTable` is the (virtual) base class for all mortality tables. It contains the name and some general values applying to all types of tables, but does not contain any data itself. Use a child class to create actual mortality tables.

Slots

`name` The human-readable name of the mortality table

`baseYear` The base year of the mortality table (e.g. for tables with trend projection)

`modification` A function that will be called with the final death probabilities to give the user a way to modify the final probabilities

`loading` Additional security loading on the resulting table (single numeric value, e.g. 0.05 adds 5% security margin to the probabilities)

`mortalityTable.ageShift-class`
Class mortalityTable.ageShift - Cohort life tables generated by age-shift

Description

A cohort life table, obtained by age-shifting from a given base table (death probabilities

Slots

`ageShifts` A `data.frame` with columns `YOB` and `shifts` giving the age shifts for each birth year

Examples

```
mortalityTables.load("Austria_Annuities_AVOe2005R")
tb = mortalityTable.ageShift(
    ages = ages(AVOe2005R.male),
    deathProbs = deathProbabilities(AVOe2005R.male, YOB = 1992),
    ageShifts = generateAgeShift(1, c(1962, 1985, 2000, 2015, 2040, 2070)))
# The cohort tables for different birth years are just the base probabilities with modified ages
plot(getCohortTable(tb, YOB = 1963), getCohortTable(tb, YOB = 2017))
```

mortalityTable.deMoivre-class

Class mortalityTable.deMoivre - Mortality table with de Moivre's law

Description

A period life table with maximum age omega dn the time of death identically distributed on the interval [0, omega]. The only required slot is the maximum age `omega`, all probabilities are calculated from it. Optionally, a name and loading can be passed (inherited from `mortalityTable`).

Slots

`omega` Maximum age

Examples

```
mm = mortalityTable.deMoivre(100)
plot(mm,
     mortalityTable.deMoivre(90),
     mortalityTable.deMoivre(50))
```

mortalityTable.improvementFactors-class

Class mortalityTable.improvementFactors - Cohort life table with improvement factors

Description

A cohort life table, obtained by an improvment factor projection from a given base table (PODs for a given observation year).

Slots

`baseYear` The base year for the improvements (`baseTable` describes the death probabilities in this year)

`improvement` Yearly improvement factors per age

Examples

```
mortalityTables.load("Austria_Annuities_AVOe2005R")
# AVOe 2005R base table with yearly improvements of 3% for age 0, linearly
# decreasing to 0% for age 120.
tb = mortalityTable.improvementFactors(
    ages = ages(AVOe2005R.male),
    deathProbs = periodDeathProbabilities(AVOe2005R.male, Period = 2002),
```

```
    baseYear = 2002,
    improvement = 0.03 * (1 - ages(AVOe2005R.male)/121),
    name = "AVOe 2005R base with linearly falling improvements (DEMO)"
)
# Yearly trend is declining:
plotMortalityTrend(tb, AVOe2005R.male, Period = 2017, title = "Mortality Trend")
# The cohort tables for different birth years:
plot(getCohortTable(tb, YOB = 1963), getCohortTable(tb, YOB = 2017))
```

`mortalityTable.jointLives-class`

Class mortalityTable.jointLives - Life table for multiple joint lives

---

Description

A cohort life table obtained by calculating joint death probabilities for multiple lives, each possibly using a different mortality table.

Slots

`table` The `mortalityTable` object for all lives (vector if different tables should be used for the different persons)

Examples

```
mortalityTables.load("Germany_Census")
table.JL = mortalityTable.jointLives(
    name = "ADSt 24/26 auf verbundene Leben",
    table = mort.DE.census.1924.26.male
)
deathProbabilities(table.JL, YOB = 1977, ageDifferences = c(1, 5, -5, 16))
deathProbabilities(table.JL, YOB = 1977, ageDifferences = c(0))
deathProbabilities(table.JL, YOB = 1977, ageDifferences = c(1, 5, 16))
```

`mortalityTable.MakehamGompertz-class`

Class mortalityTable.MakehamGompertz - Mortality table with Makeham-Gompertz's law

---

Description

A period life table following Makeham and Gompertz's law of a mortality rate μ increasing exponentially with age x ($\mu_{x+t} = A + Bc^{(x+t)}$). The only required slots are the parameters A, B and c, all probabilities are calculated from them, for technical reasons a maximum age of 120 is technically assumed. Optionally, a name and loading can be passed (inherited from `mortalityTable`).

Slots

A Parameter A of the Makeham-Gompertz distribution

B Parameter B of the Makeham-Gompertz distribution

c Parameter c of the Makeham-Gompertz distribution

omega Maximum age (default: 150)

Examples

```
# A Gompertz mortality, roughtly approximating the Austrian annuitants 2017
gmp = mortalityTable.MakehamGompertz(A = 0, B = 0.00001, c = 1.11)
mortalityTables.load("Austria_Annuities_AVOe2005R")
plot(gmp, AVOe2005R.male, Period=2017)

# A Makeham-Gompertz mortality, approximating the Austrian annuitants 2017
mg = mortalityTable.MakehamGompertz(A = 0.0002, B = 0.00001, c = 1.11)
plot(mg, gmp, AVOe2005R.male, Period=2017)
```

---

mortalityTable.mixed-class

Class mortalityTable.mixed - Life table as a mix of two life tables

---

Description

A cohort life table obtained by mixing two life tables with the given weights. Typically, when only gender-specific tables are available, unisex tables are generated by mixing the two gender-specific tables for males and for females with a pre-defined, constant proportion (e.g. 60:30 or 40:60, depending on the portfolio and on the security margins).

