Package 'mqriskR'

July 28, 2026

Type Package

Title Actuarial Risk Modeling and Life Contingencies

Version 0.1.1

Description Provides functions for actuarial risk modeling, including survival models, life annuities, multiple-decrement models, and mortality improvement projections. The package is designed to align with standard actuarial notation and supports teaching, exam preparation, and reproducible actuarial analysis. The methods are based on standard actuarial references including Camilli, Duncan and London (2014, ISBN:9781625423474) ``Models for Quantifying Risk" and Dickson, Hardy and Waters (2020, ISBN:9781108478083) ``Actuarial Mathematics for Life Contingent Risks".

License MIT + file LICENSE

Encoding UTF-8

Suggests ggplot2, testthat (>= 3.0.0), expm, knitr, rmarkdown

Config/testthat/edition 3

Imports stats, utils

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Nii Okine [aut, cre]

Maintainer Nii Okine <okinean@appstate.edu>

Repository CRAN

Date/Publication 2026-07-28 00:10:01 UTC

Contents

`A2barx` *Second moment of continuous whole life insurance PV*

Description

Computes ${}^2\bar{A}_x$ by evaluating $\bar{A}_x$ at doubled force.

Usage

```
A2barx(x, i, model, ...)
```

Arguments

`x`	Age.
`i`	Effective annual interest rate.
`model`	Parametric survival model name.
`...`	Additional model parameters passed to survival-model functions.

Value

Numeric vector of second moments.

`A2barxn` *Second moment of continuous endowment insurance PV*

Description

Computes ${}^2\bar{A}_{x:\overline{n}|}$.

Usage

```
A2barxn(x, n, i, model, ...)
```

Arguments

`x`	Age.
`n`	Term.
`i`	Effective annual interest rate.
`model`	Parametric survival model name.
`...`	Additional model parameters passed to survival-model functions.

Value

Numeric vector of second moments.

`A2barxn1` *Second moment of continuous term insurance PV*

Description

Computes ${}^2\bar{A}^1_{x:\overline{n}|}$ by evaluating $\bar{A}^1_{x:\overline{n}|}$ at doubled force.

Usage

```
A2barxn1(x, n, i, model, ...)
```

Arguments

`x`	Age.
`n`	Term.
`i`	Effective annual interest rate.
`model`	Parametric survival model name.
`...`	Additional model parameters passed to survival-model functions.

Value

Numeric vector of second moments.

`A2nAbarx` *Second moment of continuous deferred insurance PV*

Description

Computes ${}^{2}{}_{n|}\bar{A}_x$ by evaluating ${}_{n|}\bar{A}_x$ at doubled force.

Usage

```
A2nAbarx(x, n, i, model, ...)
```

Arguments

`x`	Age.
`n`	Deferral period.
`i`	Effective annual interest rate.
`model`	Parametric survival model name.
`...`	Additional model parameters passed to survival-model functions.

Value

Numeric vector of second moments.

`A2nAx` *Second moment of deferred insurance PV*

Description

Computes ${}^{2}{}_{n|}A_x$ by evaluating ${}_{n|}A_x$ at doubled force.

Usage

```
A2nAx(x, n, i, tbl = NULL, model = NULL, ..., tol = 1e-12, k_max = 5000)
```

Arguments

`x`	Age.
`n`	Deferral period.
`i`	Effective annual interest rate.
`tbl`	Optional life table object.
`model`	Optional parametric survival model name.
`...`	Additional arguments passed to survival-model functions.
`tol`	Numerical tolerance for truncating infinite sums.
`k_max`	Maximum number of terms in the sum.

Value

Numeric vector of second moments.

`A2nAx_m` *Second moment of m-thly deferred insurance PV*

Description

Second moment of m-thly deferred insurance PV

Usage

```
A2nAx_m(x, n, i, m, model, ..., tol = 1e-12, j_max = 100000L)
```

Arguments

`x`	Age.
`n`	Deferral period.
`i`	Effective annual interest rate.
`m`	Positive integer payment frequency.
`model`	Parametric survival model name.
`...`	Additional model parameters passed to survival-model functions.
`tol`	Numerical tolerance for truncating the infinite sum.
`j_max`	Maximum number of m-thly intervals in the sum.

Value

Numeric vector of second moments.

`A2nEx` *Second moment of pure endowment PV*

Description

Computes ${}^2{}_nE_x = (v')^n {}_np_x$.

Usage

```
A2nEx(x, n, i, tbl = NULL, model = NULL, ...)
```

Arguments

x	Age.
n	Term.
i	Effective annual interest rate.
tbl	Optional life table object.
model	Optional parametric survival model name.
...	Additional arguments passed to survival-model functions.

Value

Numeric vector of second moments.

A2x *Second moment of whole life insurance PV*

Description

Computes ${}^2A_x$ by evaluating A_x at doubled force.

Usage

```
A2x(x, i, tbl = NULL, model = NULL, ..., tol = 1e-12, k_max = 5000)
```

Arguments

x	Age.
i	Effective annual interest rate.
tbl	Optional life table object.
model	Optional parametric survival model name.
...	Additional arguments passed to survival-model functions.
tol	Numerical tolerance for truncating infinite sums.
k_max	Maximum number of terms in the sum.

Value

Numeric vector of second moments.

A2xn — *Second moment of endowment insurance PV*

Description

Computes ${}^2A_{x:\overline{n}|}$.

Usage

```
A2xn(x, n, i, tbl = NULL, model = NULL, ...)
```

Arguments

x	Age.
n	Term.
i	Effective annual interest rate.
tbl	Optional life table object.
model	Optional parametric survival model name.
...	Additional arguments passed to survival-model functions.

Value

Numeric vector of second moments.

A2xn1 — *Second moment of term insurance PV*

Description

Computes ${}^2A^1_{x:\overline{n}|}$ by evaluating $A^1_{x:\overline{n}|}$ at doubled force.

Usage

```
A2xn1(x, n, i, tbl = NULL, model = NULL, ...)
```

Arguments

x	Age.
n	Term.
i	Effective annual interest rate.
tbl	Optional life table object.
model	Optional parametric survival model name.
...	Additional arguments passed to survival-model functions.

Value

Numeric vector of second moments.

`A2xn1_m` *Second moment of m-thly term insurance PV*

Description

Second moment of m-thly term insurance PV

Usage

```
A2xn1_m(x, n, i, m, model, ...)
```

Arguments

`x`	Age.
`n`	Term.
`i`	Effective annual interest rate.
`m`	Positive integer payment frequency.
`model`	Parametric survival model name.
`...`	Additional model parameters passed to survival-model functions.

Value

Numeric vector of second moments.

`A2xn_m` *Second moment of m-thly endowment insurance PV*

Description

Second moment of m-thly endowment insurance PV

Usage

```
A2xn_m(x, n, i, m, model, ...)
```

Arguments

`x`	Age.
`n`	Term.
`i`	Effective annual interest rate.
`m`	Positive integer payment frequency.
`model`	Parametric survival model name.
`...`	Additional model parameters passed to survival-model functions.

Value

Numeric vector of second moments.

`A2x_m` *Second moment of m-thly whole life insurance PV*

Description

Computes ${}^2A_x^{(m)}$ by evaluating $A_x^{(m)}$ at doubled force.

Usage

```
A2x_m(x, i, m, model, ..., tol = 1e-12, j_max = 100000L)
```

Arguments

`x`	Age.
`i`	Effective annual interest rate.
`m`	Positive integer payment frequency.
`model`	Parametric survival model name.
`...`	Additional model parameters passed to survival-model functions.
`tol`	Numerical tolerance for truncating the infinite sum.
`j_max`	Maximum number of m-thly intervals in the sum.

Value

Numeric vector of second moments.

`AAL_PUC_db` *Projected Unit Credit accrued liability*

Description

Computes the actuarial present value of the portion of the projected benefit attributed to past service.

Usage

```
AAL_PUC_db(
  projected_benefit,
  past_service,
  total_service,
  v_to_ret,
  p_surv,
  adue_ret
)
```

Arguments

projected_benefit	Nonnegative projected annual benefit at retirement.
past_service	Nonnegative service completed through the valuation date.
total_service	Positive total service at retirement.
v_to_ret	Nonnegative discount factor from the valuation date to retirement.
p_surv	Survival or active-service probability to retirement in [0, 1].
adue_ret	Positive retirement annuity-due factor.

Value

A numeric vector.

Examples

```
AAL_PUC_db(
  projected_benefit = 30000,
  past_service = 10,
  total_service = 30,
  v_to_ret = 0.5,
  p_surv = 0.9,
  adue_ret = 12
)
```

AAL_TUC_db *Traditional Unit Credit accrued liability*

Description

Computes the actuarial present value of the benefit accrued through the valuation date.

Usage

```
AAL_TUC_db(accrued_benefit, v_to_ret, p_surv, adue_ret)
```

Arguments

accrued_benefit	Nonnegative accrued benefit.
v_to_ret	Nonnegative discount factor from the valuation date to retirement.
p_surv	Survival or active-service probability to retirement in [0, 1].
adue_ret	Positive retirement annuity-due factor.

Value

A numeric vector.

Examples

```
AAL_TUC_db(
  accrued_benefit = 12000,
  v_to_ret = 0.5,
  p_surv = 0.9,
  adue_ret = 12
)
```

`Abarx` *Continuous whole life insurance APV*

Description

Computes $\bar{A}_x = \int_0^\infty v^t {}_tp_x \mu_{x+t}\, dt$.

Usage

```
Abarx(x, i, model, ...)
```

Arguments

`x`	Age.
`i`	Effective annual interest rate.
`model`	Parametric survival model name.
`...`	Additional model parameters passed to survival-model functions.

Value

Numeric vector of APVs.

Abarxj_md *Continuous multiple-decrement insurance present value*

Description

Approximates the actuarial present value of a benefit payable at the moment of decrement using the trapezoidal rule.

Usage

```
Abarxj_md(t, ptau, muj, delta, benefit = 1)
```

Arguments

t	Strictly increasing nonnegative time points.
ptau	In-force probabilities at the supplied time points.
muj	Cause-specific decrement intensities at the supplied time points.
delta	Force of interest. May be scalar or vector.
benefit	Benefit payable on decrement. May be scalar or vector.

Value

A numeric vector of actuarial present values.

Examples

```
t <- seq(0, 20, by = 0.1)
ptau <- exp(-0.012 * t)
muj <- rep(0.002, length(t))
Abarxj_md(t, ptau, muj, delta = 0.05, benefit = 2000)
```

Abarxn *Continuous endowment insurance APV*

Description

Computes $\bar{A}_{x:\overline{n}|} = \bar{A}^{1}_{x:\overline{n}|} + v^{n}{}_{n}p_{x}$.

Usage

```
Abarxn(x, n, i, model, ...)
```

Arguments

x	Age.
n	Term.
i	Effective annual interest rate.
model	Parametric survival model name.
...	Additional model parameters passed to survival-model functions.

Value

Numeric vector of APVs.

Abarxn1 *Continuous term insurance APV*

Description

Computes $\bar{A}^1_{x:\overline{n}|} = \int_0^n v^t {}_tp_x \mu_{x+t}\, dt$.

Usage

```
Abarxn1(x, n, i, model, ...)
```

Arguments

x	Age.
n	Term.
i	Effective annual interest rate.
model	Parametric survival model name.
...	Additional model parameters passed to survival-model functions.

Value

Numeric vector of APVs.

Abarxn1_udd *UDD approximation of continuous term insurance*

Description

Computes $\bar{A}^1_{x:\overline{n}|} = (i/\delta)A^1_{x:\overline{n}|}$.

Usage

```
Abarxn1_udd(Axn1, i)
```

Arguments

Axn1	Discrete term insurance APV.
i	Effective annual interest rate.

Value

Continuous term insurance APV under UDD.

Abarxn_udd *UDD approximation of continuous endowment insurance*

Description

Computes $\bar{A}_{x:\overline{n}|} = (i/\delta)A^1_{x:\overline{n}|} + {}_nE_x$.

Usage

```
Abarxn_udd(Axn1, nEx, i)
```

Arguments

Axn1	Discrete term insurance APV.
nEx	Pure endowment APV, ${}_nE_x$.
i	Effective annual interest rate.

Value

Continuous endowment insurance APV under UDD.

Abarx_udd *UDD approximation of continuous whole life insurance*

Description

Computes $\bar{A}_x = (i/\delta)A_x$.

Usage

```
Abarx_udd(Ax, i)
```

Arguments

Ax	Discrete whole life insurance APV.
i	Effective annual interest rate.

Value

Continuous whole life insurance APV under UDD.

AB_cae *Accrued benefit under a career-average-earnings plan*

Description

Computes the accrued benefit using salary history through the valuation date.

Usage

```
AB_cae(salary_history, p)
```

Arguments

salary_history	Positive numeric vector of annual salaries.
p	Nonnegative scalar accrual percentage.

Value

A numeric scalar.

Examples

```
AB_cae(
  salary_history = c(100000, 104000, 108160),
  p = 1
)
```

AB_fas *Accrued benefit under a final-average-salary plan*

Description

Computes the accrued benefit using salary and service history through the valuation date.

Usage

```
AB_fas(salary_history, p, fas_years = 3)
```

Arguments

salary_history	Positive numeric vector of annual salaries.
p	Nonnegative scalar accrual percentage.
fas_years	Positive integer number of years in the salary average.

Value

A numeric scalar.

Examples

```
AB_fas(
  salary_history = c(150000, 156000),
  p = 1,
  fas_years = 2
)
```

ag38_prefunding_ratio *AG 38 prefunding ratio*

Description

Computes the excess-payment-to-required-net-single-premium ratio, capped at one.

Usage

```
ag38_prefunding_ratio(excess_payment, nsp_required)
```

Arguments

excess_payment	Nonnegative excess payment or shadow-fund amount.
nsp_required	Positive net single premium required to fully fund the guarantee.

Value

A numeric vector.

Examples

```
ag38_prefunding_ratio(60000, 100000)
```

`ag38_reserve_ul` *AG 38 reserve calculation*

Description

Computes the prefunding ratio, net additional amount, reduced deficiency reserve, intermediate reserve, and increased basic reserve.

Usage

```
ag38_reserve_ul(
  basic_reserve,
  deficiency_reserve = 0,
  excess_payment,
  nsp_required,
  valuation_nsp,
  surrender_charge = 0
)
```

Arguments

`basic_reserve`	Nonnegative basic reserve.
`deficiency_reserve`	Nonnegative deficiency reserve.
`excess_payment`	Nonnegative excess payment or shadow-fund amount.
`nsp_required`	Positive net single premium required to fully fund the guarantee.
`valuation_nsp`	Nonnegative valuation net single premium.
`surrender_charge`	Nonnegative surrender charge.

Details

Scalar inputs preserve the original named-list output. Vectorized inputs return a data frame with one row per calculation.

Value

For scalar inputs, a named list. For vectorized inputs, a data frame containing the same calculated quantities.

Examples

```
ag38_reserve_ul(
  basic_reserve = 10000,
  deficiency_reserve = 0,
  excess_payment = 60000,
  nsp_required = 100000,
  valuation_nsp = 150000,
  surrender_charge = 5000
)
```

---

annuity_annual *Annual annuity functions*

Description

Annual whole life, temporary, deferred, and actuarial accumulated value annuity functions in immediate, due, and continuous forms.

Computes $a_x = \sum_{t=1}^{\infty} v^t {}_tp_x$.

Computes $\ddot{a}_x = \sum_{t=0}^{\infty} v^t {}_tp_x$.

Computes $\bar{a}_x = \int_0^{\infty} v^t {}_tp_x dt$.

Computes $a_{x:\overline{n}|} = \sum_{t=1}^{n} v^t {}_tp_x$.

Computes $\ddot{a}_{x:\overline{n}|} = \sum_{t=0}^{n-1} v^t {}_tp_x$.

Computes $\bar{a}_{x:\overline{n}|} = \int_0^{n} v^t {}_tp_x dt$.

Computes ${}_{n|}a_x = {}_nE_x a_{x+n}$.

Computes ${}_{n|}\ddot{a}_x = {}_nE_x \ddot{a}_{x+n}$.

Computes ${}_{n|}\bar{a}_x = {}_nE_x \bar{a}_{x+n}$.

Computes $s_{x:\overline{n}|} = a_{x:\overline{n}|}/{}_nE_x$.

Computes $\ddot{s}_{x:\overline{n}|} = \ddot{a}_{x:\overline{n}|}/{}_nE_x$.

Computes $\bar{s}_{x:\overline{n}|} = \bar{a}_{x:\overline{n}|}/{}_nE_x$.

Usage

```
ax(x, i, model = NULL, ..., tbl = NULL, k_max = 5000, tol = 1e-12)

adotx(x, i, model = NULL, ..., tbl = NULL, k_max = 5000, tol = 1e-12)

abarx(x, i, model, ..., tol = 1e-10)

axn(x, n, i, model = NULL, ..., tbl = NULL)

adotxn(x, n, i, model = NULL, ..., tbl = NULL)
```

```
abarxn(x, n, i, model, ...)

nax(x, n, i, model = NULL, ..., tbl = NULL, k_max = 5000, tol = 1e-12)

nadotx(x, n, i, model = NULL, ..., tbl = NULL, k_max = 5000, tol = 1e-12)

nabarx(x, n, i, model, ..., tol = 1e-10)

sxn(x, n, i, model = NULL, ..., tbl = NULL)

sdotxn(x, n, i, model = NULL, ..., tbl = NULL)

sbarxn(x, n, i, model, ...)
```

Arguments

x	Age.
i	Effective annual interest rate.
model	Optional survival model name.
...	Additional model parameters.
tbl	Optional life table object for annual discrete annuity functions.
k_max	Maximum summation horizon for non-terminating models.
tol	Truncation tolerance for non-terminating models.
n	Term in years.

Value

Numeric vector containing the requested annuity present value or actuarial accumulated value.

`annuity_approximations_udd` *UDD annuity approximations*

Description

UDD-based approximations for annual, fractional-payment, and continuous annuity functions.

Computes

$$\ddot{a}_x^{(m)} \approx \alpha(m)\ddot{a}_x - \beta(m).$$

Computes

$$\ddot{a}_{x:\overline{n}|}^{(m)} \approx \alpha(m)\ddot{a}_{x:\overline{n}|} - \beta(m)(1 - {}_nE_x).$$

Computes

$${}_{n|}\ddot{a}_x^{(m)} \approx \alpha(m)\,{}_{n|}\ddot{a}_x - \beta(m)\,{}_nE_x.$$

Computes

$$a_x^{(m)} \approx \alpha(m) a_x + \gamma(m).$$

Computes

$$a_{x:\overline{n}|}^{(m)} \approx \alpha(m) a_{x:\overline{n}|} + \gamma(m)(1 - {}_nE_x).$$

Computes

$${}_{n|}a_x^{(m)} \approx \alpha(m)\, {}_{n|}a_x + \gamma(m)\, {}_nE_x.$$

Computes

$$\ddot{s}_{x:\overline{n}|}^{(m)} \approx \alpha(m) \ddot{s}_{x:\overline{n}|} - \beta(m) \left(\frac{1}{{}_nE_x} - 1 \right).$$

Computes

$$s_{x:\overline{n}|}^{(m)} \approx \alpha(m) s_{x:\overline{n}|} + \gamma(m) \left(\frac{1}{{}_nE_x} - 1 \right).$$

Computes

$$\bar{a}_x \approx \frac{id}{\delta^2} \ddot{a}_x - \frac{i - \delta}{\delta^2}.$$

Uses the identity

$$\bar{a}_{x:\overline{n}|} \approx \frac{1 - \bar{A}_{x:\overline{n}|}}{\delta}$$

together with the package's existing Abarxn_udd() approximation.

Computes

$${}_{n|}\bar{a}_x \approx {}_nE_x\, \bar{a}_{x+n}.$$

Usage

```
adotx_m_udd(x, m, i, model, ...)

adotxn_m_udd(x, n, m, i, model, ...)

nadotx_m_udd(x, n, m, i, model, ...)

ax_m_udd(x, m, i, model, ...)

axn_m_udd(x, n, m, i, model, ...)

nax_m_udd(x, n, m, i, model, ...)

sdotxn_m_udd(x, n, m, i, model, ...)

sxn_m_udd(x, n, m, i, model, ...)

abarx_udd(x, i, model, ...)

abarxn_udd(x, n, i, model, ...)

nabarx_udd(x, n, i, model, ...)
```

Arguments

x	Age.
m	Number of payments per year.
i	Effective annual interest rate.
model	Survival model.
...	Additional model parameters.
n	Term.

Details

These functions implement the standard Uniform Distribution of Deaths approximations linking annual, m-thly, and continuous annuity values.

