Package ‘valuer’

January 3, 2017

Type Package

Title Pricing of Variable Annuities

Version 1.1.1

Author Ivan Zoccolan [aut, cre]

Maintainer Ivan Zoccolan <ivan.zoccolan@gmail.com>

Description Pricing of variable annuity life insurance
contracts by means of Monte Carlo methods. Monte Carlo is used to price
the contract in case the policyholder cannot surrender while
Least Squares Monte Carlo is used if the insured can surrender.
This package implements the pricing framework and algorithm described in
Bacinello et al. (2011) <doi:10.1016/j.insmatheco.2011.05.003>.
It also implements the state-dependent fee structure
discussed in Bernard et al. (2014) <doi:10.1017/asb.2014.13>.

License GPL-3

LazyData TRUE

RoxygenNote 5.0.1

Depends R (>= 3.2.5), orthopolynom (>= 1.0-5)

Imports R6 (>= 2.1.2), RcppEigen (>= 0.3.2.8.1), timeDate (>=
3012.100), yuima (>= 1.1.6), ggplot2

Suggests testthat, knitr, rmarkdown, doParallel (>= 1.0.10), foreach
(>= 1.4.3)

LinkingTo Rcpp

VignetteBuilder knitr

URL http://github.com/IvanZoccolan/valuer

BugReports http://github.com/IvanZoccolan/valuer/issues

NeedsCompilation yes

Repository CRAN

Date/Publication 2017-01-03 16:32:47

R topics documented:

`calc_account` *Calculates the account*

Description

Calculates the account

Usage

```
calc_account(spot, ben, fee, barrier, penalty)
```

Arguments

spot	numeric vector with the VA reference fund values
ben	numeric vector with the living benefit cash flow
fee	numeric scalar with the fee
barrier	numeric scalar with the state-dependent barrier
penalty	numeric vector with the surrender penalty

---

constant_parameters *Constant parameter class*

Description

Class providing a constant parameter object with methods to calculate the integral of the parameter and the squared parameter over a time span.

Usage

```
constant_parameters
```

Format

R6Class object.

Value

Object of R6Class

Methods

- **integral** Calculates the integral given the initial and final times. The arguments are two timeDate object with the initial and final times. It returns a numeric scalar with the integral
- **integral_square (public)** Calculates the integral of the squared constant parameter given the initial and final times. The arguments are two timeDate object with the initial and final times. It returns a numeric scalar with the integral
- **get (public)** get the constant

Examples

```
r <- constant_parameters$new(0.01)
#Over the full year (365 days) the integral should evaluate to 0.01
r$integral(timeDate::timeDate("2016-07-09"), timeDate::timeDate("2017-07-09"))
#Over the full year the integral square should evaluate to 0.001
r$integral_square(timeDate::timeDate("2016-07-09"), timeDate::timeDate("2017-07-09"))
```

data_gatherer *Simple data gatherer*

Description

Class which defines a simple data gatherer to hold estimates calculated in a loop.

Usage

```
data_gatherer
```

Format

`R6Class` object.

Value

Object of `R6Class`

Methods

`new` Constructor method

`dump_result` Saves the argument `result` which is a numeric scalar

`get_results` Returns a numeric vector with the point estimates.

financials_BBM2010 *BBM2010 financial processes*

Description

List of parameters to initialize a va_sde_engine object to simulate the interest rate, volatility and log price processes according to the stochastic differential equations specified in BBM2010 - See **References**.

Usage

```
financials_BBM2010
```

Format

A list with elements:

[[1]] List of parameters for `simulate`

[[2]] List of parameters for `setModel`

[[3]] Vector with indices indicating the interest rate and log price in solve.variable `setModel`

References

BBM2010 *Bacinello A.R., Biffis E. e Millossovich P. "Regression-based algorithms for life insurance contracts with surrender guarantees". In: Quantitative Finance 10.9 (2010), pp. 1077-1090.*

`financials_BMOP2011` *BMOP2011 financial processes*

Description

List of parameters to initialize a va_sde_engine object to simulate the interest rate, volatility and log price processes according to the stochastic differential equations specified in BMOP2011 - See **References**.

Usage

```
financials_BMOP2011
```

Format

A list with elements:

[[1]] List of parameters for `simulate`

[[2]] List of parameters for `setModel`

[[3]] Vector with indices indicating the interest rate and log price in solve.variable `setModel`

References

BMOP2011 *Bacinello A.R., Millossovich P., Olivieri A. e Pitacco E. "Variable annuities: a unifying valuation approach." In: Insurance: Mathematics and Economics 49 (2011), pp. 285-297.*

`financials_BZ2016` *BZ2016 financial processes*

Description

List of parameters to initialize a `va_sde_engine2` object to simulate the interest rate and log price processes being the volatility constant. The interest rate and fund processes follow the stochastic differential equations specified in BMOP2011 - See **References**. The volatility is constant with default value 0.2

Usage

```
financials_BZ2016
```

Format

A list with elements:

[[1]] List of parameters for simulate

[[2]] List of parameters for setModel

[[3]] Vector with indices indicating the interest rate and log price in solve.variable setModel

References

BMOP2011 *Bacinello A.R., Millossovich P., Olivieri A. e Pitacco E. "Variable annuities: a unifying valuation approach." In: Insurance: Mathematics and Economics 49 (2011), pp. 285-297.*

Examples

```
#Sets the constant volatility to 0.3
financials_BZ2016[[1]]$K <- 0.3 ^ 2
```

---

financials_BZ2016bis *BZ2016bis financial processes*

Description

List of parameters to initialize a va_sde_engine3 object to simulate the log price and volatility processes which follow the stochastic differential equations specified in BMOP2011 - See **References**. The interest rate is constant with default value 0.03.