Slots

table1 The first mortalityTable

table2 The second mortalityTable

weight1 The weight of the first mortality table

weight2 The weight of the second mortality table

loading Additional security loading

Examples

```
mortalityTables.load("Austria_Annuities_AVOe2005R")
# Generate a unisex table with mixing relation 60:40 from male + female tables
AVOe2005R.myUnisex = mortalityTable.mixed(
    table1 = AVOe2005R.male, table2 = AVOe2005R.female,
    weight1 = 0.6, weight2 = 0.4,
    name = "My custom AVOe 2005R unisex (60:40)")
plot(AVOe2005R.myUnisex, AVOe2005R.male, AVOe2005R.female, Period = 2017)
```

mortalityTable.once *Generate a (deterministic) mortality table with only one probability set to 1 (for the given age)*

Description

Generate a (deterministic) mortality table with only one probability set to 1 (for the given age)

Usage

```
mortalityTable.once(transitionAge, name = "Deterministic mortality table",
  ages = 0:99)
```

Arguments

transitionAge	The age where the deterministic transition occurs
name	The name of the table
ages	The ages of the table

mortalityTable.onceAndFuture *Generate a (deterministic) mortality table with all probabilities starting at a given age set to 1*

Description

Generate a (deterministic) mortality table with all probabilities starting at a given age set to 1

Usage

```
mortalityTable.onceAndFuture(transitionAge,
  name = "Deterministic mortality table", ages = 0:99)
```

Arguments

transitionAge	The age where the deterministic transition occurs
name	The name of the table
ages	The ages of the table

mortalityTable.period-class

Class mortalityTable.period - Period life tables

Description

A period life table, giving death probabilities for each age, up to maximum age `omega`. The `baseYear` slot can be used to hold information about the period.

Slots

`ages` The ages corresponding to the entries of the deathProbs

`deathProbs` The one-year death probabilities for the ages

`exposures` (Optional) exposured used to determine death probabilities (can be used as weights for smoothing, for variances, etc.)

Examples

```
linTable = mortalityTable.period(name="linear mortality", ages = 0:50, deathProbs = 0:50/50)
constTable = mortalityTable.period(name="const. mortality", ages = 0:50,
                                   deathProbs = c(rep(0.1, 50), 1))
plot(linTable, constTable, title="Comparison of linear and constand death probabilities")

# A sample observation table with exposures and raw probabilities
obsTable = mortalityTable.period(
    name = "trivial observed table",
    ages = 0:15,
    deathProbs = c(
        0.0072, 0.00212, 0.00081, 0.0005, 0.0013,
        0.001, 0.00122, 0.00142, 0.007, 0.0043,
        0.0058, 0.0067, 0.0082, 0.0091, 0.0075, 0.01),
    exposures = c(
        150, 222, 350, 362, 542,
        682, 1022, 1053, 1103, 1037,
        968, 736, 822, 701, 653, 438))
plot(obsTable, title = "Observed death probabilities")
```

mortalityTable.trendProjection-class

Class mortalityTable.trendProjection - Cohort mortality table with age-specific trend

Description

A cohort mortality table, obtained by a trend projection from a given base table (PODs for a given observation year). Typically, the trend is obtained by the Lee-Carter method or some other trend estimation. The dampingFunction can be used to modify the cumulative years (e.g. G(tau+x) instead of tau+x) If trend2 is given, the G(tau+x) gives the weight of the first trend, 1-G(tau+x) the weight of the second trend

Slots

`baseYear` The base year of the trend projection (`baseTable` describes the death probabilities in this year)

`trend` The yearly improvements of the log-death probabilities (per age)

`dampingFunction` A possible damping of the trend. This is a function `damping(delta_years)` that gets a vector of years from the baseYear and should return the dampened values.

`trend2` The alternate trend. If given, the damping function interpolates between `trend` and `trend2`, otherwise the dumping function simply modifies the coefficients of `trend`.