The exported functions documented on this page are:

- ax_m_udd()
- axn_m_udd()
- nax_m_udd()
- adotx_m_udd()
- adotxn_m_udd()
- nadotx_m_udd()
- sxn_m_udd()
- sdotxn_m_udd()
- abarx_udd()
- abarxn_udd()
- nabarx_udd()

The abarxn_udd() calculation uses the existing Abarxn_udd() insurance approximation. Consequently, additional survival-model arguments are not used in that calculation.

Value

Numeric vector.

`annuity_approximations_woolhouse2` *Woolhouse 2-term annuity approximations*

Description

Woolhouse 2-term approximations for fractional-payment annuity functions.

Usage

```
ax_m_woolhouse2(x, m, i, model, ...)

adotx_m_woolhouse2(x, m, i, model, ...)

nax_m_woolhouse2(x, n, m, i, model, ...)

nadotx_m_woolhouse2(x, n, m, i, model, ...)

axn_m_woolhouse2(x, n, m, i, model, ...)

adotxn_m_woolhouse2(x, n, m, i, model, ...)

sxn_m_woolhouse2(x, n, m, i, model, ...)

sdotxn_m_woolhouse2(x, n, m, i, model, ...)

abarx_woolhouse2(x, i, model, ...)
```

Arguments

`x`	Age.
`m`	Number of payments per year.
`i`	Effective annual interest rate.
`model`	Survival model.
`...`	Additional model parameters.
`n`	Term.

Value

Numeric vector.

`annuity_approximations_woolhouse3` *Woolhouse 3-term annuity approximations*

Description

Woolhouse 3-term approximations for fractional-payment annuity functions.

Usage

```
ax_m_woolhouse3(x, m, i, model, ...)

adotx_m_woolhouse3(x, m, i, model, ...)

nax_m_woolhouse3(x, n, m, i, model, ...)

nadotx_m_woolhouse3(x, n, m, i, model, ...)

axn_m_woolhouse3(x, n, m, i, model, ...)

adotxn_m_woolhouse3(x, n, m, i, model, ...)

abarx_woolhouse3(x, i, model, ...)
```

Arguments

`x`	Age.
`m`	Number of payments per year.
`i`	Effective annual interest rate.
`model`	Survival model.
`...`	Additional model parameters.
`n`	Term.

Value

Numeric vector.

annuity_certain *Present value of a level annuity-certain*

Description

Computes the present value of an n-period annuity certain with level payments of 1 per period.

Usage

```
annuity_certain(n, i, due = FALSE, m = 1, cont = FALSE)
```

Arguments

n	Number of payments or periods. May be scalar or vector.
i	Effective interest rate per period. May be scalar or vector.
due	If 'TRUE', annuity-due; otherwise annuity-immediate.
m	Payment frequency per period. 'm = 1' means annual.
cont	If 'TRUE', use the continuous payment model.

Value

Numeric vector of present values.

Examples

```
annuity_certain(n = 10, i = 0.05)
annuity_certain(n = c(5, 10), i = 0.05)
annuity_certain(n = 10, i = c(0.03, 0.05))
annuity_certain(n = 10, i = 0.05, due = TRUE)
annuity_certain(n = 10, i = 0.05, cont = TRUE)
```

annuity_mthly *m-thly contingent annuity functions*

Description

Life annuities payable m-thly.

Usage

```
ax_m(x, m, i, model, ..., k_max = 2e+05, tol = 1e-12)

adotx_m(x, m, i, model, ..., k_max = 2e+05, tol = 1e-12)

axn_m(x, n, m, i, model, ...)

adotxn_m(x, n, m, i, model, ...)

nax_m(x, n, m, i, model, ..., k_max = 2e+05, tol = 1e-12)

nadotx_m(x, n, m, i, model, ..., k_max = 2e+05, tol = 1e-12)

sxn_m(x, n, m, i, model, ...)

sdotxn_m(x, n, m, i, model, ...)
```

Arguments

x	Age.
m	Number of payments per year.
i	Effective annual interest rate.
model	Survival model name.
...	Additional parameters passed to the survival model.
k_max	Maximum summation horizon for non-terminating models.
tol	Truncation tolerance for non-terminating models.
n	Term or deferral period in years.

Value

Numeric vector containing the requested m-thly annuity value or actuarial accumulated value.

---

annuity_relationships *Annuity-insurance relationships*

Description

Identities linking annual and continuous annuity functions to the corresponding insurance functions.

Computes $a_x = (v - A_x)/d$.

Computes $\ddot{a}_x = (1 - A_x)/d$.

Computes $\bar{a}_x = (1 - \bar{A}_x)/\delta$.

Computes $a_{x:\overline{n}|} = (1 - A_{x:\overline{n}|})/d - 1 + {}_nE_x$.

Computes $\ddot{a}_{x:\overline{n}|} = (1 - A_{x:\overline{n}|})/d$.

Computes $\bar{a}_{x:\overline{n}|} = (1 - \bar{A}_{x:\overline{n}|})/\delta$.

Computes ${}_{n|}a_x = {}_nE_x a_{x+n}$.

Computes ${}_{n|}\ddot{a}_x = {}_nE_x \ddot{a}_{x+n}$.

Computes ${}_{n|}\bar{a}_x = {}_nE_x \bar{a}_{x+n}$.

Usage

```
annuity_identity_ax(x, i, model = NULL, ..., tbl = NULL)

annuity_identity_adotx(x, i, model = NULL, ..., tbl = NULL)

annuity_identity_abarx(x, i, model, ...)

annuity_identity_axn(x, n, i, model = NULL, ..., tbl = NULL)

annuity_identity_adotxn(x, n, i, model = NULL, ..., tbl = NULL)

annuity_identity_abarxn(x, n, i, model, ...)

annuity_identity_nax(
  x,
  n,
  i,
  model = NULL,
  ...,
  tbl = NULL,
  k_max = 5000,
  tol = 1e-12
)

annuity_identity_nadotx(
  x,
  n,
  i,
  model = NULL,
  ...,
  tbl = NULL,
  k_max = 5000,
  tol = 1e-12
)

annuity_identity_nabarx(x, n, i, model, ..., tol = 1e-10)
```

Arguments

x	Age.
i	Effective annual interest rate.

model	Optional survival model name.
...	Additional model parameters.
tbl	Optional life table object for discrete identities.
n	Term or deferral period in years.
k_max	Maximum summation horizon for non-terminating models.
tol	Truncation tolerance for non-terminating models.

Value

Numeric vector containing the annuity value computed from the corresponding annuity-insurance identity.

`annuity_varying_payments` *Varying-payment annuity functions*

Description

Increasing and decreasing life annuity functions.

Usage

```
Iax(x, i, model = NULL, ..., tbl = NULL, k_max = 5000, tol = 1e-12)

Iaxn(x, n, i, model = NULL, ..., tbl = NULL)

Daxn(x, n, i, model = NULL, ..., tbl = NULL)

Iadotx(x, i, model = NULL, ..., tbl = NULL, k_max = 5000, tol = 1e-12)

Iadotxn(x, n, i, model = NULL, ..., tbl = NULL)

Dadotxn(x, n, i, model = NULL, ..., tbl = NULL)

Iabarx(x, i, model, ..., tol = 1e-10)

Iabarxn(x, n, i, model, ...)

Dabarxn(x, n, i, model, ...)
```

Arguments

x	Age.
i	Effective annual interest rate.
model	Optional survival model name.

...	Additional model parameters.
tbl	Optional life table object for discrete functions.
k_max	Maximum summation horizon for non-terminating models.
tol	Truncation tolerance for non-terminating models.
n	Term in years.

Value

Numeric vector containing the requested increasing or decreasing annuity value.

APV_gross_premiums *Actuarial present value of gross premiums*

Description

Computes the actuarial present value of premiums weighted by contract persistency at the start of each policy year.

Usage

```
APV_gross_premiums(G, r, p_tau)
```

Arguments

G	Nonnegative gross premium vector.
r	Annual effective risk discount rate. May be scalar or vector; values must be greater than -1.
p_tau	One-year in-force probabilities. For n > 1, this may have length n - 1 or n; the final value is ignored when length n. For one premium, use numeric(0) or one value.

Value

A numeric vector with one value for each rate in r.

Examples

```
APV_gross_premiums(
  G = rep(95, 3),
  r = 0.10,
  p_tau = c(0.99858, 0.99847, 0.99834)
)
```

APV_NR_db *Actuarial present value of a normal retirement benefit*

Description

Computes the actuarial present value of a projected annual retirement benefit. Scalar arguments are recycled to a common length.

Usage

```
APV_NR_db(PABz, v_to_ret, p_surv, adue_ret)
```

Arguments

PABz	Nonnegative projected annual benefit at retirement.
v_to_ret	Nonnegative discount factor from the valuation date to retirement.
p_surv	Survival or active-service probability to retirement in [0, 1].
adue_ret	Positive retirement annuity-due factor.

Value

A numeric vector.

Examples

```
APV_NR_db(
  PABz = 108008.66,
  v_to_ret = 1 / 1.06^30,
  p_surv = 0.8,
  adue_ret = 12
)
```

AS_path *Projected asset-share path for two decrement causes*

Description

Convenience interface to AS_path_md() for two causes.

Usage

```
AS_path(AS0, G, r, e, b1, b2, q1, q2, p_tau, i, b3 = NULL)
```

Arguments

AS0	Initial asset share.
G	Premium amount by policy year.
r	Percent-of-premium expense rate by policy year.
e	Fixed expense by policy year.
b1	Cause 1 benefit by policy year.
b2	Cause 2 benefit by policy year.
q1	Cause 1 probability by policy year.
q2	Cause 2 probability by policy year.
p_tau	In-force probability by policy year.
i	Effective annual interest rate by policy year.
b3	Survival benefit by policy year.

Value

A data frame with policy year k and asset share AS.

AS_path_md *General projected asset-share path*

Description

Computes projected asset shares for a multiple-decrement contract. Rows of b_mat and q_mat represent policy years and columns represent decrement causes.

Usage

```
AS_path_md(AS0, G, r, e, b_mat, q_mat, p_tau, i, b_surv = NULL)
```

Arguments

AS0	Initial asset share.
G	Premium amount by policy year.
r	Percent-of-premium expense rate by policy year.
e	Fixed expense by policy year.
b_mat	Matrix of decrement benefits.
q_mat	Matrix of decrement probabilities.
p_tau	In-force probability by policy year.
i	Effective annual interest rate by policy year.
b_surv	Survival benefit by policy year.

Value

A data frame with policy year k and asset share AS.

AVz_dc *Accumulated value of defined contribution plan contributions*

Description

Computes the accumulated value at retirement from contributions paid at the beginning of each year and accumulated to retirement.

Usage

```
AVz_dc(x, z, Sx, c, i, g = NULL, s = NULL)
```

Arguments

x	Scalar entry age.
z	Scalar retirement age.
Sx	Positive scalar salary at age x.
c	Scalar contribution rate in [0, 1].
i	Scalar annual effective investment return greater than -1.
g	Optional scalar annual salary growth rate greater than -1.
s	Optional positive salary-scale vector of length z - x.

Value

A numeric scalar.

Examples

```
AVz_dc(
  x = 30,
  z = 65,
  Sx = 50000,
  c = 0.10,
  i = 0.05,
  g = 0.04
)
```

`AV_path_ul_typeA` *Type A universal life account-value path*

Description

Computes a year-by-year Type A universal life account-value path. The death benefit is fixed, so the net amount at risk depends on the ending account value and the roll-forward is solved explicitly each period.

Usage

```
AV_path_ul_typeA(G, r, e, qx, ic, B, iq = ic, AV0 = 0)
```

Arguments

`G`	Premium amount by period.
`r`	Percent-of-premium expense rate by period. Values must lie in `[0, 1]`.
`e`	Fixed expense by period.
`qx`	Mortality probability by period.
`ic`	Credited annual effective interest rate by period. Values must be greater than `-1`.
`B`	Face amount by period.
`iq`	Interest rate used in the cost-of-insurance calculation. Defaults to `ic`; values must be greater than `-1`.
`AV0`	Nonnegative scalar initial account value.

Value

A data frame containing the policy duration, premium, and account value.

Examples

```
qx <- c(0.00076, 0.00081, 0.00085, 0.00090, 0.00095)
r <- c(0.75, rep(0.10, 4))
e <- c(100, rep(20, 4))

AV_path_ul_typeA(
  G = 5000,
  r = r,
  e = e,
  qx = qx,
  ic = 0.03,
  B = 100000
)
```

`AV_path_ul_typeB` *Type B universal life account-value path*

Description

Computes a year-by-year Type B universal life account-value path. Premiums and expenses are applied at the beginning of each period, followed by the cost-of-insurance charge and credited interest.

Usage

```
AV_path_ul_typeB(G, r, e, qx, ic, B, iq = ic, AV0 = 0)
```

Arguments

`G`	Premium amount by period.
`r`	Percent-of-premium expense rate by period. Values must lie in `[0, 1]`.
`e`	Fixed expense by period.
`qx`	Mortality probability by period.
`ic`	Credited annual effective interest rate by period. Values must be greater than `-1`.
`B`	Face amount by period.
`iq`	Interest rate used in the cost-of-insurance calculation. Defaults to `ic`; values must be greater than `-1`.
`AV0`	Nonnegative scalar initial account value.

Value

A data frame containing the policy duration, premium, net contribution, cost-of-insurance charge, and account value.

Examples

```
qx <- c(0.00076, 0.00081, 0.00085, 0.00090, 0.00095)
r <- c(0.75, rep(0.10, 4))
e <- c(100, rep(20, 4))

AV_path_ul_typeB(
  G = 5000,
  r = r,
  e = e,
  qx = qx,
  ic = 0.03,
  B = 100000
)
```

Ax *Whole life insurance APV*

Description

Computes $A_x = \sum_{k=0}^{\infty} v^{k+1}{}_{k|}q_x$.

Usage

```
Ax(x, i, tbl = NULL, model = NULL, ..., tol = 1e-12, k_max = 5000)
```

Arguments

x	Age.
i	Effective annual interest rate.
tbl	Optional life table object.
model	Optional parametric survival model name.
...	Additional arguments passed to survival-model functions.
tol	Numerical tolerance for truncating infinite sums.
k_max	Maximum number of terms in the sum.

Value

Numeric vector of APVs.

Axj_md *Discrete multiple-decrement insurance present value*

Description

Computes the actuarial present value of a benefit payable at the end of the year of decrement from a specified cause.

Usage

```
Axj_md(qj, ptau, i, benefit = 1)
```

Arguments

qj	Cause-specific decrement probabilities by policy year.
ptau	In-force probabilities at the beginning of each policy year.
i	Effective annual interest rate. May be scalar or vector.
benefit	Benefit payable on decrement. May be scalar or vector.

Value

A numeric vector of actuarial present values.

Examples

```
qj <- c(0.02, 0.02, 0.02, 0.02, 0.02)
ptau <- c(1, 0.95, 0.89, 0.82, 0.74)
Axj_md(qj, ptau, i = 0.06, benefit = 1000)
```

Axn *Endowment insurance APV*

Description

Computes $A_{x:\overline{n}|} = A^1_{x:\overline{n}|} + {}_nE_x$.

Usage

```
Axn(x, n, i, tbl = NULL, model = NULL, ...)
```

Arguments

`x`	Age.
`n`	Term.
`i`	Effective annual interest rate.
`tbl`	Optional life table object.
`model`	Optional parametric survival model name.
`...`	Additional arguments passed to survival-model functions.

Value

Numeric vector of APVs.

Axn1 *Term insurance APV*

Description

Computes $A^1_{x:\overline{n}|} = \sum_{k=0}^{n-1} v^{k+1}{}_{k|}q_x$.

Usage

```
Axn1(x, n, i, tbl = NULL, model = NULL, ...)
```

Arguments

x	Age.
n	Term.
i	Effective annual interest rate.
tbl	Optional life table object.
model	Optional parametric survival model name.
...	Additional arguments passed to survival-model functions.

Value

Numeric vector of APVs.

Axn1_m *m-thly term insurance APV*

Description

Computes $A^{1(m)}_{x:\overline{n}|} = \sum_{j=0}^{mn-1} v^{(j+1)/m} \Pr(j/m < T_x \le (j+1)/m)$.

Usage

```
Axn1_m(x, n, i, m, model, ...)
```

Arguments

x	Age.
n	Term.
i	Effective annual interest rate.
m	Positive integer payment frequency.
model	Parametric survival model name.
...	Additional model parameters passed to survival-model functions.

Value

Numeric vector of APVs.

`Axn1_m_udd` *UDD approximation of m-thly term insurance*

Description

Computes $A^{1(m)}_{x:\overline{n}|} = (i/i^{(m)})A^{1}_{x:\overline{n}|}$.

Usage

```
Axn1_m_udd(Axn1, i, m)
```

Arguments

`Axn1`	Discrete term insurance APV.
`i`	Effective annual interest rate.
`m`	Positive integer payment frequency.

Value

m-thly term insurance APV under UDD.

`Axn_m` *m-thly endowment insurance APV*

Description

Computes $A^{(m)}_{x:\overline{n}|} = A^{1(m)}_{x:\overline{n}|} + v^n {}_np_x$.

Usage

```
Axn_m(x, n, i, m, model, ...)
```

Arguments

`x`	Age.
`n`	Term.
`i`	Effective annual interest rate.
`m`	Positive integer payment frequency.
`model`	Parametric survival model name.
`...`	Additional model parameters passed to survival-model functions.

Value

Numeric vector of APVs.

Axn_m_udd *UDD approximation of m-thly endowment insurance*

Description

Computes $A^{(m)}_{x:\overline{n}|} = (i/i^{(m)})A^{1}_{x:\overline{n}|} + {}_nE_x$.

Usage

```
Axn_m_udd(Axn1, nEx, i, m)
```

Arguments

Axn1	Discrete term insurance APV.
nEx	Pure endowment APV.
i	Effective annual interest rate.
m	Positive integer payment frequency.

Value

m-thly endowment insurance APV under UDD.

Ax_m *m-thly whole life insurance APV*

Description

Computes $A^{(m)}_x = \sum_{j=0}^{\infty} v^{(j+1)/m} \Pr(j/m < T_x \le (j+1)/m)$.

Usage

```
Ax_m(x, i, m, model, ..., tol = 1e-12, j_max = 100000L)
```

Arguments

x	Age.
i	Effective annual interest rate.
m	Positive integer payment frequency.
model	Parametric survival model name.
...	Additional model parameters passed to survival-model functions.
tol	Numerical tolerance for truncating the infinite sum.
j_max	Maximum number of m-thly intervals in the sum.

Value

Numeric vector of APVs.

Ax_m_udd *UDD approximation of m-thly whole life insurance*

Description

Computes $A_x^{(m)} = (i/i^{(m)})A_x$.

Usage

```
Ax_m_udd(Ax, i, m)
```

Arguments

Ax	Discrete whole life insurance APV.
i	Effective annual interest rate.
m	Positive integer payment frequency.

Value

m-thly whole life insurance APV under UDD.

coi_ul_typeB *Cost of insurance for Type B universal life*

Description

Computes the one-period cost of insurance for a Type B universal life contract:

$$\mathrm{COI}_t = \frac{B_t q_{x+t-1}}{1 + i_t^q}.$$

Usage

```
coi_ul_typeB(B, qx, iq)
```

Arguments

B	Face amount.
qx	Mortality probability for the period.
iq	Interest rate used in the cost-of-insurance calculation. Values must be greater than -1.

Details

The arguments may be scalars or vectors. Scalar arguments are recycled to the common length.

Value

A numeric vector of cost-of-insurance charges.

Examples

```
coi_ul_typeB(B = 100000, qx = 0.00076, iq = 0.03)
coi_ul_typeB(
  B = 100000,
  qx = c(0.00076, 0.00081),
  iq = c(0.03, 0.035)
)
```

---

continuous_multilife_annuities

Continuous multi-life annuities

Description

Computes continuous joint-life, last-survivor, and reversionary whole-life annuities for two independent lives.

Usage

```
abarxy(x, y, i, tbl = NULL, model = NULL, ...)

abarxybar(x, y, i, tbl = NULL, model = NULL, ...)

abarx_y(x, y, i, tbl = NULL, model = NULL, ...)

abary_x(x, y, i, tbl = NULL, model = NULL, ...)
```

Arguments

`x`	Age of the first life. May be scalar or vector.
`y`	Age of the second life. May be scalar or vector.
`i`	Effective annual interest rate. May be scalar or vector.
`tbl`	Optional life table object retained for backward compatibility. Continuous calculations currently require `model`.
`model`	Parametric survival model.
`...`	Additional parameters passed to the survival functions.

Details

`abarxy()` computes the joint-life annuity.

`abarxybar()` computes the last-survivor annuity.

`abarx_y()` computes the reversionary annuity payable to the second life after the death of the first.

`abary_x()` computes the reversionary annuity payable to the first life after the death of the second.

These functions require a parametric survival model.

Under independence, `abarxy()` represents the continuous joint-life annuity, payable while both lives survive.

The last-survivor annuity satisfies

$$\bar{a}_{\overline{xy}} = \bar{a}_x + \bar{a}_y - \bar{a}_{xy}.$$

The reversionary annuities are obtained as the difference between the corresponding single-life annuity and the joint-life annuity.

Value

A numeric vector of actuarial present values.

Examples

```
abarxy(
  x = 40,
  y = 50,
  i = 0.05,
  model = "uniform",
  omega = 100
)

abarxybar(
  x = 40,
  y = 50,
  i = 0.05,
  model = "uniform",
  omega = 100
)
```

---

continuous_multilife_insurance

Continuous multi-life insurance

Description

Computes continuous joint-life, last-survivor, and contingent whole-life insurance values for two independent lives.

Usage

```
Abarxy(x, y, i, tbl = NULL, model = NULL, ...)

Abarxybar(x, y, i, tbl = NULL, model = NULL, ...)

Abarxy1(x, y, i, tbl = NULL, model = NULL, ...)

Abaryx1(x, y, i, tbl = NULL, model = NULL, ...)

Abarxy2(x, y, i, tbl = NULL, model = NULL, ...)

Abaryx2(x, y, i, tbl = NULL, model = NULL, ...)
```

Arguments

x	Age of the first life. May be scalar or vector.
y	Age of the second life. May be scalar or vector.
i	Effective annual interest rate. May be scalar or vector.
tbl	Optional life table object retained for backward compatibility. Continuous calculations currently require model.
model	Parametric survival model.
...	Additional parameters passed to the survival and hazard functions.

Details

Abarxy() computes joint-life insurance payable at the first death.

Abarxybar() computes last-survivor insurance payable at the second death.

Abarxy1() and Abaryx1() compute contingent insurance payable when the specified life dies first.

Abarxy2() and Abaryx2() compute contingent insurance payable when the specified life dies second.

These functions require a parametric survival model.

Under independence, Abarxy() represents insurance payable at the first death and Abarxybar() represents insurance payable at the second death.

The contingent functions distinguish both the life whose death triggers payment and whether that death occurs first or second.

Value

A numeric vector of actuarial present values.

Examples

```
Abarxy(
  x = 40,
  y = 50,
  i = 0.05,
```

```
  model = "uniform",
  omega = 100
)

Abarxy1(
  x = 40,
  y = 50,
  i = 0.05,
  model = "uniform",
  omega = 100
)
```

`contribution_rate_target` *Target contribution rate for a defined contribution plan*

Description

Calculates the contribution rate required to achieve a target replacement ratio.

Usage

```
contribution_rate_target(x, z, Sx, RR_target, i, adue_z, g = NULL, s = NULL)
```

Arguments

`x`	Scalar entry age.
`z`	Scalar retirement age.
`Sx`	Positive scalar salary at age `x`.
`RR_target`	Scalar target replacement ratio in `[0, 1]`.
`i`	Scalar annual effective investment return greater than `-1`.
`adue_z`	Positive scalar whole-life annuity-due factor at retirement.
`g`	Optional scalar annual salary growth rate greater than `-1`.
`s`	Optional positive salary-scale vector of length `z - x`.

Value

A numeric scalar. The result may exceed one when the target cannot be achieved with a contribution rate no greater than 100 percent.

Examples

```
contribution_rate_target(
  x = 30,
  z = 65,
  Sx = 60000,
  RR_target = 0.50,
  i = 0.06,
  adue_z = 11,
  g = 0.04
)
```

`cov_term_deferred` *Covariance of term and deferred insurance PVs*

Description

Computes $\mathrm{Cov}(Z^1_{x:\overline{n}|}, {}_{n|}Z_x) = -A^1_{x:\overline{n}|} \cdot {}_{n|}A_x$.

Usage

```
cov_term_deferred(x, n, i, tbl = NULL, model = NULL, ...)
```

Arguments

`x`	Age.
`n`	Term.
`i`	Effective annual interest rate.
`tbl`	Optional life table object.
`model`	Optional parametric survival model name.
`...`	Additional arguments passed to survival-model functions.

Value

Numeric vector of covariances.

cov_term_endow *Covariance of term insurance and pure endowment PVs*

Description

Computes $\mathrm{Cov}(Z^{1}_{x:\overline{n}|}, Z^{1(\text{pure endow})}_{x:\overline{n}|}) = -A^{1}_{x:\overline{n}|} \cdot {}_nE_x$.

Usage

```
cov_term_endow(x, n, i, tbl = NULL, model = NULL, ...)
```

Arguments

x	Age.
n	Term.
i	Effective annual interest rate.
tbl	Optional life table object.
model	Optional parametric survival model name.
...	Additional arguments passed to survival-model functions.

Value

Numeric vector of covariances.

cumhaz0 *Cumulative hazard for age-at-failure*

Description

Computes $\Lambda_0(t)$.

Usage

```
cumhaz0(t, model, ...)
```

Arguments

t	Numeric vector of times.
model	One of "uniform", "exponential", "gompertz", "makeham", or "weibull".
...	Model parameters.

Value

Numeric vector.

DAbarxn1 *Piecewise-continuous decreasing n-year term insurance*

Description

Computes

$$(D\bar{A})^{1}_{x:\overline{n}|} = \int_0^n \lfloor n+1-t \rfloor v^t {}_tp_x \mu_{x+t}\, dt.$$

Usage

```
DAbarxn1(x, n, i, model, ...)
```

Arguments

x	Age.
n	Term.
i	Effective annual interest rate.
model	Parametric survival model.
...	Additional model parameters.

Value

Numeric vector.

DAxn1 *Decreasing n-year term insurance*

Description

Computes

$$(DA)^{1}_{x:\overline{n}|} = \sum_{k=0}^{n-1} (n-k) v^{k+1} \Pr(K_x = k).$$

Usage

```
DAxn1(x, n, i, tbl = NULL, model = NULL, ...)
```

Arguments

x	Age.
n	Term.
i	Effective annual interest rate.
tbl	Optional life table object.
model	Optional parametric survival model.
...	Additional model parameters.

Value

Numeric vector.

`DbarAbarxn1` *Fully continuous decreasing n-year term insurance*

Description

Computes

$$(\bar{D}\bar{A})^{1}_{x:\overline{n}|} = \int_0^n (n-t) v^t {}_t p_x \mu_{x+t}\, dt.$$

Usage

```
DbarAbarxn1(x, n, i, model, ...)
```

Arguments

`x`	Age.
`n`	Term.
`i`	Effective annual interest rate.
`model`	Parametric survival model.
`...`	Additional model parameters.

Value

Numeric vector.

`decompGg_disc` *Ordered decomposition of gross gain*

Description

Decomposes gross gain into interest, mortality, and expense components in a user-specified order.

Usage

```
decompGg_disc(
  VtG,
  Vt1G,
  G,
  i_assumed,
  q_assumed,
  r_assumed = 0,
  e_assumed = 0,
  s_assumed = 0,
  i_actual,
  q_actual,
  r_actual = 0,
  e_actual = 0,
  s_actual = 0,
  b = 1,
  order = c("interest", "mortality", "expense")
)
```

Arguments

`VtG`	Gross reserve at duration `t`.
`Vt1G`	Gross reserve at duration `t + 1`.
`G`	Gross premium.
`i_assumed`	Assumed annual effective interest rate.
`q_assumed`	Assumed mortality probability.
`r_assumed`	Assumed percent-of-premium expense rate.
`e_assumed`	Assumed per-policy expense.
`s_assumed`	Assumed settlement expense.
`i_actual`	Actual annual effective interest rate.
`q_actual`	Actual mortality probability.
`r_actual`	Actual percent-of-premium expense rate.
`e_actual`	Actual per-policy expense.
`s_actual`	Actual settlement expense.
`b`	Benefit amount.
`order`	Character vector containing `"interest"`, `"mortality"`, and `"expense"` exactly once.

Details

For scalar input, the function returns a named numeric vector. For vectorized input, it returns a numeric matrix with one row per calculation.

Value

For scalar input, a named numeric vector. For vectorized input, a numeric matrix with columns `total_gain`, `interest`, `mortality`, `expense`, and `check`.

Examples

```
decompGg_disc(
  VtG = 3950.73,
  Vt1G = 4607.07,
  G = 685,
  i_assumed = 0.06,
  q_assumed = 0.00592,
  r_assumed = 0.05,
  e_assumed = 0,
  s_assumed = 300,
  i_actual = 0.065,
  q_actual = 0.005,
  r_actual = 0.06,
  e_actual = 0,
  s_actual = 100,
  b = 50000
)
```

---

deferred_insurance_reserves

Deferred insurance reserves

Description

Computes prospective reserves for deferred whole life insurance contracts.

Usage

```
tVnAx(x, n, t, i, model = NULL, ..., tbl = NULL)

htVnAx(x, n, h, t, i, model = NULL, ..., tbl = NULL)
```

Arguments

`x`	Issue age.
`n`	Deferral period in years.
`t`	Duration in years.
`i`	Effective annual interest rate.
`model`	Optional parametric survival model name.
`...`	Additional arguments passed to the underlying actuarial functions.
`tbl`	Optional life table object.
`h`	Premium-paying period in years.

Details

`tVnAx()` computes the reserve at duration `t` for an `n`-year deferred whole life insurance funded by level annual premiums during the deferral period.

`htVnAx()` computes the corresponding reserve when premiums are limited to the first `h` years, where `h <= n`.

Value

A numeric vector of prospective reserve values.

Examples

```
tVnAx(
  x = 40, n = 20, t = 10, i = 0.05,
  model = "uniform", omega = 100
)

htVnAx(
  x = 40, n = 20, h = 10, t = 5, i = 0.05,
  model = "uniform", omega = 100
)
```

---

discount *Discount factor for compound interest*

Description

Computes the discount factor $v^t = (1+i)^{-t}$.

Usage

```
discount(i, t)
```

Arguments

`i`	Effective interest rate. May be scalar or vector.
`t`	Time. May be scalar or vector.

Value

Numeric vector of discount factors.

Examples

```
discount(0.05, 0:5)
discount(c(0.03, 0.05), 1)
```

discounted_payback_period *Discounted payback period*

Description

Returns the first duration at which cumulative discounted profit is nonnegative.

Usage

```
discounted_payback_period(Pi, r)
```

Arguments

Pi	Numeric profit-signature vector.
r	Annual effective risk discount rate. May be scalar or vector; values must be greater than -1.

Value

An integer vector with one value for each discount rate. An element is NA_integer_ when payback is not reached.

Examples

```
Pi <- c(-15.00, 8.42, 8.39, 8.58)
discounted_payback_period(Pi, r = 0.10)
```

dist0 *Distribution functions for age-at-failure*

Description

Distribution functions for age-at-failure

Usage

```
F0(t, model, ...)

f0(t, model, ...)
```

Arguments

t	Numeric vector of times.
model	One of "uniform", "exponential", "gompertz", "makeham", or "weibull".
...	Model parameters.

Value

Numeric vector.

---

double_force_delta *Doubled force of interest*

Description

Computes $\delta' = 2\delta$.

Usage

```
double_force_delta(delta)
```

Arguments

delta	Numeric vector of forces of interest.

Value

Numeric vector of doubled forces of interest.

---

double_force_i *Effective annual interest at doubled force*

Description

If $\delta' = 2\delta$, then $i' = (1+i)^2 - 1$.

Usage

```
double_force_i(i)
```

Arguments

i	Numeric vector of effective annual interest rates.

Value

Numeric vector of effective annual rates corresponding to doubled force.

dx *Compute deaths between ages x and x+1*

Description

Compute deaths between ages x and x+1

Usage

```
dx(tbl, x)
```

Arguments

tbl	A life_table object.
x	Ages.

Value

Numeric vector of d_x values.

dxj *Cause-specific numbers of decrements*

Description

Computes

$$d_x^{(j)} = l_x^{(\tau)} q_x^{(j)}.$$

Usage

```
dxj(lxtau, qxj)
```

Arguments

lxtau	Number alive at age or duration x in the multiple-decrement table.
qxj	Numeric vector of cause-specific decrement probabilities.

Value

A numeric vector containing the number of decrements from each cause.

Examples

```
dxj(1000, c(0.011, 0.100))
```

dxtau *Total number of decrements*

Description

Computes

$$d_x^{(\tau)} = \sum_j d_x^{(j)}.$$

Usage

```
dxtau(lxtau, qxj)
```

Arguments

lxtau	Number alive at age or duration x in the multiple-decrement table.
qxj	Numeric vector of cause-specific decrement probabilities.

Value

A numeric scalar.

Examples

```
dxtau(1000, c(0.011, 0.100))
```

ELtx *Mean present value of loss at duration t for whole life insurance*

Description

Computes the conditional mean $E[{}_tL_x \mid K_x \geq t]$ for a fully discrete whole life insurance.

Usage

```
ELtx(x, t, i, P, model = NULL, ..., tbl = NULL)
```

Arguments

x	Issue age.
t	Duration.
i	Effective annual interest rate.
P	Annual premium.
model	Optional parametric survival model name.
...	Additional model parameters.
tbl	Optional life table object.

Details

Under the equivalence-principle premium, this equals the prospective reserve ${}_tV_x$.

Value

A numeric vector of values.

Examples

```
prem <- Px(40, i = 0.05, model = "uniform", omega = 100)
ELtx(40, t = 10, i = 0.05, P = prem, model = "uniform", omega = 100)
```

---

ex_complete *Complete expectation of life*

Description

Computes $\mathring{e}_x = \int_0^\infty {}_tp_x \, dt$.

Usage

```
ex_complete(x, model, ..., tol = 1e-10)
```

Arguments

x	Numeric vector of ages.
model	One of "uniform", "exponential", "gompertz", "makeham", or "weibull".
...	Model parameters.
tol	Tolerance used to choose a finite integration bound.

Value

Numeric vector of complete expectations.

ex_complete_tab *Complete expectation of life from a life table*

Description

Computes $\mathring{e}_x = \int_0^\infty {}_tp_x \, dt$ using a within-year assumption: `"udd"`, `"cf"`, or `"balducci"`.

Usage

```
ex_complete_tab(tbl, x, assumption = c("udd", "cf", "balducci"))
```

Arguments

`tbl`	A `life_table` object.
`x`	Numeric vector of integer ages.
`assumption`	One of `"udd"`, `"cf"`, `"balducci"`.

Value

Numeric vector of complete expectations $\mathring{e}_x$.

ex_curtate *Curtate expectation of life*

Description

Computes $e_x = E[K_x] = \sum_{k=1}^\infty {}_kp_x$.

Usage

```
ex_curtate(x, model, ..., k_max = 5000, tol = 1e-12)
```

Arguments

`x`	Numeric vector of ages.
`model`	One of `"uniform"`, `"exponential"`, `"gompertz"`, `"makeham"`, or `"weibull"`.
`...`	Model parameters.
`k_max`	Maximum integer duration to sum to.
`tol`	Stop early if the summand is smaller than this tolerance for several consecutive steps.

Value

Numeric vector of curtate expectations.

ex_curtate_tab *Curtate expectation of life from a life table*

Description

Computes the curtate expectation of life $e_x = \sum_{k=1}^{\infty} {}_kp_x$ in the discrete tabular setting.

Usage

```
ex_curtate_tab(tbl, x)
```

Arguments

tbl	A life_table object.
x	Numeric vector of integer ages.

Value

Numeric vector of curtate expectations e_x.

ex_temp_complete_tab *Temporary complete expectation of life from a life table*

Description

Computes $\mathring{e}_{x:\overline{n}|} = \int_0^n {}_tp_x \, dt$ using a within-year assumption: "udd", "cf", or "balducci".

Usage

```
ex_temp_complete_tab(tbl, x, n, assumption = c("udd", "cf", "balducci"))
```

Arguments

tbl	A life_table object.
x	Numeric vector of integer ages.
n	Numeric vector of nonnegative durations.
assumption	One of "udd", "cf", "balducci".

Value

Numeric vector of temporary complete expectations.

ex_temp_curtate_tab *Temporary curtate expectation of life from a life table*

Description

Computes $e_{x:\overline{n}|} = \sum_{k=1}^{n} {}_kp_x$ for integer n in the discrete tabular setting.

Usage

```
ex_temp_curtate_tab(tbl, x, n)
```

Arguments

tbl	A life_table object.
x	Numeric vector of integer ages.
n	Numeric vector of nonnegative integers.

Value

Numeric vector of temporary curtate expectations.

fnk_from_z *Forward rate implied by spot rates*

Description

Computes the annual effective forward rate for the interval from time n to time n + k:

$$(1 + z_{n+k})^{n+k} = (1 + z_n)^n (1 + f_{n,k})^k.$$

Usage

```
fnk_from_z(z, n, k)
```

Arguments

z	Numeric vector of annual effective spot rates for maturities 1, ..., length(z). Each value must be greater than -1.
n	Nonnegative integer forward-start time.
k	Positive integer forward period.

Value

A numeric scalar.

Examples

```
z <- c(0.03, 0.04, 0.05, 0.06, 0.07)
fnk_from_z(z, n = 1, k = 4)
fnk_from_z(z, n = 2, k = 2)
```

---

forward_matrix_from_z *Matrix of forward rates implied by spot rates*

Description

Constructs a matrix of annual effective forward rates. Rows correspond to forward-start times $n = 1, \ldots, m-1$; columns correspond to forward periods $k = 1, \ldots, m-1$. Entries requiring maturities beyond m are returned as NA.

Usage

```
forward_matrix_from_z(z)
```

Arguments

z	Numeric vector of at least two annual effective spot rates. Each value must be greater than -1.

Value

A numeric matrix.

Examples

```
forward_matrix_from_z(c(0.03, 0.04, 0.05, 0.06, 0.07))
```

---

fractional_duration_reserves *Fractional-duration whole life reserves*

Description

Computes fractional-duration whole life reserves using linear interpolation between the reserve immediately after the premium at duration t and the reserve at duration $t + 1$.

Usage

```
tsVx(x, t, s, i, tbl = NULL, model = NULL, ...)

meanVx(x, t, i, tbl = NULL, model = NULL, ...)
```

Arguments

x	Issue age. May be scalar or vector.
t	Nonnegative integer duration. May be scalar or vector.
s	Fractional duration in $[0, 1]$. May be scalar or vector.
i	Effective annual interest rate. May be scalar or vector.
tbl	Optional life table object.
model	Optional parametric survival model.
...	Additional parameters passed to the actuarial functions.

Details

`tsVx()` computes the reserve at fractional duration $t + s$, where $0 \leq s \leq 1$.

`meanVx()` computes the reserve at the midpoint of the policy year ($s = 0.5$).

The fractional reserve is computed using

$$ {}_{t+s}V_x = ({}_tV_x + P_x)(1 - s) + {}_{t+1}V_x s. $$

The function `meanVx()` is a convenience wrapper corresponding to $s = 0.5$.

Value

A numeric vector of reserve values.

Examples

```
tsVx(
  40,
  t = 10,
  s = 0.5,
  i = 0.05,
  model = "uniform",
  omega = 100
)

meanVx(
  40,
  t = 10,
  i = 0.05,
  model = "uniform",
  omega = 100
)
```

`fractional_duration_term_endowment_reserves` *Fractional-duration term and endowment reserves*

Description

Computes fractional-duration reserves for term and endowment insurance using linear interpolation between the reserve immediately after the premium at duration t and the reserve at duration $t + 1$.

Usage

```
tsVxn(x, n, t, s, i, tbl = NULL, model = NULL, ...)

tsVxn1(x, n, t, s, i, tbl = NULL, model = NULL, ...)
```

Arguments

`x`	Issue age. May be scalar or vector.
`n`	Positive integer term. May be scalar or vector.
`t`	Nonnegative integer duration satisfying $t < n$. May be scalar or vector.
`s`	Fractional duration in $[0, 1]$. May be scalar or vector.
`i`	Effective annual interest rate. May be scalar or vector.
`tbl`	Optional life table object.
`model`	Optional parametric survival model.
`...`	Additional parameters passed to the actuarial functions.

Details

`tsVxn()` computes the fractional-duration reserve for endowment insurance.

`tsVxn1()` computes the fractional-duration reserve for term insurance.

The fractional reserve is computed using

$$ {}_{t+s}V = ({}_tV + P)(1 - s) + {}_{t+1}Vs. $$

The premium P is the net annual premium for the corresponding insurance contract (term or endowment).

Value

A numeric vector of reserve values.

Examples

```
tsVxn(
  40,
  n = 20,
  t = 10,
  s = 0.5,
  i = 0.05,
  model = "uniform",
  omega = 100
)

tsVxn1(
  40,
  n = 20,
  t = 10,
  s = 0.5,
  i = 0.05,
  model = "uniform",
  omega = 100
)
```

---

full_preliminary_term *Full preliminary term modified premiums and reserves*

Description

Computes modified premiums and reserves under the full preliminary term method for whole-life insurance.