Examples

```
obsTable = mortalityTable.trendProjection(
    name = "Const. table with trend",
    baseYear = 2018,
    ages = 0:15,
    deathProbs = rep(0.02, 16),
    trend = c(
         0.045, 0.04, 0.03, 0.04, 0.042, 0.041, 0.038, 0.035,
         0.032, 0.031, 0.028, 0.020, 0.015, 0.01, 0.005, 0))
# In 2018 the flat mortality can be seen
plotMortalityTables(obsTable, Period = 2018, title = "Period death probabilities 2018")
# In 2038, the age-specific trend affected the probabilities differently for 20 years:
plotMortalityTables(obsTable, Period = 2038, title = "Period death probabilities 2038")
# Consequently, a person born 2018 will also not have constand death probabilities
plotMortalityTables(obsTable, YOB = 2018, title = "Cohort death probabilities, YOB 2018")

plotMortalityTables(
    lapply(2018:2033, function(y) getCohortTable(obsTable, YOB = y)),
    title = "Cohort tables for different YOBs", legend.position = c(0.99, 0.01))
plotMortalityTables(
   lapply(2018:2033, function(y) getPeriodTable(obsTable, Period = y)),
   title = "Period tables for different years", legend.position = c(0.99, 0.01))
```

`mortalityTable.Weibull-class`

Class mortalityTable.Weibull - Mortality table with Weibull's law

---

Description

A period life table following Weibulls's law of a mortality rate μ increasing as a power of t:

$$\mu_{x+t} = k * (x + t)^n$$

The only required slots are the parameters $k > 0$ and $n > 0$, all probabilities are calculated from them, for technical reasons a maximum age of 150 is technically assumed. Optionally, a name and loading can be passed (inherited from `mortalityTable`).

Slots

`k` Parameter k of the Weibull distribution

`n` Parameter n of the Weibull distribution

`omega` Maximum age (default: 120)

Examples

```
# A Weibull mortality
wbl = mortalityTable.Weibull(k = 0.0000000001, n = 4.8)
mortalityTables.load("Austria_Annuities_AVOe2005R")
plot(wbl, AVOe2005R.male, Period=2017, ylim = c(0.00005, 1))
```

`mortalityTable.zeroes` *Generate a mortality table with all probabilities set to zero.*

Description

Generate a mortality table with all probabilities set to zero.

Usage

```
mortalityTable.zeroes(name = "Zero mortality table", ages = 0:99)
```

Arguments

`name`	The name of the table
`ages`	The ages of the table

`mortalityTables.list` *List all available sets of life tables provided by the MortalityTables-package package An existing life table can then be loaded with mortalityTables.load.*

Description

List all available sets of life tables provided by the MortalityTables-package package An existing life table can then be loaded with mortalityTables.load.

Usage

```
mortalityTables.list(pattern = "*", package = c("MortalityTables",
  "MortalityTablesPrivate"), prefix = "MortalityTables")
```

Arguments

`pattern`	Restrict the results only to life table sets that match the pattern (default: "*" to show all sets)
`package`	The package that contains the desired dataset in its `extdata/` directory. Defaults to the "MortalityTables" package. Multiple packages can be given as a vector.
`prefix`	The file prefix, defaults to MortalityTables. Can be overridden to list other types of files, like "PensionTables"

Examples

```
mortalityTables.list()
mortalityTables.list("Austria_*")
mortalityTables.list("*Annuities")
mortalityTables.list(package = c("MyCustomPackage"))
```

`mortalityTables.load` *Load a named set of mortality tables provided by the MortalityTables package*

Description

Load a named set of mortality tables provided by the MortalityTables package

Usage

```
mortalityTables.load(dataset, package = c("MortalityTables",
  "MortalityTablesPrivate"), prefix = "MortalityTables")
```

Arguments

dataset	The set(s) of life tables to be loaded. A list of all available data sets is provided by the function mortalityTables.list. Wildcards (*) are allowed to match and load multiple datasets.
package	The package that contains the dataset in its extdata/ directory. Defaults to the "MortalityTables" package. Multiple packages can be given as a vector.
prefix	The prefix for the data sets (default is "MortalityTables").

Examples

```
mortalityTables.list()
mortalityTables.load("Austria_Annuities_*")
mortalityTables.load("Austria_Annuities_AVOe2005R")
mortalityTables.load("*Annuities")
mortalityTables.load("MyCustomTable", package = c("MyCustomPackage"))
```

---

pensionTable-class *Class pensionTable*

Description

Class pensionTable is the (virtual) base class for all pensions tables. It contains the name and some general values applying to all types of tables. In particular, it holds individual tables for each of the transition probabilities. Possible states are:

- active: healty, no pension, typically paying some kin of premium
- incapacity: disablity pension, in most cases permanent, not working, early pension
- retirement: old age pension, usually starting with a fixed age
- dead
  - Widow/widower pension

Correspondingly, the following transition probabilities can be given:

qxaa death probability of actives (active -> dead)

ix invalidity probability (active -> incapacity)

qix death probability of invalid (invalid -> dead)

rx reactivation probability (incapacity -> active)

apx retirement probability (active -> retirement), typically 1 for a fixed age

qpx death probability of retired (retired -> dead)

hx probability of a widow at moment of death (dead -> widow), y(x) age difference

qxw death probability of widows/widowers

qgx death probability of total group (irrespective of state)

invalids.retire Flag to indicate whether invalid persons retire like active (one death probability for all retirees) or whether they stay invalid until death with death probabilities specific to invalids.