Usage

```
alphaF(x, i, tbl = NULL, model = NULL, ...)

betaF(x, i, tbl = NULL, model = NULL, ...)

tVFx(x, t, i, tbl = NULL, model = NULL, ...)
```

Arguments

x	Issue age. May be scalar or vector.
i	Effective annual interest rate. May be scalar or vector.
tbl	Optional life table object.
model	Optional parametric survival model.
...	Additional parameters passed to the actuarial functions.
t	Nonnegative integer duration. May be scalar or vector.

Details

`alphaF()` computes the first-year modified premium $\alpha^F = vq_x$.

`betaF()` computes the renewal modified premium $\beta^F = P_{x+1}$.

`tVFx()` computes the full preliminary term reserve. The reserve is zero at durations 0 and 1. For $t > 1$, it equals the net level premium reserve at duration $t - 1$ for a policy issued at age $x + 1$.

Under the full preliminary term method, the first policy year is treated as one-year term insurance. The first-year modified premium is therefore the actuarial present value of one-year term insurance at age x.

Renewal premiums are based on a whole-life policy issued one year later, at age $x+1$. Accordingly, reserves after the first policy year are obtained from the corresponding net level premium reserve for that deferred issue age.

Value

A numeric vector of modified premiums or reserves.

Examples

```
alphaF(
  x = 40,
  i = 0.05,
  model = "uniform",
  omega = 100
)

betaF(
  x = 40,
  i = 0.05,
  model = "uniform",
  omega = 100
)

tVFx(
  x = 40,
  t = 5,
  i = 0.05,
  model = "uniform",
  omega = 100
)
```

---

fx *Conditional density*

Description

Computes $f_x(t) = f_0(x + t)/S_0(x)$.

Usage

```
fx(t, x, model, ...)
```

Arguments

t	Numeric vector of durations.
x	Numeric vector of ages.
model	One of "uniform", "exponential", "gompertz", "makeham", or "weibull".
...	Model parameters.

Value

Numeric vector of density values.

---

fx_tab *Fractional conditional density from a life table*

Description

Computes the conditional density $f_x(t \mid T_0 > x) = {}_tp_x \mu_{x+t}$.

Usage

```
fx_tab(tbl, x, t, assumption = c("udd", "cf", "balducci"))
```

Arguments

tbl	A life_table object.
x	Numeric vector of integer ages.
t	Numeric vector of fractional durations with $0 \leq t \leq 1$.
assumption	One of "udd", "cf", "balducci".

Value

Numeric vector of conditional density values.

`gain_loss_md` *Gain or loss in a two-cause multiple-decrement model*

Description

Computes one-year gain or loss under simultaneous within-year decrement probabilities or an ordered case in which Cause 2 occurs at year-end. Numeric arguments may be scalar or compatible vectors.

Usage

```
gain_loss_md(
  Vt,
  G,
  r,
  e,
  i,
  b1,
  b2,
  s1 = 0,
  s2 = 0,
  q1,
  q2,
  Vt1,
  year_end_cause2 = FALSE,
  q1prime = NULL,
  q2prime = NULL
)
```

Arguments

`Vt`	Gross reserve at the beginning of the year.
`G`	Gross premium.
`r`	Percent-of-premium expense rate.
`e`	Fixed beginning-of-year expense.
`i`	Earned effective annual interest rate.
`b1`	Cause 1 benefit.
`b2`	Cause 2 benefit.
`s1`	Cause 1 settlement expense.
`s2`	Cause 2 settlement expense.
`q1`	Cause 1 decrement probability.
`q2`	Cause 2 decrement probability.
`Vt1`	Gross reserve at the end of the year.

year_end_cause2	Whether Cause 2 occurs only at year-end.
q1prime	Single-decrement Cause 1 probability for the ordered case.
q2prime	Single-decrement Cause 2 probability for the ordered case.

Value

A numeric vector of gains or losses.

Examples

```
gain_loss_md(
  Vt = 115, G = 16, r = 0, e = 3, i = 0.06,
  b1 = 1000, b2 = 110, q1 = 0.01, q2 = 0.10, Vt1 = 128.83
)
```

GI_cont *Interest gain for a continuous-style recursion*

Description

Evaluates the one-step gain using the actual force of interest and the assumed survival probability.

Usage

```
GI_cont(Vt, Vt1, P, delta_actual, p_assumed, benefit = 0, h = 1)
```

Arguments

Vt	Reserve at time ‘t‘.
Vt1	Reserve at time ‘t + h‘.
P	Premium rate.
delta_actual	Actual force of interest.
p_assumed	Assumed survival probability over the step.
benefit	Benefit paid at the start of the step.
h	Positive step length.

Value

Numeric vector of interest gain values.

Examples

```
GI_cont(
  Vt = 10,
  Vt1 = 11,
  P = 1,
  delta_actual = 0.05,
  p_assumed = 0.99
)
```

`GI_disc` *Interest gain for a discrete insurance contract*

Description

Interest gain for a discrete insurance contract

Usage

```
GI_disc(Vt, Vt1, P, i_actual, q_assumed, B = 1)
```

Arguments

`Vt`	Reserve at duration t.
`Vt1`	Reserve at duration t+1.
`P`	Net premium for the year.
`i_actual`	Actual annual effective interest rate.
`q_assumed`	Assumed mortality rate for the year.
`B`	Benefit amount. Defaults to 1.

Value

A numeric vector of values.

Examples

```
GI_disc(Vt = 0.1, Vt1 = 0.11, P = 0.02, i_actual = 0.05, q_assumed = 0.01)
```

GMF_rollforward_ul *Guaranteed maturity fund roll-forward*

Description

Computes a one-period guaranteed maturity fund roll-forward. Arguments may be scalars or vectors and follow common-length recycling.

Usage

```
GMF_rollforward_ul(GMF_prev, GMP, r, policy_charge, i)
```

Arguments

GMF_prev	Prior guaranteed maturity fund.
GMP	Guaranteed maturity premium.
r	Percent-of-premium expense rate in [0, 1].
policy_charge	Guaranteed policy charge.
i	Guaranteed annual effective interest rate greater than -1.

Value

A numeric vector.

Examples

```
GMF_rollforward_ul(140.40, 14.49, 0.04, 11.80, 0.03)
```

GM_cont *Mortality gain for a continuous-style recursion*

Description

Evaluates the one-step gain using the assumed force of interest and the actual survival probability.

Usage

```
GM_cont(Vt, Vt1, P, delta_assumed, p_actual, benefit = 0, h = 1)
```

Arguments

Vt	Reserve at time ‘t‘.
Vt1	Reserve at time ‘t + h‘.
P	Premium rate.
delta_assumed	Assumed force of interest.
p_actual	Actual survival probability over the step.
benefit	Benefit paid at the start of the step.
h	Positive step length.

Value

Numeric vector of mortality gain values.

Examples

```
GM_cont(
  Vt = 10,
  Vt1 = 11,
  P = 1,
  delta_assumed = 0.05,
  p_actual = 0.99
)
```

GM_disc *Mortality gain for a discrete insurance contract*

Description

Mortality gain for a discrete insurance contract

Usage

```
GM_disc(Vt, Vt1, P, i_assumed, q_actual, B = 1)
```

Arguments

Vt	Reserve at duration t.
Vt1	Reserve at duration t+1.
P	Net premium for the year.
i_assumed	Assumed annual effective interest rate.
q_actual	Actual mortality rate for the year.
B	Benefit amount. Defaults to 1.

Value

A numeric vector of values.

Examples

```
GM_disc(Vt = 0.1, Vt1 = 0.11, P = 0.02, i_assumed = 0.04, q_actual = 0.01)
```

---

gross_premium_expense_reserves

Whole life gross premium and expense reserves

Description

Computes prospective gross premium and expense reserves for fully discrete whole life insurance after issue.

Usage

```
tVGx(
  x,
  t,
  i,
  G,
  benefit = 1,
  renewal_premium_pct = 0,
  renewal_policy_exp = 0,
  settlement_exp = 0,
  tbl = NULL,
  model = NULL,
  ...
)

tVEx(
  x,
  t,
  i,
  G,
  benefit = 1,
  renewal_premium_pct = 0,
  renewal_policy_exp = 0,
  settlement_exp = 0,
  tbl = NULL,
  model = NULL,
  ...
)
```

Arguments

`x`	Issue age. May be scalar or vector.
`t`	Nonnegative integer duration. May be scalar or vector.
`i`	Effective annual interest rate. May be scalar or vector.
`G`	Gross annual premium. May be scalar or vector.
`benefit`	Insurance benefit amount.
`renewal_premium_pct`	Renewal percent-of-premium expense in $[0, 1]$.
`renewal_policy_exp`	Renewal per-policy expense.
`settlement_exp`	Settlement expense paid at death.
`tbl`	Optional life table object.
`model`	Optional parametric survival model.
`...`	Additional parameters passed to the actuarial functions.

Details

`tVGx()` computes the prospective gross premium reserve.

`tVEx()` computes the corresponding expense reserve, defined as the difference between the gross premium reserve and the net benefit reserve.

The gross premium reserve is calculated as

$$ {}_tV_x^G = (b + s)A_{x+t} - [(1 - r)G - e]\,\ddot{a}_{x+t}, $$

where

- b is the insurance benefit,
- s is the settlement expense,
- G is the gross annual premium,
- r is the renewal percent-of-premium expense, and
- e is the renewal per-policy expense.

The expense reserve is obtained as the gross premium reserve minus the corresponding net benefit reserve.

Value

A numeric vector of reserve values.

Examples

```
tVGx(
  x = 40,
  t = 10,
  i = 0.05,
  G = 0.03,
  benefit = 1,
  renewal_premium_pct = 0.10,
  renewal_policy_exp = 0.002,
  settlement_exp = 0.02,
  model = "uniform",
  omega = 100
)

tVEx(
  x = 40,
  t = 10,
  i = 0.05,
  G = 0.03,
  benefit = 1,
  renewal_premium_pct = 0.10,
  renewal_policy_exp = 0.002,
  settlement_exp = 0.02,
  model = "uniform",
  omega = 100
)
```

GTg_disc *Total gross gain for a discrete insurance contract*

Description

Computes total gain during one policy year under gross premiums, gross reserves, actual mortality, actual interest, and actual expenses.

Usage

```
GTg_disc(
  VtG,
  Vt1G,
  G,
  i_actual,
  q_actual,
  r_actual = 0,
  e_actual = 0,
  s_actual = 0,
  b = 1
)
```

Arguments

VtG	Gross reserve at duration t.
Vt1G	Gross reserve at duration t + 1.
G	Gross premium.
i_actual	Actual annual effective interest rate.
q_actual	Actual mortality probability.
r_actual	Actual percent-of-premium expense rate.
e_actual	Actual per-policy expense.
s_actual	Actual settlement expense.
b	Benefit amount.

Value

A numeric vector of total gains.

Examples

```
GTg_disc(
  VtG = 0.10,
  Vt1G = 0.12,
  G = 0.02,
  i_actual = 0.05,
  q_actual = 0.01,
  r_actual = 0.03,
  e_actual = 0,
  s_actual = 0.01,
  b = 1
)
```

GT_cont *Total gain for a continuous-style one-step recursion*

Description

Computes the amount accumulated during a step, less the expected reserve required at the end of the step.

Usage

```
GT_cont(Vt, Vt1, P, delta_actual, p_actual, benefit = 0, h = 1)
```

Arguments

Vt	Reserve at time ‘t‘.
Vt1	Reserve at time ‘t + h‘.
P	Premium rate.
delta_actual	Actual force of interest.
p_actual	Actual survival probability over the step.
benefit	Benefit paid at the start of the step.
h	Positive step length.

Details

Reserves, premium rates, benefits, and forces of interest may be negative when such values are meaningful for the application. The step length must be positive, and the survival probability must lie in $[0, 1]$.

Value

Numeric vector of gain values.

Examples

```
GT_cont(
  Vt = 10,
  Vt1 = 11,
  P = 1,
  delta_actual = 0.05,
  p_actual = 0.99
)
```

GT_disc *Total gain for a discrete insurance contract*

Description

Computes the total gain: amount on hand at year-end minus amount required.

Usage

```
GT_disc(Vt, Vt1, P, i_actual, q_actual, B = 1)
```

Arguments

Vt	Reserve at duration t.
Vt1	Reserve at duration t+1.
P	Net premium for the year.
i_actual	Actual annual effective interest rate.
q_actual	Actual mortality rate for the year.
B	Benefit amount. Defaults to 1.

Value

A numeric vector of values.

Examples

```
GT_disc(Vt = 0.1, Vt1 = 0.11, P = 0.02, i_actual = 0.05, q_actual = 0.01)
```

---

hazard0 *Hazard or force of mortality for age-at-failure*

Description

Computes $\lambda_0(t)$.

Usage

```
hazard0(t, model, ...)
```

Arguments

t	Numeric vector of times.
model	One of "uniform", "exponential", "gompertz", "makeham", or "weibull".
...	Model parameters.

Value

Numeric vector.

---

htVx *h-pay whole life net level premium reserve*

Description

Computes the prospective reserve for an h-pay whole life policy.

Usage

```
htVx(x, h, t, i, model = NULL, ..., tbl = NULL)
```

Arguments

x	Issue age.
h	Premium-paying period in years.
t	Duration.
i	Effective annual interest rate.
model	Optional parametric survival model name.
...	Additional model parameters.
tbl	Optional life table object.

Value

A numeric vector of values.

Examples

```
htVx(40, h = 10, t = 5, i = 0.05, model = "uniform", omega = 100)
```

IAbarx *Piecewise-continuous increasing whole life insurance*

Description

Computes

$$(I\bar{A})_x = \int_0^\infty \lfloor t+1 \rfloor v^t {}_tp_x \mu_{x+t}\, dt.$$

Usage

```
IAbarx(x, i, model, ...)
```

Arguments

x	Age.
i	Effective annual interest rate.
model	Parametric survival model.
...	Additional model parameters.

Value

Numeric vector.

IAx *Increasing whole life insurance*

Description

Computes

$$(IA)_x = \sum_{k=0}^{\infty} (k+1) v^{k+1} \Pr(K_x = k).$$

Usage

```
IAx(x, i, tbl = NULL, model = NULL, ..., tol = 1e-12, k_max = 5000)
```

Arguments

x	Age.
i	Effective annual interest rate.
tbl	Optional life table object.
model	Optional parametric survival model.
...	Additional model parameters.
tol	Numerical tolerance for truncation.
k_max	Maximum number of terms.

Value

Numeric vector.

IAxn1 *Increasing n-year term insurance*

Description

Computes

$$(IA)^1_{x:\overline{n}|} = \sum_{k=0}^{n-1} (k+1) v^{k+1} \Pr(K_x = k).$$

Usage

```
IAxn1(x, n, i, tbl = NULL, model = NULL, ...)
```

Arguments

x	Age.
n	Term.
i	Effective annual interest rate.
tbl	Optional life table object.
model	Optional parametric survival model.
...	Additional model parameters.

Value

Numeric vector.

IbarAbarx *Fully continuous increasing whole life insurance*

Description

Computes

$$(\bar{I}\bar{A})_x = \int_0^\infty t v^t {}_tp_x \mu_{x+t}\, dt.$$

Usage

```
IbarAbarx(x, i, model, ...)
```

Arguments

x	Age.
i	Effective annual interest rate.
model	Parametric survival model.
...	Additional model parameters.

Value

Numeric vector.

IbarAbarxn1 *Fully continuous increasing n-year term insurance*

Description

Computes

$$(\bar{I}\bar{A})^{1}_{x:\overline{n}|} = \int_0^n tv^t {}_tp_x \mu_{x+t}\, dt.$$

Usage

```
IbarAbarxn1(x, n, i, model, ...)
```

Arguments

x	Age.
n	Term.
i	Effective annual interest rate.
model	Parametric survival model.
...	Additional model parameters.

Value

Numeric vector.

iMA_eiul *Monthly-average index growth rate*

Description

Computes the monthly-average growth rate from an initial index value and twelve monthly closing values.

Usage

```
iMA_eiul(index)
```

Arguments

index	Numeric vector of length 13 containing a strictly positive initial index value followed by twelve nonnegative monthly closing values.

Value

A numeric scalar.

Examples

```
index <- c(
  1000, 1020, 1100, 1150, 1080, 1040, 960,
  1030, 1000, 1070, 1150, 1200, 1150
)
iMA_eiul(index)
```

`Income_dc` *Retirement income from a defined contribution accumulation*

Description

Converts an accumulated account value into annual annuity-due income. Scalar arguments are recycled to a common length.

Usage

```
Income_dc(AVz, adue_z)
```

Arguments

`AVz`	Nonnegative accumulated value at retirement.
`adue_z`	Positive whole-life annuity-due factor at retirement.

Value

A numeric vector.

Examples

```
Income_dc(AVz = 824211.35, adue_z = 12)
```

`interest_convert` *Convert between compound-interest quantities*

Description

Provides consistent conversions between effective interest rate, effective discount rate, force of interest, and optional nominal interest rate convertible m-thly.

Usage

```
interest_convert(i = NULL, d = NULL, delta = NULL, m = NULL)
```

Arguments

i	Effective interest rate. May be scalar or vector.
d	Effective discount rate. May be scalar or vector.
delta	Force of interest. May be scalar or vector.
m	Optional positive integer compounding frequency for the nominal rate convertible m-thly.

Details

Exactly one of ‘i‘, ‘d‘, or ‘delta‘ must be provided.

Value

A list with elements ‘i‘, ‘d‘, ‘delta‘, and, if ‘m‘ is supplied, ‘im‘ and ‘m‘.

Examples

```
interest_convert(i = 0.05)
interest_convert(i = c(0.03, 0.05, 0.07))
interest_convert(d = 0.04761905)
interest_convert(delta = log(1.05))
```

---

iP_eiul *Point-to-point index growth rates*

Description

Computes consecutive point-to-point growth rates from index values.

Usage

```
iP_eiul(index)
```

Arguments

index	Numeric vector of strictly positive index values.

Value

A numeric vector with length one less than `index`.

Examples

```
iP_eiul(c(1000, 1050, 1200, 1100))
```

IRR_profit *Internal rate of return*

Description

Computes a root of the profit-signature net present value.

Usage

```
IRR_profit(Pi, interval = c(0, 1), tol = .Machine$double.eps^0.5)
```

Arguments

Pi	Numeric profit-signature vector.
interval	Numeric vector of length two giving the root-search interval. Its lower endpoint must be greater than -1.
tol	Positive scalar tolerance passed to stats::uniroot().

Value

A numeric scalar.

Examples

```
Pi <- c(-15.00, 8.42, 8.39, 8.58)
IRR_profit(Pi)
```

i_credit_eiul *Credited rates from index growth rates*

Description

Applies a participation rate, floor, cap, and optional margin to raw index growth rates.

Usage

```
i_credit_eiul(
  i_raw,
  part = 1,
  floor = 0,
  cap = Inf,
  margin = 0,
  margin_after_participation = TRUE
)
```

Arguments

i_raw	Numeric vector of raw index growth rates.
part	Nonnegative scalar participation rate.
floor	Scalar minimum credited rate.
cap	Scalar maximum credited rate. The default is Inf.
margin	Nonnegative scalar index margin.
margin_after_participation	Logical scalar. If TRUE, the margin is subtracted after applying participation; otherwise it is subtracted before applying participation.

Value

A numeric vector of credited rates.

Examples

```
raw <- iP_eiul(c(1000, 1050, 1200, 1100))
i_credit_eiul(raw, part = 1.10, floor = 0.01, cap = 0.10)
i_credit_eiul(raw)
```

---

joint_life_annuities *Joint-life annuities*

Description

Computes temporary and whole-life joint-life annuities-due and annuities-immediate for two independent lives.

Usage

```
adotxyn(x, y, n, i, tbl = NULL, model = NULL, ...)

axyn(x, y, n, i, tbl = NULL, model = NULL, ...)

adotxy(x, y, i, tbl = NULL, model = NULL, ..., k_max = 5000L, tol = 1e-12)

axy(x, y, i, tbl = NULL, model = NULL, ..., k_max = 5000L, tol = 1e-12)
```

Arguments

x	Age of the first life. May be scalar or vector.
y	Age of the second life. May be scalar or vector.
n	Nonnegative integer term. May be scalar or vector.
i	Effective annual interest rate. May be scalar or vector.

tbl	Optional life table object.
model	Optional parametric survival model.
...	Additional parameters passed to the survival model.
k_max	Maximum summation horizon used for non-terminating parametric survival models.
tol	Positive convergence tolerance for whole-life summations.

Details

adotxyn() computes an n-year temporary joint-life annuity-due.

axyn() computes an n-year temporary joint-life annuity-immediate.

adotxy() computes a whole-life joint-life annuity-due.

axy() computes a whole-life joint-life annuity-immediate.

All calculations assume independence between the two future lifetimes.

The temporary annuity-due is

$$\ddot{a}_{xy:\overline{n}|} = \sum_{k=0}^{n-1} v^k \, {}_kp_{xy}.$$

The temporary annuity-immediate is

$$a_{xy:\overline{n}|} = \sum_{k=1}^{n} v^k \, {}_kp_{xy}.$$

For life-table calculations, the whole-life sums terminate at the last duration supported jointly by the two lives. For parametric models, the sums are evaluated until convergence or until k_max is reached.

Value

A numeric vector of annuity actuarial present values.

Examples

```
adotxyn(
  x = 40,
  y = 50,
  n = 10,
  i = 0.05,
  model = "uniform",
  omega = 100
)

axyn(
  x = 40,
  y = 50,
  n = 10,
  i = 0.05,
```

```
  model = "uniform",
  omega = 100
)

adotxy(
  x = 40,
  y = 50,
  i = 0.05,
  model = "uniform",
  omega = 100
)

axy(
  x = 40,
  y = 50,
  i = 0.05,
  model = "uniform",
  omega = 100
)
```

`joint_life_insurance` *Joint-life insurance functions*

Description

Computes temporary and whole-life insurance values for two lives under the joint-life status.

Usage

```
Axyn1(x, y, n, i, tbl = NULL, model = NULL, ...)

Axyn(x, y, n, i, tbl = NULL, model = NULL, ...)

Axy(x, y, i, tbl = NULL, model = NULL, ..., k_max = 5000L, tol = 1e-12)
```

Arguments

`x`	Age of the first life. May be scalar or vector.
`y`	Age of the second life. May be scalar or vector.
`n`	Term in years. May be scalar or vector.
`i`	Effective annual interest rate. May be scalar or vector.
`tbl`	Optional life table object.
`model`	Optional parametric survival model.
`...`	Additional parameters passed to the survival model or life-table functions.
`k_max`	Maximum number of terms used for an infinite series.
`tol`	Numerical tolerance used to assess convergence.

Details

`Axyn1()` computes an n-year joint-life term insurance.

`Axyn()` computes an n-year joint-life endowment insurance.

`Axy()` computes joint-life whole-life insurance.

Value

A numeric vector of actuarial present values.

`last_survivor_annuities`

Last-survivor annuity functions

Description

Computes temporary and whole-life annuities for two lives under the last-survivor status.

Usage

```
adotxybarn(x, y, n, i, tbl = NULL, model = NULL, ...)

axybarn(x, y, n, i, tbl = NULL, model = NULL, ...)

adotxybar(x, y, i, tbl = NULL, model = NULL, ..., k_max = 5000L, tol = 1e-12)

axybar(x, y, i, tbl = NULL, model = NULL, ..., k_max = 5000L, tol = 1e-12)
```

Arguments

`x`	Age of the first life. May be scalar or vector.
`y`	Age of the second life. May be scalar or vector.
`n`	Term in years. May be scalar or vector.
`i`	Effective annual interest rate. May be scalar or vector.
`tbl`	Optional life table object.
`model`	Optional parametric survival model.
`...`	Additional parameters passed to the survival model or life-table functions.
`k_max`	Maximum number of terms used for an infinite series.
`tol`	Numerical tolerance used to assess convergence.

Value

A numeric vector of actuarial present values.

`last_survivor_insurance` *Last-survivor insurance functions*

Description

Computes temporary and whole-life insurance values for two lives under the last-survivor status.

Usage

```
Axybarn1(x, y, n, i, tbl = NULL, model = NULL, ...)

Axybarn(x, y, n, i, tbl = NULL, model = NULL, ...)

Axybar(x, y, i, tbl = NULL, model = NULL, ..., k_max = 5000L, tol = 1e-12)
```

Arguments

`x`	Age of the first life. May be scalar or vector.
`y`	Age of the second life. May be scalar or vector.
`n`	Term in years. May be scalar or vector.
`i`	Effective annual interest rate. May be scalar or vector.
`tbl`	Optional life table object.
`model`	Optional parametric survival model.
`...`	Additional parameters passed to the survival model or life-table functions.
`k_max`	Maximum number of terms used for an infinite series.
`tol`	Numerical tolerance used to assess convergence.

Details

`Axybarn1()` computes temporary insurance payable at the second death within the term.

`Axybarn()` computes last-survivor endowment insurance.

`Axybar()` computes last-survivor whole-life insurance.

Value

A numeric vector of actuarial present values.

life_table *Construct a life table*

Description

Build a discrete life table from one of `lx`, `qx`, `px`, or `S0`.

Usage

```
life_table(x, lx = NULL, qx = NULL, px = NULL, S0 = NULL, radix = 1e+05)
```

Arguments

`x`	Numeric vector of ages.
`lx`	Numeric vector of l_x values.
`qx`	Numeric vector of q_x values.
`px`	Numeric vector of p_x values.
`S0`	Numeric vector of $S_0(x)$ values.
`radix`	Radix used when converting `S0` to `lx`, or when building from `qx` or `px`.

Value

A data frame with class `"life_table"`.

lx *Extract life-table survivor values*

Description

Extract life-table survivor values

Usage

```
lx(tbl, x)
```

Arguments

`tbl`	A `life_table` object.
`x`	Ages.

Value

Numeric vector of l_x values.

lx_select *Extract select-table survivor value*

Description

Returns $l_{[x]+t}$ from a select life table.

Usage

```
lx_select(tbl, x_sel, t)
```

Arguments

tbl	A select_life_table object.
x_sel	Numeric vector of ages at selection.
t	Numeric vector of durations since selection.

Value

Numeric vector of survivor values.

lx_to_S0 *Convert life-table values to survival probabilities*

Description

Converts life-table survivor values l_x into survival probabilities $S_0(x) = l_x/l_0$.

Usage

```
lx_to_S0(lx)
```

Arguments

lx	Numeric vector of life-table survivor values.

Value

Numeric vector of $S_0(x)$ values.

markov_nstep_prob *Multi-step transition probability*

Description

Computes an entry of the matrix power P^n.

Usage

```
markov_nstep_prob(P, n, i, j)
```

Arguments

P	Square transition-probability matrix.
n	Nonnegative integer number of steps.
i	Starting-state index.
j	Ending-state index.

Value

A numeric scalar.

Examples

```
P <- matrix(c(0.9, 0.1, 0, 1), nrow = 2, byrow = TRUE)
markov_nstep_prob(P, n = 3, i = 1, j = 2)
```

md_table *Construct a multiple-decrement table*

Description

Constructs a discrete multiple-decrement table from cause-specific one-year decrement probabilities.

Usage

```
md_table(x, qxj, radix = 1e+05)
```

Arguments

x	Integer vector of consecutive ages or durations.
qxj	Numeric matrix or data frame of cause-specific decrement probabilities. Rows correspond to values in x, and columns correspond to causes.
radix	Initial value of $l_x^{(\tau)}$.

Value

An object of classes `"md_table"` and `"data.frame"` containing age or duration, cause-specific decrement probabilities, total decrement and survival probabilities, numbers alive, and cause-specific and total decrements.

Examples

```
ages <- 45:50
qmat <- cbind(
  withdrawal = c(0.011, 0.012, 0.013, 0.014, 0.015, 0.016),
  retirement = rep(0.100, 6)
)

md_table(ages, qmat, radix = 1000)
```

---

mortality_improvement_projection

Mortality improvement projection functions

Description

Functions for projecting one-year death and survival probabilities under mortality improvement and evaluating annuity values under projected mortality.

Usage

```
qx_proj(qx_base, AAx, base_year, proj_year)

px_proj(qx_base, AAx, base_year, proj_year)

tpx_improved(x0, n, qx_base_vec, AAx_vec, base_year, issue_year)

axn_improved(x0, n, i, qx_base_vec, AAx_vec, base_year, issue_year)

naxn_improved(x0, u, n, i, qx_base_vec, AAx_vec, base_year, issue_year)

ax_improved(x0, i, qx_base_vec, AAx_vec, base_year, issue_year)
```

Arguments

`qx_base`	Base-year one-year death probability.
`AAx`	Mortality improvement factor.
`base_year`	Base year.
`proj_year`	Projection year. May be scalar or vector.
`x0`	Issue age.

n	Number of years.
qx_base_vec	Base-year death probabilities for successive ages.
AAx_vec	Mortality improvement factors for successive ages.
issue_year	Issue year.
i	Effective annual interest rate.
u	Deferral period in years.

Details

The standard projection used is

$$q_x^{[Y]} = q_x^{[B]}(1 - AA_x)^{Y-B},$$

where B is the base year, Y is the projection year, and AA_x is the mortality improvement factor at age x.

Value

Numeric vector of projected one-year death probabilities.

---

multilife_contingent_probabilities

Contingent multi-life probabilities

Description

Computes probabilities associated with the order of death of two independent lives over a specified term.

Usage

```
tqxy1(x, y, n, tbl = NULL, model = NULL, ...)

tqyx1(x, y, n, tbl = NULL, model = NULL, ...)

tqxy2(x, y, n, tbl = NULL, model = NULL, ...)

tqyx2(x, y, n, tbl = NULL, model = NULL, ...)
```

Arguments

x	Age of the first life. May be scalar or vector.
y	Age of the second life. May be scalar or vector.
n	Term in years. May be scalar or vector.
tbl	Optional life table object retained for backward compatibility. Continuous calculations currently require model.
model	Parametric survival model.
...	Additional parameters passed to the survival and hazard functions.

Details

`tqxy1()` computes the probability that the first life dies before the second life within n years.

`tqyx1()` computes the probability that the second life dies before the first life within n years.

`tqxy2()` computes the probability that the first life dies after the second life but within n years.

`tqyx2()` computes the probability that the second life dies after the first life but within n years.

These functions require a parametric survival model because an annual life table does not uniquely determine the within-year order of death without an additional interpolation assumption.

All calculations assume the two future lifetimes are independent.

The probabilities are obtained by integrating the joint survival function together with the appropriate force of mortality.

The four functions partition the probability that one of the two lives dies within the specified term according to the order of death.

Value

A numeric vector of contingent probabilities.

Examples

```
tqxy1(
  40, 50,
  n = 10,
  model = "uniform",
  omega = 100
)

tqyx1(
  40, 50,
  n = 10,
  model = "uniform",
  omega = 100
)

tqxy2(
  40, 50,
  n = 10,
  model = "uniform",
  omega = 100
)

tqyx2(
  40, 50,
  n = 10,
  model = "uniform",
  omega = 100
)
```

`multilife_pure_endowments` *Multi-life pure endowments*

Description

Computes pure endowment actuarial present values for joint-life and last-survivor statuses for two independent lives.

Usage

```
nExy(x, y, n, i, tbl = NULL, model = NULL, ...)

nExybar(x, y, n, i, tbl = NULL, model = NULL, ...)
```

Arguments

`x`	Age of the first life. May be scalar or vector.
`y`	Age of the second life. May be scalar or vector.
`n`	Nonnegative integer term. May be scalar or vector.
`i`	Effective annual interest rate. May be scalar or vector.
`tbl`	Optional life table object.
`model`	Optional parametric survival model.
`...`	Additional parameters passed to the survival model.

Details

`nExy()` computes an n-year joint-life pure endowment, payable at time n if both lives survive.

`nExybar()` computes an n-year last-survivor pure endowment, payable at time n if at least one life survives.

The joint-life pure endowment is

$$ {}_nE_{xy} = v^n \, {}_np_{xy}. $$

The last-survivor pure endowment is

$$ {}_nE_{\overline{xy}} = v^n \, {}_np_{\overline{xy}}. $$

Under independence,

$$ {}_np_{xy} = {}_np_x \, {}_np_y $$

and

$$ {}_np_{\overline{xy}} = {}_np_x + {}_np_y - {}_np_x \, {}_np_y. $$

Value

A numeric vector of actuarial present values.

Examples

```
nExy(
  x = 40,
  y = 50,
  n = 10,
  i = 0.05,
  model = "uniform",
  omega = 100
)

nExybar(
  x = 40,
  y = 50,
  n = 10,
  i = 0.05,
  model = "uniform",
  omega = 100
)
```

---

multilife_survival_probabilities

Multi-life survival and failure probabilities

Description

Computes joint-life and last-survivor survival and failure probabilities for two independent lives.

Usage

```
tpxy(x, y, t, tbl = NULL, model = NULL, ...)

tqxy(x, y, t, tbl = NULL, model = NULL, ...)

tpxybar(x, y, t, tbl = NULL, model = NULL, ...)

tqxybar(x, y, t, tbl = NULL, model = NULL, ...)
```

Arguments

x	Age of the first life. May be scalar or vector.
y	Age of the second life. May be scalar or vector.
t	Nonnegative duration. May be scalar or vector.
tbl	Optional life table object.
model	Optional parametric survival model.
...	Additional parameters passed to the survival model.

Details

`tpxy()` computes the joint-life survival probability

$$ {}_tp_{xy} = {}_tp_x \, {}_tp_y. $$

`tqxy()` computes the joint-life failure probability

$$ {}_tq_{xy} = 1 - {}_tp_{xy}. $$

`tpxybar()` computes the last-survivor survival probability

$$ {}_tp_{\overline{xy}} = {}_tp_x + {}_tp_y - {}_tp_x \, {}_tp_y. $$

`tqxybar()` computes the last-survivor failure probability

$$ {}_tq_{\overline{xy}} = 1 - {}_tp_{\overline{xy}}. $$

All calculations assume the future lifetimes are independent.

Joint-life probabilities require both lives to satisfy the survival condition, whereas last-survivor probabilities require at least one life to survive.

Value

A numeric vector of probabilities.

Examples

```
tpxy(
  40, 50,
  t = 10,
  model = "uniform",
  omega = 100
)

tqxy(
  40, 50,
  t = 10,
  model = "uniform",
  omega = 100
)

tpxybar(
  40, 50,
  t = 10,
  model = "uniform",
  omega = 100
)

tqxybar(
  40, 50,
  t = 10,
```

```
  model = "uniform",
  omega = 100
)
```

`mux_tab` *Fractional force of mortality from a life table*

Description

Computes μ_{x+t} under UDD, constant force, or Balducci.

Usage

```
mux_tab(tbl, x, t, assumption = c("udd", "cf", "balducci"))
```

Arguments

`tbl`	A life_table object.
`x`	Numeric vector of integer ages.
`t`	Numeric vector of fractional durations with $0 \le t \le 1$.
`assumption`	One of `"udd"`, `"cf"`, `"balducci"`.

Value

Numeric vector of μ_{x+t} values.

`nAbarx` *Continuous deferred insurance APV*

Description

Computes ${}_{n|}\bar{A}_x = v^n {}_np_x \bar{A}_{x+n}$.

Usage

```
nAbarx(x, n, i, model, ...)
```

Arguments

`x`	Age.
`n`	Deferral period.
`i`	Effective annual interest rate.
`model`	Parametric survival model name.
`...`	Additional model parameters passed to survival-model functions.

Value

Numeric vector of APVs.

`nAbarx_udd` *UDD approximation of continuous deferred insurance*

Description

Computes ${}_{n|}\bar{A}_x = (i/\delta)\,{}_{n|}A_x$.

Usage

```
nAbarx_udd(nAx, i)
```

Arguments

`nAx`	Discrete deferred insurance APV.
`i`	Effective annual interest rate.

Value

Continuous deferred insurance APV under UDD.

`nAx` *Deferred insurance APV*

Description

Computes ${}_{n|}A_x = {}_nE_x A_{x+n}$.

Usage

```
nAx(x, n, i, tbl = NULL, model = NULL, ..., tol = 1e-12, k_max = 5000)
```

Arguments

`x`	Age.
`n`	Deferral period.
`i`	Effective annual interest rate.
`tbl`	Optional life table object.
`model`	Optional parametric survival model name.
`...`	Additional arguments passed to survival-model functions.
`tol`	Numerical tolerance for truncating infinite sums.
`k_max`	Maximum number of terms in the sum.

Value

Numeric vector of APVs.

`nAx_m` *m-thly deferred insurance APV*

Description

Computes ${}_{n|}A_x^{(m)} = v^n {}_np_x A_{x+n}^{(m)}$.

Usage

```
nAx_m(x, n, i, m, model, ..., tol = 1e-12, j_max = 100000L)
```

Arguments

`x`	Age.
`n`	Deferral period.
`i`	Effective annual interest rate.
`m`	Positive integer payment frequency.
`model`	Parametric survival model name.
`...`	Additional model parameters passed to survival-model functions.
`tol`	Numerical tolerance for truncating the infinite sum.
`j_max`	Maximum number of m-thly intervals in the sum.

Value

Numeric vector of APVs.

`nAx_m_udd` *UDD approximation of m-thly deferred insurance*

Description

Computes ${}_{n|}A_x^{(m)} = (i/i^{(m)})\, {}_{n|}A_x$.

Usage

```
nAx_m_udd(nAx, i, m)
```

Arguments

nAx	Discrete deferred insurance APV.
i	Effective annual interest rate.
m	Positive integer payment frequency.

Value

m-thly deferred insurance APV under UDD.

NC_EAN_db *Entry Age Normal normal cost*

Description

Computes Entry Age Normal normal cost as total benefit APV divided by an active-service annuity-due factor.

Usage

```
NC_EAN_db(APV_total, adue_active)
```

Arguments

APV_total	Nonnegative total actuarial present value of benefits.
adue_active	Positive active-service annuity-due factor.

Value

A numeric vector.

Examples

```
NC_EAN_db(APV_total = 25000, adue_active = 15)
```

NC_PUC_db *Projected Unit Credit normal cost*

Description

Computes the actuarial present value of the portion of the projected benefit attributed to the current year of service.

Usage

```
NC_PUC_db(projected_benefit, total_service, v_to_ret, p_surv, adue_ret)
```

Arguments

projected_benefit	Nonnegative projected annual benefit at retirement.
total_service	Positive total service at retirement.
v_to_ret	Nonnegative discount factor from the valuation date to retirement.
p_surv	Survival or active-service probability to retirement in [0, 1].
adue_ret	Positive retirement annuity-due factor.

Value

A numeric vector.

Examples

```
NC_PUC_db(
  projected_benefit = 30000,
  total_service = 30,
  v_to_ret = 0.5,
  p_surv = 0.9,
  adue_ret = 12
)
```

NC_TUC_db *Traditional Unit Credit normal cost*

Description

Computes the actuarial present value of the benefit accrued during the current year.

Usage

```
NC_TUC_db(accrual_benefit, v_to_ret, p_surv, adue_ret)
```

Arguments

accrual_benefit	Nonnegative benefit accrued during the current year.
v_to_ret	Nonnegative discount factor from the valuation date to retirement.
p_surv	Survival or active-service probability to retirement in [0, 1].
adue_ret	Positive retirement annuity-due factor.

Value

A numeric vector.

Examples

```
NC_TUC_db(
  accrual_benefit = 1560,
  v_to_ret = 0.5,
  p_surv = 0.9,
  adue_ret = 12
)
```

---

ndx *Compute deaths over an n-year interval from a life table*

Description

Compute deaths over an n-year interval from a life table

Usage

```
ndx(tbl, x, n)
```

Arguments

tbl	A life_table object.
x	Ages.
n	Nonnegative integer durations.

Value

Numeric vector of ${}_nd_x$ values.

nEx *Pure endowment APV*

Description

Computes ${}_nE_x = v^n {}_np_x$.

Usage

```
nEx(x, n, i, tbl = NULL, model = NULL, ...)
```

Arguments

x	Age.
n	Term.
i	Effective annual interest rate.
tbl	Optional life table object.
model	Optional parametric survival model name.
...	Additional arguments passed to survival-model functions.

Value

Numeric vector of APVs.

nkqx *Curtate death probability from a life table*

Description

Computes ${}_{k|}q_x = {}_kp_x - {}_{k+1}p_x$.

Usage

```
nkqx(tbl, x, k)
```

Arguments

tbl	A life_table object.
x	Numeric vector of ages.
k	Nonnegative integer.

Value

Numeric vector of ${}_{k|}q_x$ values.

nmxq — *Deferred death probability from a life table*

Description

Computes the probability that a life aged x survives n years and then dies within the following m years: ${}_{n|m}q_x = {}_np_x \cdot {}_mq_{x+n}$.

Usage

```
nmxq(tbl, x, n, m)
```

Arguments

tbl	A life_table object.
x	Numeric vector of ages.
n	Nonnegative integer deferred period.
m	Nonnegative integer subsequent period.

Details

This function is for integer n and m in the discrete tabular setting.