Slots

`qx` Death probability table of actives (derived from mortalityTable)

`ix` Invalidity probability of actives (derived from mortalityTable)

`qix` Death probability table of invalids (derived from mortalityTable)

`rx` Reactivation probability of invalids (derived from mortalityTable)

`apx` Retirement probability of actives (derived from mortalityTable)

`qpx` Death probability of old age pensioners (derived from mortalityTable)

`hx` Probability of a widow at the moment of death (derived from mortalityTable)

`qwy` Death probability of widow(er)s (derived from mortality Table)

`yx` Age difference of the widow to the deceased

`qgx` Death probability of whole group (derived from mortalityTable), irrespective of state

`invalids.retire` Whether invalids retire like actives or stay invalid until death

---

`pensionTables.list` *List all available sets of pension tables provided by the MortalityTables-package package An existing pension table can then be loaded with pensionTables.load.*

Description

List all available sets of pension tables provided by the MortalityTables-package package An existing pension table can then be loaded with pensionTables.load.

Usage

```
pensionTables.list(pattern = "*", package = c("MortalityTables",
  "MortalityTablesPrivate"))
```

Arguments

`pattern`	Restrict the results only to pension table sets that match the pattern (default: "*" to show all sets)
`package`	The package that contains the desired dataset in its `extdata/` directory. Defaults to the "MortalityTables" package. Multiple packages can be given as a vector.

Examples

```
pensionTables.list()
pensionTables.list("USA_*")
pensionTables.list(package = c("MyCustomPackage"))
```

`pensionTables.load` *Load a named set of pension tables provided by the MortalityTables package*

Description

Load a named set of pension tables provided by the MortalityTables package

Usage

```
pensionTables.load(dataset, package = c("MortalityTables",
  "MortalityTablesPrivate"))
```

Arguments

`dataset`	The set of lifpensione tables to be loaded. A list of all available data sets is provided by the function `pensionTables.list`. Wildcards (*) are allowed to match and load multiple datasets.
`package`	The package that contains the dataset in its `extdata/` directory. Defaults to the "MortalityTables" package. Multiple packages can be given as a vector.

pensionTables.list() pensionTables.load("*") pensionTables.load("USA_PensionPlan_RP2014")

`periodDeathProbabilities` *Return the (period) death probabilities of the life table for a given observation year*

Description

Return the (period) death probabilities of the life table for a given observation year

Usage

```
periodDeathProbabilities(object, ..., ages = NULL, Period = 1975)

## S4 method for signature 'mortalityTable.period'
periodDeathProbabilities(object, ...,
  ages = NULL, Period = 1975)

## S4 method for signature 'mortalityTable.ageShift'
periodDeathProbabilities(object, ...,
  ages = NULL, Period = 1975)

## S4 method for signature 'mortalityTable.trendProjection'
periodDeathProbabilities(object, ...,
```

```
  ages = NULL, Period = 1975)

## S4 method for signature 'mortalityTable.improvementFactors'
periodDeathProbabilities(object,
  ..., ages = NULL, Period = 1975)

## S4 method for signature 'mortalityTable.mixed'
periodDeathProbabilities(object, ...,
  ages = NULL, Period = 1975)

## S4 method for signature 'mortalityTable.jointLives'
periodDeathProbabilities(object, ...,
  ageDifferences = c(), ages = NULL, Period = 1975)
```

Arguments

object	The life table object (class inherited from mortalityTable)
...	Other parameters (currently unused)
ages	Desired age range (if NULL, the probabilities of the age range provided by the table will be returned), missing ages will be filled with NA
Period	The observation year for which the period death probabilities should be determined
ageDifferences	A vector of age differences of all joint lives.

Methods (by class)

- mortalityTable.period: Return the (period) death probabilities of the life table for a given observation year
- mortalityTable.ageShift: Return the (period) death probabilities of the life table for a given observation year
- mortalityTable.trendProjection: Return the (period) death probabilities of the life table for a given observation year
- mortalityTable.improvementFactors: Return the (period) death probabilities of the life table for a given observation year
- mortalityTable.mixed: Return the (period) death probabilities of the life table for a given observation year
- mortalityTable.jointLives: Return the (period) death probabilities of the joint lives mortality table for a given observation year

Examples

```
mortalityTables.load("Austria_Annuities")
periodDeathProbabilities(AVOe2005R.male, Period = 1975)
periodDeathProbabilities(AVOe2005R.male, Period = 2017)

mortalityTables.load("Germany_Census")
table.JL = mortalityTable.jointLives(
```

```
    name = "ADSt 24/26 auf verbundene Leben",
    table = mort.DE.census.1924.26.male
)
periodDeathProbabilities(table.JL, Period = 2017, ageDifferences = c(1, 5, -5, 16))
periodDeathProbabilities(table.JL, Period = 2017, ageDifferences = c(0))
periodDeathProbabilities(table.JL, Period = 2017, ageDifferences = c(1, 5, 16))
```