Value

Numeric vector of ${}_{n|m}q_x$ values.

nmxq_select — *Deferred select-life death probability*

Description

Computes ${}_{n|m}q_{[x]+t} = {}_np_{[x]+t} \cdot {}_mq_{[x]+t+n}$.

Usage

```
nmxq_select(tbl, x_sel, t, n, m)
```

Arguments

tbl	A select_life_table object.
x_sel	Numeric vector of ages at selection.
t	Numeric vector of current durations since selection.
n	Numeric vector of nonnegative integer deferred periods.
m	Numeric vector of nonnegative integer death windows.

Value

Numeric vector of deferred death probabilities.

`NPV_partial` *Partial net present values*

Description

Computes cumulative discounted profits through each duration. A scalar discount rate returns a named vector; vectorized rates return a matrix.

Usage

```
NPV_partial(Pi, r)
```

Arguments

`Pi`	Numeric profit-signature vector.
`r`	Annual effective risk discount rate. May be scalar or vector; values must be greater than `-1`.

Value

A named numeric vector for scalar `r`, or a numeric matrix for vectorized `r`.

Examples

```
Pi <- c(-15.00, 8.42, 8.39, 8.58)
NPV_partial(Pi, r = 0.10)
```

`NPV_profit` *Net present value of a profit signature*

Description

Discounts a profit signature at one or more annual effective risk discount rates.

Usage

```
NPV_profit(Pi, r)
```

Arguments

`Pi`	Numeric profit-signature vector.
`r`	Annual effective risk discount rate. May be scalar or vector; values must be greater than `-1`.

Value

A numeric vector with one value for each rate in `r`.

Examples

```
Pi <- c(-15.00, 8.42, 8.39, 8.58)
NPV_profit(Pi, r = 0.10)
NPV_profit(Pi, r = c(0.08, 0.10, 0.12))
```

`npx` *Compute n-year survival probability from a life table*

Description

Compute n-year survival probability from a life table

Usage

```
npx(tbl, x, n)
```

Arguments

`tbl`	A `life_table` object.
`x`	Ages.
`n`	Nonnegative integer durations.

Value

Numeric vector of ${}_np_x$ values.

`npxtau_md` *Multiple-decrement survival probability from a table*

Description

Computes

$$ {}_np_x^{(\tau)} = \prod_{k=0}^{n-1} p_{x+k}^{(\tau)}. $$

Usage

```
npxtau_md(tbl, x, n)
```

Arguments

tbl	A multiple-decrement table produced by `md_table()`.
x	Starting age or duration. May be scalar or vector.
n	Nonnegative integer term. May be scalar or vector.

Value

A numeric vector of survival probabilities.

Examples

```
ages <- 45:50
qmat <- cbind(
  q1 = c(0.011, 0.012, 0.013, 0.014, 0.015, 0.016),
  q2 = rep(0.100, 6)
)
tbl <- md_table(ages, qmat, radix = 1000)

npxtau_md(tbl, x = 46, n = 3)
```

---

npx_select *Select-life survival probability*

Description

Computes ${}_np_{[x]+t} = l_{[x]+t+n}/l_{[x]+t}$ in the discrete select-table setting.

Usage

```
npx_select(tbl, x_sel, t, n)
```

Arguments

tbl	A `select_life_table` object.
x_sel	Numeric vector of ages at selection.
t	Numeric vector of current durations since selection.
n	Numeric vector of nonnegative integer future durations.

Value

Numeric vector of survival probabilities.

nqx *Compute n-year death probability from a life table*

Description

Compute n-year death probability from a life table

Usage

```
nqx(tbl, x, n)
```

Arguments

tbl	A life_table object.
x	Ages.
n	Nonnegative integer durations.

Value

Numeric vector of ${}_nq_x$ values.

nqxj_md *Cause-specific multiple-decrement probability from a table*

Description

Computes the probability of decrement from cause j within n years:

$$ {}_nq_x^{(j)} = \sum_{k=0}^{n-1} {}_kp_x^{(\tau)} q_{x+k}^{(j)}. $$

Usage

```
nqxj_md(tbl, x, n, j)
```

Arguments

tbl	A multiple-decrement table produced by md_table().
x	Starting age or duration. May be scalar or vector.
n	Nonnegative integer term. May be scalar or vector.
j	Positive integer cause index. May be scalar or vector.

Value

A numeric vector of cause-specific decrement probabilities.

Examples

```
ages <- 45:50
qmat <- cbind(
  q1 = c(0.011, 0.012, 0.013, 0.014, 0.015, 0.016),
  q2 = rep(0.100, 6)
)
tbl <- md_table(ages, qmat, radix = 1000)

nqxj_md(tbl, x = 46, n = 2, j = 1)
```

`nqxtau_md` *Total multiple-decrement probability from a table*

Description

Computes

$$ {}_nq_x^{(\tau)} = 1 - {}_np_x^{(\tau)}. $$

Usage

```
nqxtau_md(tbl, x, n)
```

Arguments

`tbl`	A multiple-decrement table produced by `md_table()`.
`x`	Starting age or duration. May be scalar or vector.
`n`	Nonnegative integer term. May be scalar or vector.

Value

A numeric vector of total decrement probabilities.

Examples

```
ages <- 45:50
qmat <- cbind(
  q1 = c(0.011, 0.012, 0.013, 0.014, 0.015, 0.016),
  q2 = rep(0.100, 6)
)
tbl <- md_table(ages, qmat, radix = 1000)

nqxtau_md(tbl, x = 46, n = 2)
```

nqx_select *Select-life death probability*

Description

Computes ${}_nq_{[x]+t} = 1 - {}_np_{[x]+t}$.

Usage

```
nqx_select(tbl, x_sel, t, n)
```

Arguments

tbl	A select_life_table object.
x_sel	Numeric vector of ages at selection.
t	Numeric vector of current durations since selection.
n	Numeric vector of nonnegative integer future durations.

Value

Numeric vector of death probabilities.

PAB_cae *Projected annual benefit under a career-average-earnings plan*

Description

Projects the annual benefit using a career-average-earnings formula.

Usage

```
PAB_cae(x, z, CASx, p, past_salary_total = 0, g = NULL, s = NULL)
```

Arguments

x	Scalar current or entry age.
z	Scalar retirement age.
CASx	Positive scalar current annual salary.
p	Nonnegative scalar accrual percentage.
past_salary_total	Nonnegative total of actual prior salaries.
g	Optional scalar annual salary growth rate greater than -1.
s	Optional positive salary-scale vector of length z - x.

Value

A numeric scalar.

Examples

```
PAB_cae(
  x = 30,
  z = 65,
  CASx = 100000,
  p = 1,
  g = 0.04
)
```

`PAB_fas` *Projected annual benefit under a final-average-salary plan*

Description

Projects the annual benefit using a final-average-salary formula.

Usage

```
PAB_fas(x, z, CASx, p, fas_years = 3, past_service = 0, g = NULL, s = NULL)
```

Arguments

`x`	Scalar current or entry age.
`z`	Scalar retirement age.
`CASx`	Positive scalar current annual salary.
`p`	Nonnegative scalar accrual percentage, such as `2` for 2 percent.
`fas_years`	Positive integer number of years in the final salary average.
`past_service`	Nonnegative scalar years of service already completed.
`g`	Optional scalar annual salary growth rate greater than `-1`.
`s`	Optional positive salary-scale vector of length `z - x`.

Value

A numeric scalar.

Examples

```
PAB_fas(
  x = 35,
  z = 65,
  CASx = 60000,
  p = 2,
  fas_years = 3,
  g = 0.04
)
```

Pbar_trapz_ms *Continuous premium approximation in a disability model*

Description

Approximates a continuous premium rate by trapezoidal integration.

Usage

```
Pbar_trapz_ms(t, tp00, tp01, delta, mu02, mu12, B02 = 1, B12 = 1, R = 0)
```

Arguments

t	Strictly increasing nonnegative time points.
tp00	Healthy-state probabilities.
tp01	Disabled-state probabilities.
delta	Force of interest.
mu02	Healthy-to-deceased intensity function.
mu12	Disabled-to-deceased intensity function.
B02	Benefit on death while healthy.
B12	Benefit on death while disabled.
R	Continuous disability income rate.

Value

A numeric scalar.

Examples

```
mu01 <- function(t) 0.10 * t + 0.20
mu02 <- function(t) 0.20
mu10 <- function(t) 0.50
mu12 <- function(t) 0.125 * t + 0.20

probs <- tp00_tp01_euler(
  h = 0.10,
  n = 2,
  mu01 = mu01,
  mu02 = mu02,
  mu10 = mu10,
  mu12 = mu12
)

Pbar_trapz_ms(
  t = probs$t,
  tp00 = probs$tp00,
  tp01 = probs$tp01,
  delta = 0.04,
  mu02 = mu02,
  mu12 = mu12,
  B02 = 1000,
  B12 = 1000,
  R = 1000
)
```

Pi_signature *Profit signature*

Description

Converts policy-year expected profits into a profit signature by weighting each future expected profit by the probability that the contract is in force at the start of that policy year.

Usage

```
Pi_signature(Pr, p_tau)
```

Arguments

Pr	Profit vector of length n + 1.
p_tau	One-year in-force probabilities. For n > 1, this may have length n - 1 or n; the final value is ignored when length n. For a one-year contract, use numeric(0) or a single probability.

Value

A named numeric vector with the same length as Pr.

Examples

```
Pr <- c(-15.00, 8.42, 8.40, 8.61)
Pi_signature(Pr, p_tau = c(0.99858, 0.99847, 0.99834))
```

`PnAdotx` *Net premium for a deferred annuity-due*

Description

Computes

$$P({}_{n|}\ddot{a}_x) = \frac{{}_{n|}\ddot{a}_x}{\ddot{a}_{x:\overline{n}|}}.$$

Usage

```
PnAdotx(x, n, i, model = NULL, ..., tbl = NULL)
```

Arguments

`x`	Issue age. May be scalar or vector.
`n`	Positive integer deferral period. May be scalar or vector.
`i`	Effective annual interest rate. May be scalar or vector.
`model`	Optional parametric survival model.
`...`	Additional parameters passed to the actuarial functions.
`tbl`	Optional life table object. Supply by name.

Value

Numeric vector of net annual premiums.

Examples

```
PnAdotx(
  40,
  n = 20,
  i = 0.05,
  model = "uniform",
  omega = 100
)
```

Pnax *Net premium for a deferred annuity-immediate*

Description

Computes

$$P({}_{n|}a_x) = \frac{{}_{n|}a_x}{\ddot{a}_{x:\overline{n}|}}.$$

Usage

```
Pnax(x, n, i, model = NULL, ..., tbl = NULL)
```

Arguments

x	Issue age. May be scalar or vector.
n	Positive integer deferral period. May be scalar or vector.
i	Effective annual interest rate. May be scalar or vector.
model	Optional parametric survival model.
...	Additional parameters passed to the actuarial functions.
tbl	Optional life table object. Supply by name.

Value

Numeric vector of net annual premiums.

Examples

```
Pnax(
  40,
  n = 20,
  i = 0.05,
  model = "uniform",
  omega = 100
)
```

premium_functions *Premium, loss, and expense functions*

Description

Functions for net and gross premiums, present-value-of-loss moments, continuous-payment premium rates, and premiums payable more frequently than annually.

Usage

```
Px(x, i, tbl = NULL, model = NULL, ...)

Pxn1(x, n, i, tbl = NULL, model = NULL, ...)

PnEx(x, n, i, tbl = NULL, model = NULL, ...)

Pxn(x, n, i, tbl = NULL, model = NULL, ...)

tPx(x, t, i, tbl = NULL, model = NULL, ...)

tPxn1(x, n, t, i, tbl = NULL, model = NULL, ...)

tPnEx(x, n, t, i, tbl = NULL, model = NULL, ...)

tPxn(x, n, t, i, tbl = NULL, model = NULL, ...)

PnAx(x, n, i, tbl = NULL, model = NULL, ...)

tPnAx(x, n, t, i, tbl = NULL, model = NULL, ...)

Pbarx(x, i, model, ..., tol = 1e-10)

Pbarxn1(x, n, i, model, ...)

Pbarxn(x, n, i, model, ...)

PbarAbarx(x, i, model, ..., tol = 1e-10)

PbarAbarxn1(x, n, i, model, ...)

PbarAbarxn(x, n, i, model, ...)

Px_m(x, m, i, tbl = NULL, model = NULL, ...)

Pxn1_m(x, n, m, i, tbl = NULL, model = NULL, ...)

Pxn_m(x, n, m, i, tbl = NULL, model = NULL, ...)
```

```
PnAx_m(x, n, m, i, tbl = NULL, model = NULL, ...)

EL0x(x, P, i, tbl = NULL, model = NULL, ...)

varL0x(x, P, i, tbl = NULL, model = NULL, ..., tol = 1e-12, k_max = 5000)

EL0xn1(x, n, P, i, tbl = NULL, model = NULL, ...)

varL0xn1(x, n, P, i, tbl = NULL, model = NULL, ...)

EL0xn(x, n, P, i, tbl = NULL, model = NULL, ...)

varL0xn(x, n, P, i, tbl = NULL, model = NULL, ...)

EL0barAbarx(x, P, i, model, ..., tol = 1e-10)

varL0barAbarx(x, P, i, model, ...)

Gx(
  x,
  i,
  benefit = 1,
  first_premium_pct = 0,
  renewal_premium_pct = 0,
  first_policy_exp = 0,
  renewal_policy_exp = 0,
  settlement_exp = 0,
  tbl = NULL,
  model = NULL,
  ...
)
```

Arguments

x	Age. May be scalar or vector.
i	Effective annual interest rate. May be scalar or vector.
tbl	Optional life table object.
model	Optional parametric survival model name.
...	Additional arguments passed to survival-model functions.
n	Term. May be scalar or vector of nonnegative integers.
t	Premium-paying period. May be scalar or vector of nonnegative integers.
tol	Numerical tolerance for functions that truncate infinite sums.
m	Number of payments per year. Must be a positive integer scalar.
P	Premium amount or premium rate. May be scalar or vector.
k_max	Maximum summation horizon for functions that truncate infinite sums.

`benefit` Benefit amount. May be scalar or vector.

`first_premium_pct`
First-year premium expense proportion. May be scalar or vector.

`renewal_premium_pct`
Renewal premium expense proportion. May be scalar or vector.

`first_policy_exp`
First-year fixed expense. May be scalar or vector.

`renewal_policy_exp`
Renewal fixed expense after the first year. May be scalar or vector.

`settlement_exp` Settlement expense incurred at benefit payment. May be scalar or vector.

Details

The functions include:

- net annual premiums under the equivalence principle,
- limited-payment premiums,
- continuous-payment premium rates,
- fully continuous premium rates,
- true fractional premiums,
- present-value-of-loss means and variances,
- a basic gross premium formula for whole life insurance.

The discrete premium functions may be evaluated from either a life table supplied through `tbl` or a parametric survival model supplied through `model`. Continuous-payment premium functions are evaluated through the corresponding continuous insurance and annuity functions.

Scalar inputs retain their existing behavior. Where mathematically meaningful, numeric inputs may also be vectors. Inputs are evaluated elementwise when they have a common length, and scalar inputs are recycled.

Value

Numeric vector.

`profit_margin` *Profit margin*

Description

Computes net present value divided by the actuarial present value of gross premiums. Scalar arguments are recycled to the common length.

Usage

```
profit_margin(NPV, APV_GP)
```

Arguments

NPV	Net present value of profits.
APV_GP	Positive actuarial present value of gross premiums.

Value

A numeric vector.

Examples

```
profit_margin(6.03, 259.52)
```

Pr_vector_disc *Profit vector for a discrete profit-analysis model*

Description

Computes expected profit by policy year for a discrete contract with up to two decrements. The first element is the negative pre-contract expense.

Usage

```
Pr_vector_disc(
  V,
  G,
  i,
  r = 0,
  e = 0,
  q1,
  q2 = 0,
  b1,
  b2 = 0,
  s1 = 0,
  s2 = 0,
  p_tau = NULL,
  pre_contract_expense = 0
)
```

Arguments

V	Numeric vector of gross premium reserves with length n + 1, including the issue-time and terminal reserves.
G	Gross premium by policy year.
i	Annual effective interest rate by policy year. Values must be greater than -1.
r	Percent-of-premium expense rate by policy year. Values must lie in [0, 1].

e	Fixed expense by policy year.
q1	Probability of the first decrement by policy year.
q2	Probability of the second decrement by policy year.
b1	Benefit payable on the first decrement.
b2	Benefit payable on the second decrement.
s1	Settlement expense associated with the first decrement.
s2	Settlement expense associated with the second decrement.
p_tau	Optional in-force probability by policy year. If omitted, it is calculated as 1 - q1 - q2.
pre_contract_expense	Nonnegative scalar pre-contract expense.

Details

For policy year k, the expected profit is

$$[V_{k-1} + G_k(1 - r_k) - e_k](1 + i_k) - [(b_k^{(1)} + s_k^{(1)})q_k^{(1)} + (b_k^{(2)} + s_k^{(2)})q_k^{(2)} + V_k p_k^{(\tau)}].$$

Scalar yearly inputs are recycled to the number of policy years determined by `length(V) - 1`.

Value

A named numeric vector of length `n + 1`.

Examples

```
V <- c(0, 5.66, 6.17, 0)
qx <- c(0.00142, 0.00153, 0.00166)
Pr_vector_disc(
  V = V, G = 95, i = 0.06, r = 0.05, e = 10,
  q1 = qx, b1 = 50000, pre_contract_expense = 15
)
```

---

pv_cashflows *Present value of cash flows at time 0*

Description

Present value of cash flows at time 0

Usage

```
pv_cashflows(cf, t, i)
```

Arguments

cf	Cash flow amounts.
t	Times of cash flows.
i	Effective interest rate.

Value

Present value at time 0.

Examples

```
pv_cashflows(c(-100, 60, 60), c(0, 1, 2), i = 0.10)
```

---

pv_spot_cashflows *Present value of deterministic cash flows using spot rates*

Description

Discounts each cash flow using the spot rate matched to its payment time. Time-zero cash flows are not discounted.

Usage

```
pv_spot_cashflows(
  amounts,
  times,
  spot,
  compounding = c("annual", "semiannual")
)
```

Arguments

amounts	Numeric vector of cash-flow amounts.
times	Numeric vector of payment times in years.
spot	Numeric vector of spot rates matched elementwise to times. The value corresponding to a time-zero cash flow is ignored.
compounding	Character string equal to "annual" or "semiannual".

Value

A numeric scalar.

Examples

```
pv_spot_cashflows(
  amounts = c(200000, 50000, 50000, 100000),
  times = c(0, 0.5, 1, 2),
  spot = c(0, 0.02440, 0.02601, 0.02936),
  compounding = "semiannual"
)
```

---

pxtau *Total one-year survival probability*

Description

Computes

$$p_x^{(\tau)} = 1 - q_x^{(\tau)}.$$

Usage

```
pxtau(qxj)
```

Arguments

qxj	Numeric vector of cause-specific decrement probabilities.

Value

A numeric scalar.

Examples

```
pxtau(c(0.011, 0.100))
```

---

pxtau_ul *Universal life persistency probabilities*

Description

`pxtau_ul()` computes one-year persistency probabilities under mortality and withdrawal.

Usage

```
pxtau_ul(qd, qw, year_end_withdrawal = TRUE)

tpxtau_ul(qd, qw, year_end_withdrawal = TRUE)
```

Arguments

qd	Mortality probabilities.
qw	Withdrawal probabilities.
year_end_withdrawal	Logical scalar. If TRUE, withdrawal is modeled at year-end and persistency is $(1-q^{(d)})(1-q^{(w)})$. Otherwise persistency is $1-q^{(d)}-q^{(w)}$.

Details

`tpxtau_ul()` computes cumulative persistency through the end of each policy year.

Value

A numeric vector.

Examples

```
qd <- c(0.001, 0.002, 0.003)
qw <- c(0.02, 0.02, 0.03)

pxtau_ul(qd, qw)
tpxtau_ul(qd, qw)
```

---

px_to_lx *Construct life-table values from p_x values*

Description

Builds life-table survivor values recursively from $l_{x+1} = l_x p_x$, starting from a chosen radix.

Usage

```
px_to_lx(px, radix = 1e+05)
```

Arguments

px	Numeric vector of one-year survival probabilities p_x.
radix	Positive radix l_0.

Value

Numeric vector of l_x values of length `length(px) + 1`.

qxprime_mudd *Associated single-decrement probabilities under MUDD*

Description

Converts dependent multiple-decrement probabilities to associated single-decrement probabilities under the multiple-decrement uniform distribution assumption.

Usage

```
qxprime_mudd(qxj)
```

Arguments

qxj	Numeric vector of dependent cause-specific decrement probabilities.

Value

A numeric vector of associated single-decrement probabilities.

Examples

```
qxprime_mudd(c(0.20, 0.10))
```

qxprime_sudd *Associated single-decrement probabilities under SUDD*

Description

Converts two dependent multiple-decrement probabilities to associated single-decrement probabilities under the single-decrement uniform distribution assumption.

Usage

```
qxprime_sudd(q1, q2)
```

Arguments

q1	Dependent decrement probability for cause 1. May be scalar or vector.
q2	Dependent decrement probability for cause 2. May be scalar or vector.

Value

For scalar input, a named numeric vector containing q1prime and q2prime. For vectorized input, a numeric matrix with columns q1prime and q2prime.

Examples

```
qxprime_sudd(0.20, 0.10)
```

---

qxtau *Total one-year decrement probability*

Description

Computes the total one-year multiple-decrement probability

$$q_x^{(\tau)} = \sum_j q_x^{(j)}.$$

Usage

```
qxtau(qxj)
```

Arguments

qxj	Numeric vector of cause-specific decrement probabilities.

Value

A numeric scalar.

Examples

```
qxtau(c(0.011, 0.100))
```

---

qx_dep_cf *Multiple-decrement probabilities under constant forces*

Description

Converts associated single-decrement probabilities to dependent cause-specific decrement probabilities under constant forces.

Usage

```
qx_dep_cf(qxprime)
```

Arguments

qxprime	Numeric vector of associated single-decrement probabilities. Each value must be less than one.

Value

A numeric vector of dependent cause-specific decrement probabilities.

Examples

```
qx_dep_cf(c(0.20, 0.10))
```

`qx_dep_sudd` *Multiple-decrement probabilities under SUDD*

Description

Converts two associated single-decrement probabilities to dependent multiple-decrement probabilities under the single-decrement uniform distribution assumption.

Usage

```
qx_dep_sudd(q1prime, q2prime)
```

Arguments

`q1prime`	Associated single-decrement probability for cause 1. May be scalar or vector.
`q2prime`	Associated single-decrement probability for cause 2. May be scalar or vector.

Value

For scalar input, a named numeric vector containing `q1` and `q2`. For vectorized input, a numeric matrix with columns `q1` and `q2`.

Examples

```
qx_dep_sudd(0.20, 0.10)
```

qx_tab *Compute one-year death probability from a life table*

Description

Compute one-year death probability from a life table

Usage

```
qx_tab(tbl, x)
```

Arguments

tbl	A life_table object.
x	Ages.

Value

Numeric vector of q_x values.

qx_to_lx *Construct life-table values from q_x values*

Description

Builds life-table survivor values recursively from $l_{x+1} = l_x(1 - q_x)$, starting from a chosen radix.

Usage

```
qx_to_lx(qx, radix = 1e+05)
```

Arguments

qx	Numeric vector of one-year death probabilities q_x.
radix	Positive radix l_0.

Value

Numeric vector of l_x values of length `length(qx) + 1`.

replacement_ratio_db *Replacement ratio for a defined benefit plan*

Description

Computes annual benefit divided by a selected salary measure. Scalar arguments are recycled to a common length.

Usage

```
replacement_ratio_db(benefit, salary)
```

Arguments

benefit	Nonnegative annual retirement benefit.
salary	Positive salary measure used in the denominator.

Value

A numeric vector.

Examples

```
replacement_ratio_db(
  benefit = 108008.66,
  salary = 187119.09
)
```

replacement_ratio_dc *Replacement ratio for a defined contribution plan*

Description

Computes annual retirement income divided by salary in the final pre-retirement year.

Usage

```
replacement_ratio_dc(x, z, Sx, c, i, adue_z, g = NULL, s = NULL)
```

Arguments

x	Scalar entry age.
z	Scalar retirement age.
Sx	Positive scalar salary at age x.
c	Scalar contribution rate in [0, 1].
i	Scalar annual effective investment return greater than -1.
adue_z	Positive scalar whole-life annuity-due factor at retirement.
g	Optional scalar annual salary growth rate greater than -1.
s	Optional positive salary-scale vector of length z - x.

Value

A numeric scalar.

Examples

```
replacement_ratio_dc(
  x = 30,
  z = 65,
  Sx = 50000,
  c = 0.10,
  i = 0.05,
  adue_z = 12,
  g = 0.04
)
```

---

reversionary_annuities

Reversionary annuity functions

Description

Computes reversionary whole-life annuities payable to one life after the death of the other life.

Usage

```
ax_y(x, y, i, tbl = NULL, model = NULL, ..., k_max = 5000L, tol = 1e-12)

ay_x(x, y, i, tbl = NULL, model = NULL, ..., k_max = 5000L, tol = 1e-12)
```

Arguments

x	Age of the first life. May be scalar or vector.
y	Age of the second life. May be scalar or vector.
i	Effective annual interest rate. May be scalar or vector.
tbl	Optional life table object.
model	Optional parametric survival model.
...	Additional parameters passed to the survival model or life-table functions.
k_max	Maximum number of terms used for an infinite series.
tol	Numerical tolerance used to assess convergence.

Details

`ax_y()` computes an annuity payable to the second life after the death of the first life.

`ay_x()` computes an annuity payable to the first life after the death of the second life.

Value

A numeric vector of actuarial present values.

---

rt_ul *Account-value to guaranteed-fund ratio*

Description

Computes the ratio of account value to guaranteed maturity fund, capped at one.

Usage

```
rt_ul(AV, GMF)
```

Arguments

AV	Nonnegative account value.
GMF	Positive guaranteed maturity fund.

Value

A numeric vector.

Examples

```
rt_ul(AV = 4918.20, GMF = 14678.57)
```

S0 *Survival function for age-at-failure*

Description

Computes $S_0(t) = Pr(T_0 > t)$.

Usage

```
S0(t, model, ...)
```

Arguments

t	Numeric vector of times.
model	One of "uniform", "exponential", "gompertz", "makeham", or "weibull".
...	Model parameters.

Value

Numeric vector of survival probabilities.

S0_to_lx *Convert survival probabilities to life-table values*

Description

Converts survival function values $S_0(x)$ into life-table survivor values $l_x = l_0 S_0(x)$ using a chosen radix.

Usage

```
S0_to_lx(S0, radix = 1e+05)
```

Arguments

S0	Numeric vector of survival probabilities.
radix	Positive radix l_0.

Value

Numeric vector of l_x values.

salary_scale *Salary scale under constant annual growth*

Description

Constructs salary-scale values under a constant annual growth rate.

Usage

```
salary_scale(k, g, base_age = min(k), s_base = 1)
```

Arguments

k	Numeric vector of ages or durations.
g	Annual salary growth rate greater than -1.
base_age	Scalar age or duration at which the scale is normalized.
s_base	Positive scalar salary-scale value at base_age.

Value

A numeric vector with the same length as k.

Examples

```
salary_scale(k = 30:34, g = 0.04, base_age = 30)
```

select_life_table *Construct a select life table*

Description

Builds a select-life-table object from vectors of selection age, duration since selection, attained age, and survivor values.

Usage

```
select_life_table(x_sel, duration, attained_age, lx)
```

Arguments

x_sel	Numeric vector of ages at selection.
duration	Numeric vector of durations since selection.
attained_age	Numeric vector of attained ages.
lx	Numeric vector of select-table survivor values.

Value

A data frame with class "select_life_table".

`solve_yield` *Solve the yield rate by the equation of value*

Description

Finds the interest rate ‘i‘ such that the present value of the cash flows is 0.

Usage

```
solve_yield(cf, t, interval = c(-0.99, 1), tol = 1e-10)
```

Arguments

`cf`	Cash flows.
`t`	Times.
`interval`	Two-length numeric vector bracketing the root.
`tol`	Tolerance passed to ‘uniroot()‘.

Value

Yield rate ‘i‘.

Examples

```
solve_yield(c(-100, 60, 60), c(0, 1, 2), interval = c(-0.5, 1))
```

`spot_interest_apvs` *Actuarial present values under spot rates*

Description

Computes life-contingent actuarial present values using annual effective spot rates by maturity.

Usage

```
nEx_spot(qx, z, benefit = 1)

Axn1_spot(qx, z, benefit = 1)

Axn_spot(qx, z, benefit = 1)

axn_spot(qx, z, type = c("immediate", "due"), benefit = 1)
```

Arguments

qx	Numeric vector of one-year mortality probabilities.
z	Numeric vector of annual effective spot rates for maturities 1, ..., n. Each value must be greater than -1.
benefit	Nonnegative scalar benefit or annuity payment amount.
type	Character string equal to "immediate" or "due".

Details

If z_t denotes the annual effective spot rate for maturity t, the corresponding discount factor is

$$(1 + z_t)^{-t}.$$

nEx_spot() computes a pure endowment.

Axn1_spot() computes term insurance payable at the end of the year of death.

Axn_spot() computes endowment insurance.

axn_spot() computes a temporary annuity-immediate or annuity-due.

Each payment is discounted using the spot rate corresponding to its maturity rather than a single level interest rate.

The pure endowment is

$$_nE = {}_np_x(1 + z_n)^{-n}.$$

Term insurance is obtained by discounting each death benefit using the spot rate corresponding to its payment year.

Endowment insurance equals the sum of the corresponding term insurance and pure endowment.

Temporary annuities discount each payment using the spot rate for its payment time.

Value

A numeric scalar.

Examples

```
qx <- c(0.02, 0.03, 0.04, 0.05, 0.06)
spot <- c(0.03, 0.04, 0.05, 0.06, 0.07)

nEx_spot(qx, spot, benefit = 1000)
Axn1_spot(qx, spot)
Axn_spot(qx, spot)
axn_spot(qx, spot, type = "due")
```

`thiele_backward_path` *Backward reserve path from a terminal value*

Description

Starting from a terminal reserve at the final time, computes reserves backward over a strictly increasing time grid using ‘thiele_backward_step()‘.

Usage

```
thiele_backward_path(times, V_terminal, P, delta, mu, benefit = 1)
```

Arguments

`times`	Finite numeric vector of strictly increasing times.
`V_terminal`	Finite scalar reserve at the final time.
`P`	Premium rate, scalar or vector with one value per time step.
`delta`	Force of interest, scalar or vector with one value per step.
`mu`	Nonnegative force of mortality, scalar or vector with one value per step.
`benefit`	Benefit amount, scalar or vector with one value per step.

Value

Numeric vector of reserve values corresponding to ‘times‘.

Examples

```
times <- seq(19, 20, by = 0.25)

thiele_backward_path(
  times,
  V_terminal = 1000,
  P = 26.96,
  delta = 0.058,
  mu = 0.002,
  benefit = 1000
)
```

`thiele_backward_step` *One backward numerical step for Thiele's equation*

Description

Approximates the reserve at time 't' from a known reserve at time 't + h'.

Usage

```
thiele_backward_step(V_next, P, delta, mu, benefit = 1, h = 1)
```

Arguments

`V_next`	Reserve at time 't + h'.
`P`	Premium rate.
`delta`	Force of interest.
`mu`	Nonnegative force of mortality at time 't'.
`benefit`	Benefit amount.
`h`	Positive step size.

Details

The implemented step is

$$V_t = \frac{V_{t+h} - hP + h\mu B}{1 + h(\delta + \mu)}.$$

Value

Numeric vector of reserve approximations.

Examples

```
thiele_backward_step(
  V_next = 1000,
  P = 26.96,
  delta = 0.058,
  mu = 0.002,
  benefit = 1000,
  h = 1
)
```

thiele_dVdt *Reserve derivative from Thiele's equation*

Description

Computes

$$\frac{dV}{dt} = P + \delta V - \mu(B - V).$$

Usage

```
thiele_dVdt(V, P, delta, mu, benefit = 1)
```

Arguments

V	Reserve at time 't'.
P	Premium rate.
delta	Force of interest.
mu	Nonnegative force of mortality.
benefit	Benefit amount.

Value

Numeric vector of reserve derivatives.

Examples

```
thiele_dVdt(
  V = 900,
  P = 25,
  delta = 0.05,
  mu = 0.002,
  benefit = 1000
)
```

thiele_dVdt_01 *Reserve derivatives for a disability model with recovery*

Description

Computes the coupled Thiele reserve derivatives. Numeric arguments may be scalar or compatible vectors.

Usage

```
thiele_dVdt_01(t, V0, V1, delta, Pbar, B, R, mu01, mu02, mu10, mu12)
```

Arguments

t	Time.
V0	Healthy-state reserve.
V1	Disabled-state reserve.
delta	Force of interest.
Pbar	Continuous premium rate.
B	Death benefit.
R	Continuous disability income rate.
mu01	Healthy-to-disabled intensity function.
mu02	Healthy-to-deceased intensity function.
mu10	Disabled-to-healthy intensity function.
mu12	Disabled-to-deceased intensity function.

Value

A named vector for scalar input or a two-column matrix for vectorized input.

---

thiele_path_01 *Backward reserve path for a disability model with recovery*

Description

Computes a backward Euler reserve path from terminal healthy-state and disabled-state reserves.

Usage

```
thiele_path_01(
  h,
  n,
  delta,
  Pbar,
  B,
  R,
  mu01,
  mu02,
  mu10,
  mu12,
  V0_n = 0,
  V1_n = 0
)
```

Arguments

h	Positive step size.
n	Nonnegative final time.
delta	Force of interest.
Pbar	Continuous premium rate.
B	Death benefit.
R	Continuous disability income rate.
mu01	Healthy-to-disabled intensity function.
mu02	Healthy-to-deceased intensity function.
mu10	Disabled-to-healthy intensity function.
mu12	Disabled-to-deceased intensity function.
V0_n	Terminal healthy-state reserve.
V1_n	Terminal disabled-state reserve.

Value

A data frame with columns `t`, `tV0`, and `tV1`.

`tp00_tp01_euler` *Euler approximation of disability-state probabilities*

Description

Approximates probabilities of being healthy, disabled, or deceased in a three-state model that allows recovery from disability. The final time is always included, with a shorter final step when needed.

Usage

```
tp00_tp01_euler(h, n, mu01, mu02, mu10, mu12, p00_0 = 1, p01_0 = 0)
```

Arguments

h	Positive step size.
n	Nonnegative final time.
mu01	Healthy-to-disabled intensity function.
mu02	Healthy-to-deceased intensity function.
mu10	Disabled-to-healthy intensity function.
mu12	Disabled-to-deceased intensity function.
p00_0	Initial healthy-state probability.
p01_0	Initial disabled-state probability.

Value

A data frame with columns `t`, `tp00`, `tp01`, and `tp02`.

Examples

```
mu01 <- function(t) 0.10 * t + 0.20
mu02 <- function(t) 0.20
mu10 <- function(t) 0.50
mu12 <- function(t) 0.125 * t + 0.20
tp00_tp01_euler(0.10, 2, mu01, mu02, mu10, mu12)
```

---

tpx *Conditional survival probability*

Description

Computes ${}_tp_x = S_0(x+t)/S_0(x)$.

Usage

```
tpx(t, x, model, ...)
```

Arguments

`t`	Numeric vector of durations.
`x`	Numeric vector of ages.
`model`	One of `"uniform"`, `"exponential"`, `"gompertz"`, `"makeham"`, or `"weibull"`.
`...`	Model parameters.

Value

Numeric vector of survival probabilities.

tpxprimej_cf *Single-decrement survival under a constant force*

Description

Computes

$$ {}_tp_x'^{(j)} = \exp(-\mu_j t). $$

Usage

```
tpxprimej_cf(mu, t)
```

Arguments

mu	Nonnegative constant force of decrement. May be scalar or vector.
t	Nonnegative time. May be scalar or vector.

Value

A numeric vector.

Examples

```
tpxprimej_cf(0.10, 5)
tpxprimej_cf(0.10, c(1, 5, 10))
```

tpx_tab *Fractional survival probability from a life table*

Description

Computes ${}_tp_x$ for $0 \le t \le 1$ from a discrete life table under UDD, constant force, or Balducci.

Usage

```
tpx_tab(tbl, x, t, assumption = c("udd", "cf", "balducci"))
```

Arguments

tbl	A life_table object.
x	Numeric vector of integer ages.
t	Numeric vector of fractional durations with $0 \le t \le 1$.
assumption	One of "udd", "cf", "balducci".

Value

Numeric vector of ${}_tp_x$ values.

tpx_tau_cf *Total survival under constant cause-specific forces*

Description

Computes

$$ {}_tp_x^{(\tau)} = \exp\left(-t \sum_j \mu_j\right). $$

Usage

```
tpx_tau_cf(mu, t)
```

Arguments

mu	Numeric vector of nonnegative cause-specific forces.
t	Nonnegative time. May be scalar or vector.

Value

A numeric vector.

Examples

```
tpx_tau_cf(c(0.10, 0.20), 5)
tpx_tau_cf(c(0.10, 0.20), c(1, 5, 10))
```

tqx *Conditional failure probability*

Description

Computes ${}_tq_x = 1 - {}_tp_x$.

Usage

```
tqx(t, x, model, ...)
```

Arguments

t	Numeric vector of durations.
x	Numeric vector of ages.
model	One of "uniform", "exponential", "gompertz", "makeham", or "weibull".
...	Model parameters.

Value

Numeric vector of failure probabilities.

`tqxj_cf` *Cause-specific decrement probability under constant forces*

Description

Computes

$$ {}_tq_x^{(j)} = \frac{\mu_j}{\sum_k \mu_k} \left[1 - \exp\left(-t \sum_k \mu_k \right) \right]. $$

Usage

```
tqxj_cf(mu, j, t)
```

Arguments

`mu`	Numeric vector of nonnegative cause-specific forces.
`j`	Positive integer cause index.
`t`	Nonnegative time. May be scalar or vector.

Value

A numeric vector.

Examples

```
tqxj_cf(c(0.10, 0.20), j = 1, t = 5)
```

`tqxprimej_cf` *Single-decrement failure under a constant force*

Description

Computes

$$ {}_tq_x'^{(j)} = 1 - \exp(-\mu_j t). $$

Usage

```
tqxprimej_cf(mu, t)
```

Arguments

mu	Nonnegative constant force of decrement. May be scalar or vector.
t	Nonnegative time. May be scalar or vector.

Value

A numeric vector.

Examples

```
tqxprimej_cf(0.10, 5)
```

---

tqxprime_mudd *Fractional-year associated single-decrement probabilities under MUDD*

Description

Computes the associated single-decrement probabilities through time t under the multiple-decrement uniform distribution assumption.

Usage

```
tqxprime_mudd(qxj, t)
```

Arguments

qxj	Numeric vector of dependent cause-specific decrement probabilities.
t	A single time in $[0, 1]$.

Value

A numeric vector of associated single-decrement probabilities.

Examples

```
tqxprime_mudd(c(0.20, 0.10), t = 0.5)
```

tqx_tab *Fractional failure probability from a life table*

Description

Computes ${}_tq_x = 1 - {}_tp_x$ for $0 \leq t \leq 1$.

Usage

```
tqx_tab(tbl, x, t, assumption = c("udd", "cf", "balducci"))
```

Arguments

tbl	A life_table object.
x	Numeric vector of integer ages.
t	Numeric vector of fractional durations with $0 \leq t \leq 1$.
assumption	One of "udd", "cf", "balducci".

Value

Numeric vector of ${}_tq_x$ values.

tVbarAbarx *Fully continuous whole life reserve*

Description

Computes the reserve for a whole life insurance with continuous premiums and immediate payment of claims.

Usage

```
tVbarAbarx(x, t, i, model = NULL, ..., tbl = NULL)
```

Arguments

x	Issue age.
t	Duration, allowed to be any nonnegative numeric value.
i	Effective annual interest rate.
model	Optional parametric survival model name.
...	Additional model parameters.
tbl	Optional life table object.

Details

In the fully continuous setting, reserve time t may be any nonnegative real value.

Value

A numeric vector of values.

Examples

```
tVbarAbarx(40, t = 10, i = 0.05, model = "uniform", omega = 100)
tVbarAbarx(40, t = c(19, 19.25, 19.5, 19.75, 20), i = 0.06,
           model = "uniform", omega = 100)
```

---

tVbarx *Whole life reserve with continuous premiums*

Description

Computes the reserve for a discrete whole life insurance funded by continuous premiums.

Usage

```
tVbarx(x, t, i, model = NULL, ..., tbl = NULL)
```

Arguments

x	Issue age.
t	Duration.
i	Effective annual interest rate.
model	Optional parametric survival model name.
...	Additional model parameters.
tbl	Optional life table object.

Value

A numeric vector of values.

Examples

```
tVbarx(40, t = 10, i = 0.05, model = "uniform", omega = 100)
```

tVnAdotx *Reserve for a deferred annuity-due*

Description

Computes the reserve during the deferral period, where $0 \leq t < n$.