`periodDeathProbabilitiesIndividual`

Return a matrix of the persons' individual period death probabilities of a joint-life table (instance of `mortalityTable.jointLives`*)*

Description

Return a matrix of the persons' individual period death probabilities of a joint-life table (instance of `mortalityTable.jointLives`)

Usage

```
periodDeathProbabilitiesIndividual(tables, period, ageDifferences)
```

Arguments

`tables`	List of life table objects (object inherited from `mortalityTable`)
`period`	The observation period
`ageDifferences`	The age differences to the first person

Examples

```
mortalityTables.load("Germany_Census")
periodDeathProbabilitiesIndividual(list(mort.DE.census.1924.26.male), 1977, c(0, 0))
periodDeathProbabilitiesIndividual(list(mort.DE.census.1924.26.male), 1977, c(0, -5, 13))
```

`periodTransitionProbabilities`

Return all period transition probabilities of the pension table

---

Description

Return all period transition probabilities of the pension table

Usage

```
periodTransitionProbabilities(object, ...)

## S4 method for signature 'pensionTable'
periodTransitionProbabilities(object, Period = 2017,
  ..., ages = NULL, OverallMortality = FALSE, retirement = NULL,
  invalids.retire = object@invalids.retire, as.data.frame = TRUE)
```

Arguments

`object`	A pension table object (instance of a `pensionTable` class)
`...`	Currently unused
`Period`	Observation year
`ages`	Desired age range (if NULL, the probabilities of the age range provided by the table will be returned), missing ages will be filled with NA
`OverallMortality`	Whether the overall mortality should be returned for actives, or the active mortality
`retirement`	Override the retirement transition probabilities of the pension table. Possible values are: • Single age (describing a deterministic retirement at the given age) • mortalityTable object: transition probabilities for retirement
`invalids.retire`	Override the `pensionTable`'s `invalids.retire` flag, which indicates whether invalids retire like actives (i.e. same death probabilities after retirement) or stay invalid until death.
`as.data.frame`	Whether the return value should be a data.frame or an array containing transition matrices

Methods (by class)

- `pensionTable`: Return all transition probabilities of the pension table for the period Period

Examples

```
pensionTables.load("USA_PensionPlans")
# transitionProbabilities internally calls periodTransitionProbabilities
# if a Period is given:
transitionProbabilities(RP2014.male, Period = 1955)
periodTransitionProbabilities(RP2014.male, Period = 1955)
periodTransitionProbabilities(RP2014.male, Period = 2025)
```

`plot.mortalityTable` *Plot multiple mortality tables (life tables) in one plot*

Description

`plot.mortalityTable` displays multiple life tables (objects of child classes of `mortalityTable`) in one plot, with a legend showing the names of the tables. If the argument `reference` not given, all mortality rates are plotted on a log-linear scale for comparison. If the argument `reference` is given and is a valid life table, then all death probabilities are scaled by the given reference table and the y-axis shows the death rates as percentage of the reference table.

Usage

```
## S3 method for class 'mortalityTable'
plot(x, ..., reference = NULL, trend = FALSE)
```

Arguments

`x`	First life table to be plotted. Must be a `mortalityTable` object for the dispatcher to call this function
`...`	Additional life tables to be plotted (`mortalityTable` objects) as well as any of the following parameters (which are passed on to `plotMortalityTables` or `plotMortalityTableComparisons`): `xlim,ylim` Axes limitatation (as a two-element vectors) `xlab,ylab` Axes labels (default for x-axis: "Alter", default for y-axis: "Sterbewahrscheinlichkeit q_x") `title` The plot title `legend.position` The position of the legend (default is `c(0.9,0.1)`) `legend.key.width` The keywith of the lines in the legend (default is `unit(25,"mm")`)
`reference`	The reference table that determines the 100% values. If not given, the absolute mortality values are compared and plotted on a log-linear scale.
`trend`	If set to `TRUE`, the function `plotMortalityTrend` is used to plot the trends of the given tables.

See Also

`plotMortalityTables` and `plotMortalityTableComparisons`

Examples

```
# Load the Austrian census data
mortalityTables.load("Austria_Census")

# Plot some select census tables in a log-linear plot
plot(mort.AT.census.1869.male, mort.AT.census.1869.female,
     mort.AT.census.1971.male, mort.AT.census.1971.female,
```

```
    mort.AT.census.2011.male, mort.AT.census.2011.female,
    title="Austrian census tables",
    ylab=expression(q[x]), xlab="Age",
    xlim=c(0,90),
    legend.position=c(0.95,0.05))

# Compare some census tables with the mortality of 2011 Austrian males
plot(mort.AT.census.1869.male, mort.AT.census.1869.female,
    mort.AT.census.1971.male, mort.AT.census.1971.female,
    mort.AT.census.2011.male, mort.AT.census.2011.female,
    title="Austrian Census tables, relative to 2011 males",
    reference=mort.AT.census.2011.male)
```

`plotMortalityTableComparisons` *Plot multiple mortality tables (life tables) in one plot, relative to a given reference table*

Description

`plotMortalityTableComparisons` prints multiple life tables (objects of child classes of `mortalityTable`) in one plot and scales each by the given reference table, so that the relative mortality can be easily seen. A legend is added showing the names of the tables.

Usage

```
plotMortalityTableComparisons(data, ..., ages = NULL, xlim = NULL,
  ylim = NULL, xlab = NULL, ylab = NULL, title = "",
  legend.position = c(0.9, 0.1), legend.justification = c(1, 0),
  legend.key.width = unit(25, "mm"), reference = NULL)
```

Arguments

`data`	First life table to be plotted. Either a `data.frame` generated by `makeQxDataFrame` or a `mortalityTable` object
`...`	Additional life tables to be plotted (if `data` is a `mortalityTable` object)
`ages`	Plot only the given ages
`xlim`	X-axis limitatation (as a two-element vector)
`ylim`	Y-axis limitatation (as a two-element vector)
`xlab`	X-axis label (default: "Alter")
`ylab`	Y-axis label (default: "Sterbewahrscheinlichkeit q_x relativ zu ....")
`title`	The plot title
`legend.position`	The position of the legend (default is `c(0.9,0.1)`)

legend.justification
: The justification of the legend (default is c(1,))

legend.key.width
: The keywith of the lines in the legend (default is unit(25,"mm"))

reference
: The reference table that determines the 100% values. If not given, the first argument of data is used as reference table.

Examples

```
# Load the Austrian census data
mortalityTables.load("Austria_Census")

# Compare some census tables with the mortality of 2011 Austrian males
# plot with the reference argument is the same as calling plotMortalityTableComparisons
plot(mort.AT.census.1869.male, mort.AT.census.1869.female,
     mort.AT.census.1971.male, mort.AT.census.1971.female,
     mort.AT.census.2011.male, mort.AT.census.2011.female,
     title = "Austrian Census tables, relative to 1971 males",
     reference = mort.AT.census.1971.male)
plotMortalityTableComparisons(mort.AT.census.1869.male, mort.AT.census.1869.female,
     mort.AT.census.1971.male, mort.AT.census.1971.female,
     mort.AT.census.2011.male, mort.AT.census.2011.female,
     title = "Austrian Census tables, relative to 1971 males",
     reference = mort.AT.census.1971.male)
```

---

plotMortalityTables *Plot multiple mortality tables (life tables) in one plot*

Description

plotMortalityTables prints multiple life tables (objects of child classes of mortalityTable) in one log-linear plot, with a legend showing the names of the tables.

Usage

```
plotMortalityTables(data, ..., ages = NULL, legend.title = "Sterbetafel",
  xlim = NULL, ylim = NULL, xlab = NULL, ylab = NULL, title = "",
  legend.position = c(0.9, 0.1), legend.justification = c(1, 0),
  legend.key.width = unit(25, "mm"))
```

Arguments

data
: First life table to be plotted. Either a data.frame generated by makeQxDataFrame or a mortalityTable object

...
: Additional life tables to be plotted (if data is a mortalityTable object)

ages
: Plot only the given ages

legend.title	Title of the legend
xlim	X-axis limitatation (as a two-element vector)
ylim	Y-axis limitatation (as a two-element vector)
xlab	X-axis label (default: "Alter")
ylab	Y-axis label (default: "Sterbewahrscheinlichkeit q_x relativ zu ....")
title	The plot title
legend.position	The position of the legend (default is c(0.9,0.1))
legend.justification	The justification of the legend (default is c(1,))
legend.key.width	The keywith of the lines in the legend (default is unit(25,"mm"))

Examples

```
# Load the Austrian census data
mortalityTables.load("Austria_Annuities")
mortalityTables.load("Austria_Census")

# Plot some select census tables in a log-linear plot (plot called
# with mortalityTable objects is equla to calling plotMortalitytTables directly)
plot(mort.AT.census.1869.male, mort.AT.census.1869.female,
     mort.AT.census.1971.male, mort.AT.census.1971.female,
     mort.AT.census.2011.male, mort.AT.census.2011.female,
     title="Austrian census tables",
     ylab=expression(q[x]), xlab="Age",
     xlim=c(0,90),
     legend.position=c(0.95,0.05))

# To compare period or cohort life tables, use the YOB and Period arguments:
plot(AVOe2005R.male, AVOe2005R.female, AVOe1996R.male, AVOe1996R.female,
    Period = 2018, title = "Austrian Annuity Tables, Period 2018")
plot(AVOe2005R.male, AVOe2005R.female, AVOe1996R.male, AVOe1996R.female,
    YOB = 2000, title = "Austrian Annuity Tables for cohort YOB=2000")
```

---

plotMortalityTrend *Plot the trends of multiple mortality tables (life tables) in one chart*

Description

plotMortalityTrend prints the trends of multiple life tables (objects of child classes of mortalityTable) in one plot, with a legend showing the names of the tables.