Usage

```
tVnAdotx(x, n, t, i, model = NULL, ..., tbl = NULL)
```

Arguments

x	Issue age. May be scalar or vector.
n	Positive integer deferral period. May be scalar or vector.
t	Nonnegative integer duration. May be scalar or vector.
i	Effective annual interest rate. May be scalar or vector.
model	Optional parametric survival model.
...	Additional parameters passed to the actuarial functions.
tbl	Optional life table object. Supply by name.

Value

Numeric vector of reserve values.

Examples

```
tVnAdotx(
  40,
  n = 20,
  t = 10,
  i = 0.05,
  model = "uniform",
  omega = 100
)
```

tVnax *Reserve for a deferred annuity-immediate*

Description

Computes the reserve during the deferral period, where $0 \leq t < n$.

Usage

```
tVnax(x, n, t, i, model = NULL, ..., tbl = NULL)
```

Arguments

x	Issue age. May be scalar or vector.
n	Positive integer deferral period. May be scalar or vector.
t	Nonnegative integer duration. May be scalar or vector.
i	Effective annual interest rate. May be scalar or vector.
model	Optional parametric survival model.
...	Additional parameters passed to the actuarial functions.
tbl	Optional life table object. Supply by name.

Value

Numeric vector of reserve values.

Examples

```
tVnax(
  40,
  n = 20,
  t = 10,
  i = 0.05,
  model = "uniform",
  omega = 100
)
```

tVnEx — *Pure endowment net level premium reserve*

Description

Computes the prospective reserve for an n-year pure endowment.

Usage

```
tVnEx(x, n, t, i, model = NULL, ..., tbl = NULL)
```

Arguments

x	Issue age.
n	Term in years.
t	Duration.
i	Effective annual interest rate.
model	Optional parametric survival model name.
...	Additional model parameters.
tbl	Optional life table object.

Value

A numeric vector of values.

Examples

```
tVnEx(40, n = 20, t = 10, i = 0.05, model = "uniform", omega = 100)
```

tVx — *Whole life net level premium reserve*

Description

Computes the prospective reserve for a whole life insurance with annual premiums: reserve at duration t equals future APV of benefits minus future APV of net premiums.

Usage

```
tVx(x, t, i, model = NULL, ..., tbl = NULL)
```

Arguments

x	Issue age.
t	Duration.
i	Effective annual interest rate.
model	Optional parametric survival model name.
...	Additional model parameters.
tbl	Optional life table object.

Value

A numeric vector of values.

Examples

```
tVx(40, t = 10, i = 0.05, model = "uniform", omega = 100)
```

---

tVxn *Endowment insurance net level premium reserve*

Description

Computes the prospective reserve for an n-year endowment insurance.

Usage

```
tVxn(x, n, t, i, model = NULL, ..., tbl = NULL)
```

Arguments

x	Issue age.
n	Term in years.
t	Duration.
i	Effective annual interest rate.
model	Optional parametric survival model name.
...	Additional model parameters.
tbl	Optional life table object.

Value

A numeric vector of values.

Examples

```
tVxn(40, n = 20, t = 10, i = 0.05, model = "uniform", omega = 100)
```

tVxn1 *Term insurance net level premium reserve*

Description

Computes the prospective reserve for an n-year term insurance.

Usage

```
tVxn1(x, n, t, i, model = NULL, ..., tbl = NULL)
```

Arguments

x	Issue age.
n	Term in years.
t	Duration.
i	Effective annual interest rate.
model	Optional parametric survival model name.
...	Additional model parameters.
tbl	Optional life table object.

Value

A numeric vector of values.

Examples

```
tVxn1(40, n = 20, t = 10, i = 0.05, model = "uniform", omega = 100)
```

tVxn1_ret *Retrospective term insurance reserve*

Description

Computes the retrospective reserve for a term insurance at a duration satisfying $0 \le t \le n$. At expiry, the reserve is 0.

Usage

```
tVxn1_ret(x, n, t, i, model = NULL, ..., tbl = NULL)
```

Arguments

x	Issue age. May be scalar or vector.
n	Nonnegative integer contract term. May be scalar or vector.
t	Nonnegative integer duration. May be scalar or vector.
i	Effective annual interest rate. May be scalar or vector.
model	Optional parametric survival model.
...	Additional parameters passed to the actuarial functions.
tbl	Optional life table object. Supply by name.

Value

Numeric vector of retrospective reserve values.

Examples

```
tVxn1_ret(
  40,
  n = 20,
  t = 10,
  i = 0.05,
  model = "uniform",
  omega = 100
)
```

---

tVxn_ret *Retrospective endowment insurance reserve*

Description

Computes the retrospective reserve for an endowment insurance at a duration satisfying $0 \leq t \leq n$. At maturity, the reserve is 1.

Usage

```
tVxn_ret(x, n, t, i, model = NULL, ..., tbl = NULL)
```

Arguments

x	Issue age. May be scalar or vector.
n	Nonnegative integer contract term. May be scalar or vector.
t	Nonnegative integer duration. May be scalar or vector.
i	Effective annual interest rate. May be scalar or vector.
model	Optional parametric survival model.
...	Additional parameters passed to the actuarial functions.
tbl	Optional life table object. Supply by name.

Value

Numeric vector of retrospective reserve values.

Examples

```
tVxn_ret(
  40,
  n = 20,
  t = 10,
  i = 0.05,
  model = "uniform",
  omega = 100
)
```

`tVx_m` *Whole life reserve with m-thly premiums*

Description

Computes the reserve for a whole life insurance funded by true m-thly premiums.

Usage

```
tVx_m(x, t, m, i, model = NULL, ..., tbl = NULL)
```

Arguments

`x`	Issue age.
`t`	Duration.
`m`	Number of premium payments per year.
`i`	Effective annual interest rate.
`model`	Optional parametric survival model name.
`...`	Additional model parameters.
`tbl`	Optional life table object.

Value

A numeric vector of values.

Examples

```
tVx_m(40, t = 10, m = 12, i = 0.05, model = "uniform", omega = 100)
```

`tVx_ret` *Retrospective whole life reserve*

Description

Computes the retrospective net level premium reserve

$$ {}_tV_x = P_x \ddot{s}_{x:\bar{t}|} - \frac{A^1_{x:\bar{t}|}}{{}_tE_x}. $$

Usage

```
tVx_ret(x, t, i, model = NULL, ..., tbl = NULL)
```

Arguments

`x`	Issue age. May be scalar or vector.
`t`	Nonnegative integer duration. May be scalar or vector.
`i`	Effective annual interest rate. May be scalar or vector.
`model`	Optional parametric survival model.
`...`	Additional parameters passed to the actuarial functions.
`tbl`	Optional life table object. Supply by name.

Details

The mortality basis may be supplied through either a life table or a parametric survival model.

Value

Numeric vector of retrospective reserve values.

Examples

```
tVx_ret(
  40,
  t = 10,
  i = 0.05,
  model = "uniform",
  omega = 100
)
```

udd_continuous_multiplier *UDD multiplier for continuous insurance approximations*

Description

Under UDD, $\bar{A}_x = (i/\delta)A_x$ and similarly for term and deferred insurance.

Usage

```
udd_continuous_multiplier(i)
```

Arguments

i	Numeric vector of effective annual interest rates.

Value

Numeric vector equal to i/δ.

udd_mthly_multiplier *UDD multiplier for m-thly insurance approximations*

Description

Under UDD, $A_x^{(m)} = (i/i^{(m)})A_x$ and similarly for term and deferred insurance.

Usage

```
udd_mthly_multiplier(i, m)
```

Arguments

i	Numeric vector of effective annual interest rates.
m	Positive integer payment frequency.

Value

Numeric vector equal to $i/i^{(m)}$.

`variable_interest_apvs` *Actuarial present values under variable annual interest rates*

Description

Computes life-contingent actuarial present values using a specified sequence of annual effective interest rates.

Usage

```
nEx_var(qx, i, benefit = 1)

Axn1_var(qx, i, benefit = 1)

Axn_var(qx, i, benefit = 1)

axn_var(qx, i, type = c("immediate", "due"), benefit = 1)
```

Arguments

`qx`	Numeric vector of one-year mortality probabilities.
`i`	Numeric vector of annual effective interest rates. Each value must be greater than `-1`.
`benefit`	Nonnegative scalar benefit or annuity payment amount.
`type`	Character string equal to `"immediate"` or `"due"`.

Details

The vectors `qx` and `i` represent one valuation scenario and must have the same positive length.

`nEx_var()` computes a pure endowment.

`Axn1_var()` computes term insurance payable at the end of the year of death.

`Axn_var()` computes endowment insurance.

`axn_var()` computes a temporary annuity-immediate or annuity-due.

Each year's payment is discounted using the cumulative product of the annual effective interest rates supplied in `i`.

The pure endowment is

$$ {}_nE = {}_np_x \, v_n, $$

where v_n is the cumulative discount factor implied by the sequence of annual effective interest rates.

Term insurance is obtained by discounting each possible death benefit using the cumulative discount factor applicable to its payment year.

Endowment insurance equals the sum of the corresponding term insurance and pure endowment.

Temporary annuities discount each payment using the cumulative discount factors derived from the interest-rate sequence.

Value

A numeric scalar.

Examples

```
qx <- c(0.03, 0.04, 0.05, 0.06, 0.07)
rates <- c(0.06, 0.07, 0.08, 0.09, 0.10)

nEx_var(qx, rates, benefit = 1000)
Axn1_var(qx, rates)
Axn_var(qx, rates)
axn_var(qx, rates, type = "due")
```

---

varLtx *Variance of present value of loss at duration t for whole life insurance*

Description

Computes the conditional variance $\mathrm{Var}({}_tL_x \mid K_x \geq t)$ for a fully discrete whole life insurance.

Usage

```
varLtx(x, t, i, P, model = NULL, ..., tbl = NULL)
```

Arguments

x	Issue age.
t	Duration.
i	Effective annual interest rate.
P	Annual premium.
model	Optional parametric survival model name.
...	Additional model parameters.
tbl	Optional life table object.

Value

A numeric vector of values.

Examples

```
prem <- Px(40, i = 0.05, model = "uniform", omega = 100)
varLtx(40, t = 10, i = 0.05, P = prem, model = "uniform", omega = 100)
```

var_Abarx *Variance of continuous whole life insurance PV*

Description

Computes $\mathrm{Var}(\bar{Z}_x) = {}^2\bar{A}_x - \bar{A}_x^2$.

Usage

```
var_Abarx(x, i, model, ...)
```

Arguments

x	Age.
i	Effective annual interest rate.
model	Parametric survival model name.
...	Additional model parameters passed to survival-model functions.

Value

Numeric vector of variances.

var_Abarxn *Variance of continuous endowment insurance PV*

Description

Variance of continuous endowment insurance PV

Usage

```
var_Abarxn(x, n, i, model, ...)
```

Arguments

x	Age.
n	Term.
i	Effective annual interest rate.
model	Parametric survival model name.
...	Additional model parameters passed to survival-model functions.

Value

Numeric vector of variances.

var_Abarxn1 *Variance of continuous term insurance PV*

Description

Computes $\mathrm{Var}(\bar{Z}^1_{x:\overline{n}|}) = {}^2\bar{A}^1_{x:\overline{n}|} - (\bar{A}^1_{x:\overline{n}|})^2$.

Usage

```
var_Abarxn1(x, n, i, model, ...)
```

Arguments

x	Age.
n	Term.
i	Effective annual interest rate.
model	Parametric survival model name.
...	Additional model parameters passed to survival-model functions.

Value

Numeric vector of variances.

var_Ax *Variance of whole life insurance PV*

Description

Computes $\mathrm{Var}(Z_x) = {}^2A_x - A_x^2$.

Usage

```
var_Ax(x, i, tbl = NULL, model = NULL, ..., tol = 1e-12, k_max = 5000)
```

Arguments

x	Age.
i	Effective annual interest rate.
tbl	Optional life table object.
model	Optional parametric survival model name.
...	Additional arguments passed to survival-model functions.
tol	Numerical tolerance for truncating infinite sums.
k_max	Maximum number of terms in the sum.

Value

Numeric vector of variances.

`var_Axn` *Variance of endowment insurance PV*

Description

Variance of endowment insurance PV

Usage

```
var_Axn(x, n, i, tbl = NULL, model = NULL, ...)
```

Arguments

`x`	Age.
`n`	Term.
`i`	Effective annual interest rate.
`tbl`	Optional life table object.
`model`	Optional parametric survival model name.
`...`	Additional arguments passed to survival-model functions.

Value

Numeric vector of variances.

`var_Axn1` *Variance of term insurance PV*

Description

Computes $\mathrm{Var}(Z^1_{x:\overline{n}|}) = {}^2A^1_{x:\overline{n}|} - (A^1_{x:\overline{n}|})^2$.

Usage

```
var_Axn1(x, n, i, tbl = NULL, model = NULL, ...)
```

Arguments

`x`	Age.
`n`	Term.
`i`	Effective annual interest rate.
`tbl`	Optional life table object.
`model`	Optional parametric survival model name.
`...`	Additional arguments passed to survival-model functions.

Value

Numeric vector of variances.

`var_Axn1_m` *Variance of m-thly term insurance PV*

Description

Variance of m-thly term insurance PV

Usage

```
var_Axn1_m(x, n, i, m, model, ...)
```

Arguments

`x`	Age.
`n`	Term.
`i`	Effective annual interest rate.
`m`	Positive integer payment frequency.
`model`	Parametric survival model name.
`...`	Additional model parameters passed to survival-model functions.

Value

Numeric vector of variances.

`var_Axn_m` *Variance of m-thly endowment insurance PV*

Description

Variance of m-thly endowment insurance PV

Usage

```
var_Axn_m(x, n, i, m, model, ...)
```

Arguments

`x`	Age.
`n`	Term.
`i`	Effective annual interest rate.
`m`	Positive integer payment frequency.
`model`	Parametric survival model name.
`...`	Additional model parameters passed to survival-model functions.

Value

Numeric vector of variances.

`var_Ax_m` *Variance of m-thly whole life insurance PV*

Description

Computes $\mathrm{Var}(Z_x^{(m)}) = {}^2A_x^{(m)} - (A_x^{(m)})^2$.

Usage

```
var_Ax_m(x, i, m, model, ..., tol = 1e-12, j_max = 100000L)
```

Arguments

`x`	Age.
`i`	Effective annual interest rate.
`m`	Positive integer payment frequency.
`model`	Parametric survival model name.
`...`	Additional model parameters passed to survival-model functions.
`tol`	Numerical tolerance for truncating the infinite sum.
`j_max`	Maximum number of m-thly intervals in the sum.

Value

Numeric vector of variances.

`var_nAbarx` *Variance of continuous deferred insurance PV*

Description

Variance of continuous deferred insurance PV

Usage

```
var_nAbarx(x, n, i, model, ...)
```

Arguments

x	Age.
n	Deferral period.
i	Effective annual interest rate.
model	Parametric survival model name.
...	Additional model parameters passed to survival-model functions.

Value

Numeric vector of variances.

---

var_nAx *Variance of deferred insurance PV*

Description

Variance of deferred insurance PV

Usage

```
var_nAx(x, n, i, tbl = NULL, model = NULL, ..., tol = 1e-12, k_max = 5000)
```

Arguments

x	Age.
n	Deferral period.
i	Effective annual interest rate.
tbl	Optional life table object.
model	Optional parametric survival model name.
...	Additional arguments passed to survival-model functions.
tol	Numerical tolerance for truncating infinite sums.
k_max	Maximum number of terms in the sum.

Value

Numeric vector of variances.

var_nAx_m *Variance of m-thly deferred insurance PV*

Description

Variance of m-thly deferred insurance PV

Usage

```
var_nAx_m(x, n, i, m, model, ..., tol = 1e-12, j_max = 100000L)
```

Arguments

x	Age.
n	Deferral period.
i	Effective annual interest rate.
m	Positive integer payment frequency.
model	Parametric survival model name.
...	Additional model parameters passed to survival-model functions.
tol	Numerical tolerance for truncating the infinite sum.
j_max	Maximum number of m-thly intervals in the sum.

Value

Numeric vector of variances.

var_nEx *Variance of pure endowment PV*

Description

Variance of pure endowment PV

Usage

```
var_nEx(x, n, i, tbl = NULL, model = NULL, ...)
```

Arguments

x	Age.
n	Term.
i	Effective annual interest rate.
tbl	Optional life table object.
model	Optional parametric survival model name.
...	Additional arguments passed to survival-model functions.

Value

Numeric vector of variances.

`Vprefloor_crvm_ul` *Pre-floor CRVM reserve*

Description

Computes the pre-floor reserve by multiplying the funding ratio by the difference between the present value of future benefits and future premiums.

Usage

```
Vprefloor_crvm_ul(r, pvfb_minus_pvfp)
```

Arguments

`r`	Funding ratio. Values must lie in `[0, 1]`.
`pvfb_minus_pvfp`	Numeric difference between the present value of future benefits and future premiums.

Value

A numeric vector.

Examples

```
Vprefloor_crvm_ul(r = 0.33506, pvfb_minus_pvfp = 70)
```

`vt_var` *Discount factors under variable annual interest rates*

Description

Computes cumulative discount factors for a sequence of annual effective interest rates:

$$v_t = \prod_{k=1}^{t} (1 + i_k)^{-1}, \qquad t = 1, \ldots, n.$$

Usage

```
vt_var(i)
```

Arguments

i	Numeric vector of annual effective interest rates. Each value must be greater than -1.

Value

A numeric vector of cumulative discount factors with the same length as i.

Examples

```
vt_var(c(0.06, 0.07, 0.08))
vt_var(c(-0.01, 0.02, 0.03))
```

V_zeroized *Zeroized reserves for a discrete death-benefit contract*

Description

Computes reserves backward by setting expected profit in each policy year equal to zero. Negative reserves may optionally be floored at zero.

Usage

```
V_zeroized(qx, i, G, benefit, r = 0, e = 0, V_terminal = 0, floor_zero = TRUE)
```

Arguments

qx	Mortality probability by policy year.
i	Annual effective interest rate by policy year. Values must be greater than -1.
G	Gross premium by policy year.
benefit	Death benefit by policy year.
r	Percent-of-premium expense rate by policy year. Values must lie in [0, 1].
e	Fixed expense by policy year.
V_terminal	Nonnegative scalar terminal reserve.
floor_zero	Logical scalar. If TRUE, negative reserves are replaced by zero.

Value

A named numeric vector of length length(qx) + 1.

Examples

```
V_zeroized(
  qx = c(0.015, 0.017, 0.019, 0.021, 0.024),
  i = 0.06,
  G = 19279,
  benefit = 1000000,
  e = 240
)
```

`z_from_coupon_annual` *Bootstrap annual effective spot rates*

Description

Bootstraps annual effective spot rates from par coupon yields at consecutive integer maturities.

Usage

```
z_from_coupon_annual(maturity, coupon_yield, par = 1000)
```

Arguments

`maturity`	Numeric vector of positive integer maturities in strictly increasing order. Maturities must be consecutive and begin at 1.
`coupon_yield`	Numeric vector of annual effective par coupon yields. Values must be greater than `-1`.
`par`	Positive scalar par value.

Value

A numeric vector of annual effective spot rates.

Examples

```
maturity <- 1:4
coupon_yield <- c(0.02, 0.04, 0.06, 0.08)
z_from_coupon_annual(maturity, coupon_yield)
```

z_from_coupon_semi *Bootstrap semiannual nominal spot rates*

Description

Bootstraps nominal annual spot rates convertible semiannually from par coupon yields at consecutive half-year maturities.

Usage

```
z_from_coupon_semi(maturity, coupon_yield, par = 1000)
```

Arguments

maturity	Numeric vector of positive maturities in years, in strictly increasing order. Maturities must be consecutive multiples of 0.5.
coupon_yield	Numeric vector of nominal annual par coupon yields convertible semiannually. Values must be greater than -2.
par	Positive scalar par value.

Value

A numeric vector of nominal annual spot rates convertible semiannually.

Examples

```
maturity <- c(0.5, 1.0, 1.5, 2.0)
coupon_yield <- c(0.0244, 0.0260, 0.0276, 0.0293)
z_from_coupon_semi(maturity, coupon_yield)
```

z_from_fn1 *Spot rates from one-year forward rates*

Description

Converts annual effective one-year forward rates $f_{0,1}, f_{1,1}, \ldots, f_{n-1,1}$ into annual effective spot rates:

$$(1 + z_n)^n = \prod_{j=0}^{n-1} (1 + f_{j,1}).$$

Usage

```
z_from_fn1(fn1)
```

Arguments

fn1	Numeric vector of annual effective one-year forward rates. Each value must be greater than -1.

Value

A numeric vector of annual effective spot rates.

Examples

```
z_from_fn1(c(0.04, 0.05, 0.06, 0.07, 0.08))
```

Index