Usage

```
plotMortalityTrend(data, ..., xlim = NULL, ylim = NULL, xlab = NULL,
  ylab = NULL, title = "", legend.position = c(0.9, 0.9),
  legend.key.width = unit(25, "mm"))
```

Arguments

data	First life table to be plotted. Either a data.frame generated by makeQxDataFrame or a mortalityTable object
...	Additional life tables to be plotted (if data is a mortalityTable object)
xlim	X-axis limitatation (as a two-element vector)
ylim	Y-axis limitatation (as a two-element vector)
xlab	X-axis label (default: "Alter")
ylab	Y-axis label (default: "Sterbewahrscheinlichkeit q_x relativ zu ....")
title	The plot title
legend.position	The position of the legend (default is c(0.9,0.1))
legend.key.width	The keywith of the lines in the legend (default is unit(25,"mm"))

Examples

```
# Load the Austrian aunnity data
mortalityTables.load("Austria_Annuities")

# Compare the trends of these tables
plotMortalityTrend(AVOe2005R.male, AVOe2005R.female, AVOe1996R.male, AVOe1996R.female,
    Period = 2002, title = "Trends of Austrian Annuity Tables")
# For tables with a non-constant trend, the Period and YOB can be used to compare
# the age-specific trends that apply to the death probabilities during a given
# period or for a given birth year
plotMortalityTrend(AVOe2005R.male, AVOe2005R.female, AVOe1996R.male, AVOe1996R.female,
    YOB = 1950, title = "Trends of Austrian Annuity Tables for cohort YOB=1950")
plotMortalityTrend(AVOe2005R.male, AVOe2005R.female, AVOe1996R.male, AVOe1996R.female,
    YOB = 2000, title = "Trends of Austrian Annuity Tables for cohort YOB=2000")
plotMortalityTrend(AVOe2005R.male, AVOe2005R.female, AVOe1996R.male, AVOe1996R.female,
   Period = 1999, title = "Trends of Austrian Annuity Tables for Period 2002")
plotMortalityTrend(AVOe2005R.male, AVOe2005R.female, AVOe1996R.male, AVOe1996R.female,
   Period = 2030, title = "Trends of Austrian Annuity Tables for Period 2030")
#' @import scales
```

setLoading *Return a copy of the table with an additional loading added*

Description

Return a copy of the table with an additional loading added

Usage

```
setLoading(object, loading = 0)

## S4 method for signature 'mortalityTable'
setLoading(object, loading = 0)
```

Arguments

object	A life table object (instance of a mortalityTable class)
loading	The additional (security) loading to be added to the table.

Methods (by class)

- mortalityTable: Return the life table with the given loading set

Examples

```
mortalityTables.load("Austria_Census")
# Austrian census mortality 2011 reduced by 30%
setLoading(mort.AT.census.2011.male, loading = -0.3)
```

setModification *Return a copy of the table with the given modification function added*

Description

Return a copy of the table with the given modification function added

Usage

```
setModification(object, modification = 0)

## S4 method for signature 'mortalityTable'
setModification(object, modification = 0)
```

Arguments

object	A life table object (instance of a mortalityTable class)
modification	The postprocessing modification function (for example, so enforce a lower bound).

Methods (by class)

- mortalityTable: Return the life table with the given modification set

Examples

```
mortalityTables.load("Austria_Census")
# Austrian census mortality 2011, with a lower floor of 0.1% death probability
at11.mod1perm = setModification(mort.AT.census.2011.male,
    modification = function(qx) {pmax(qx, 0.001)})
at11.mod1perm@name = paste(at11.mod1perm@name, "at least 0.1%")
# Austrian census mortality 2011, modified with 40% selection for ages
# below 60, vanishing linearly to age 80
at11.modSelection = setModification(mort.AT.census.2011.male,
    modification = function(qx) {
        qx * c(rep(0.6, 60), 0.6 + 0.4 * (0:20)/20, rep(1, length(qx)-81))
    })
at11.modSelection@name = paste(at11.modSelection@name, " 40% selection below 60")

plot(mort.AT.census.2011.male, at11.mod1perm, at11.modSelection,
   title = "Austrian census mortality with modifications", legend.position = c(0.99, 0.01))
```

---

transitionProbabilities

Return all transition probabilities of the pension table (generational probabilities)

Description

Return all transition probabilities of the pension table (generational probabilities)

Usage

```
transitionProbabilities(object, ...)

## S4 method for signature 'pensionTable'
transitionProbabilities(object, YOB = 1982, ...,
  ages = NULL, OverallMortality = FALSE, Period = NULL,
  retirement = NULL, invalids.retire = object@invalids.retire,
  as.data.frame = TRUE)
```

Arguments

object	A pension table object (instance of a pensionTable class)
...	Currently unused
YOB	Year of birth
ages	Desired age range (if NULL, the probabilities of the age range provided by the table will be returned), missing ages will be filled with NA
OverallMortality	Whether the overall mortality should be returned for actives, or the active mortality
Period	Observation year to calculate period transition probabilities. If given, this arguments overrides the YOB parameter and this function returns period transition probabilities. If this argument is not given or is null, then this function returns generational transition probabilities.
retirement	Override the retirement transition probabilities of the pension table. Possible values are: • Single age (describing a deterministic retirement at the given age) • mortalityTable object: transition probabilities for retirement
invalids.retire	Override the pensionTable's invalids.retire flag, which indicates whether invalids retire like actives (i.e. same death probabilities after retirement) or stay invalid until death.
as.data.frame	Whether the return value should be a data.frame or an array containing transition matrices

Methods (by class)

- pensionTable: Return all transition probabilities of the pension table for the generation YOB

Examples

```
pensionTables.load("USA_PensionPlans")
transitionProbabilities(RP2014.male, YOB = 1962)
transitionProbabilities(RP2014.male, Period = 1955)
transitionProbabilities(RP2014.male, Period = 2025)
```

---

undampenTrend *Return a* mortalityTable.trendProjection *object with the trend damping removed.*

Description

Return a mortalityTable.trendProjection object with the trend damping removed.

Usage

```
undampenTrend(object)

## S4 method for signature 'mortalityTable.trendProjection'
undampenTrend(object)
```

Arguments

object	The life table object (class inherited from mortalityTable)

Methods (by class)

- mortalityTable.trendProjection: Return a mortalityTable.trendProjection object with the trend damping removed.

Examples

```
mortalityTables.load("Austria_Annuities")
AVOe2005R.male.undamped = undampenTrend(AVOe2005R.male)
AVOe2005R.male.undamped@name = paste(AVOe2005R.male.undamped@name, "no trend dampening")

plot(AVOe2005R.male, AVOe2005R.male.undamped,
    title = "AVOe 2005R with trend dampening and without", YOB = 2000)
```

---

whittaker.mortalityTable

Smooth a life table using the Whittaker-Henderson method, intepolation possibly missing values

---

Description

whittaker.mortalityTable uses the Whittaker-Henderson graduation method to smooth a table of raw observed death probabilities, optionally using the exposures stored in the table as weights (if no exposures are given, equal weights are applied). The weights (either explicitly given, implicitly taken from the exposures or implicit equal weights) will be normalized to sum 1. The parameter lambda indicates the importance of smootheness. A lower value of lambda will put more emphasis on reproducing the observation as good as possible at the cost of less smoothness. In turn, a higher value of lambda will force the smoothed result to be as smooth as possible with possibly larger deviation from the input data. All ages with a death probability of NA will be interpolated in the Whittaker-Henderson method (see e.g. Lowrie)

Usage

```
whittaker.mortalityTable(table, lambda = 10, d = 2,
  name.postfix = ", smoothed", ..., weights = NULL)
```

Arguments

table	Mortality table to be graduated. Must be an instance of a mortalityTable-derived class.
lambda	Smoothing parameter (default 10)
d	order of differences (default 2)
name.postfix	Postfix appended to the name of the graduated table
...	additional arguments (currently unused)
weights	Vector of weights used for graduation. Entries with weight 0 will be interpolated. If not given, the exposures of the table or equal weights are used. Weight 0 for a certain age indicates that the observation will not be used for smoothing at all, and will rather be interpolated from the smoothing of all other values.

References

Walter B. Lowrie: An Extension of the Whittaker-Henderson Method of Graduation, Transactions of Society of Actuaries, 1982, Vol. 34, pp. 329–372

See Also

whittaker

Examples

```
# A sample observation table with exposures and raw probabilities
obsTable = mortalityTable.period(
    name = "trivial observed table",
    ages = 0:15,
    deathProbs = c(
        0.0072, 0.00212, 0.00081, 0.0005, 0.0013,
        0.001, 0.00122, 0.00142, 0.007, 0.0043,
        0.0058, 0.0067, 0.0082, 0.0091, 0.0075, 0.01),
    exposures = c(
        150, 222, 350, 362, 542,
        682, 1022, 1053, 1103, 1037,
        968, 736, 822, 701, 653, 438))

# Effect of the different parameters
obsTable.smooth = whittaker.mortalityTable(obsTable,
    lambda = 1/10, d = 2, name.postfix = " smoothed (d=2, lambda=1/10)")
obsTable.smooth1 = whittaker.mortalityTable(obsTable,
    lambda = 1, d = 2, name.postfix = " smoothed (d=2, lambda=1)")
obsTable.smooth2 = whittaker.mortalityTable(obsTable,
    lambda = 1/10, d = 3, name.postfix = " smoothed (d=3, lambda=1/10)")
plot(obsTable, obsTable.smooth, obsTable.smooth1, obsTable.smooth2,
    title = "Observed death probabilities")

# Missing values are interpolated from the Whittaker Henderson
obsTable.missing = obsTable
obsTable.missing@deathProbs[c(6,10,11,12)] = NA_real_
```

```
obsTable.interpolated = whittaker.mortalityTable(obsTable,
    lambda = 1/10, d = 2, name.postfix = " missing values interpolated")
plot(obsTable.missing, obsTable.interpolated,
    title = "Missing values are automatically interpolated") + geom_point(size = 3)
```

Index