Usage

```
financials_BZ2016bis
```

Format

A list with elements:

[[1]] List of parameters for simulate

[[2]] List of parameters for setModel

[[3]] Vector with indices indicating the log price in solve.variable setModel

References

BMOP2011 *Bacinello A.R., Millossovich P., Olivieri A. e Pitacco E. "Variable annuities: a unifying valuation approach." In: Insurance: Mathematics and Economics 49 (2011), pp. 285-297.*

Examples

```
#Sets the interest rate to 2%
financials_BZ2016bis[[1]]$r <- 0.02
```

GMAB *Variable Annuity with GMAB guarantee*

Description

Class for VA with Guaranteed Minimum Accumulation Benefit (GMAB). It supports a simple state-dependent fee structure with a single barrier.
See **References** for a description of variable annuities life insurance products, their guarantees and fee structures.

Usage

GMAB

Format

R6Class object.

Value

Object of R6Class

Methods

new Constructor method with arguments:

- payoff payoff object of the GMAB guarantee
- t0 timeDate object with the issue date of the contract
- t timeDate object with the end date of the accumulation period
- t1 timeDate object with the end date of the life benefit payment
- age numeric positive scalar with the age of the policyholder
- fee constant_parameters object with the fee
- barrier numeric positive scalar with the state-dependent fee barrier
- penalty penalty_class object with the penalty

get_times get method for the product time-line. Returns a timeDate object

get_age get method for the age of the insured

set_age set method for the age of the insured

get_barrier get method for the state-dependent fee barrier. Returns a positive scalar with the barrier

set_barrier set method for the state-dependent fee barrier. Argument must be a positive scalar.

set_penalty_object the argument penalty is a penalty_class object which is stored in a private field.

get_penalty_object gets the penalty_class object.

set_penalty set method for the penalty applied in case of surrender. The argument must be a scalar between 0 and 1.

get_penalty get method for the surrender penalties. It can be a scalar between 0 and 1 in case the penalty is constant or a numeric vector in case the penalty varies with time.

set_fee set method for the contract fee. The argument is a constant_parameters object with the fee.

set_payoff set method for the payoff_guarantee object.

survival_benefit_times returns a numeric vector with the survival benefit time indexes.

surrender_times returns a numeric vector with the surrender time indexes. Takes as argument a string with the frequency of the decision if surrendering the contract, e.g. "3m" corresponds to a surrender decision taken every 3 months.

times_in_yrs returns the product time-line in fraction of year

cash_flows returns a numeric vector with the cash flows of the product. It takes as argument spot_values a numeric vector which holds the values of the underlying fund and death_time a time index with the time of death

survival_benefit Returns a numeric scalar corresponding to the survival benefit. The arguments are spot_values vector which holds the values of the underlying fund and t the time index of the survival benefit.

get_premium Returns the premium as non negative scalar

References

BMOP2011 *Bacinello A.R., Millossovich P., Olivieri A., Pitacco E., "Variable annuities: a unifying valuation approach." In: Insurance: Mathematics and Economics 49 (2011), pp. 285-297.*

BHM2014 *Bernard C., Hardy M. and Mackay A. "State-dependent fees for variable annuity guarantees." In: Astin Bulletin 44 (2014), pp. 559-585.*

Examples

```
#Sets up the payoff as a roll-up of premiums with roll-up rate 1%

rate <- constant_parameters$new(0.01)

premium <- 100
rollup <- payoff_rollup$new(premium, rate)

#Five years time-line
begin <- timeDate::timeDate("2016-01-01")
end <- timeDate::timeDate("2020-12-31")

age <- 60
# A constant fee of 2% per year (365 days)
fee <- constant_parameters$new(0.02)

#Barrier for a state-dependent fee. The fee will be applied only if
#the value of the account is below the barrier
barrier <- 200

#Withdrawal penalty applied in case the insured surrenders the contract
#It is a constant penalty in this case
```

```
penalty <- penalty_class$new(type = 1, 0.01)


#Sets up a VA contract with GMAB guarantee. The guaranteed miminum
#is the roll-up of premiums with rate 1%
contract <- GMAB$new(rollup, t0 = begin, t = end, age = age,  fee = fee,
barrier = barrier, penalty = penalty)
```

`GMAB_GMDB` *Variable Annuity with GMAB and GMDB guarantees*

Description

Class for a VA with Guaranteed Minimum Accumulation Benefit (GMAB) and Guaranteed Minimun Accumulation Benefit (GMDB). It supports a simple state-dependent fee structure with a single barrier.
See **References** for a description of variable annuities life insurance products, their guarantees and fee structures.

Usage

```
GMAB_GMDB
```

Format

`R6Class` object.

Value

Object of `R6Class`

Methods

`new` Constructor method with arguments:

- `payoff` `payoff` object of the GMAB guarantee
- `t0` `timeDate` object with the issue date of the contract
- `t` `timeDate` object with the end date of the accumulation period
- `t1` `timeDate` object with the end date of the life benefit payment
- `age` `numeric` positive scalar with the age of the policyholder
- `fee` `constant_parameters` object with the fee
- `barrier` `numeric` positive scalar with the state-dependent fee barrier
- `penalty` `penalty_class` object with the penalty
- `death_payoff` `payoff` object with the payoff of the GMDB guarantee

`get_times` get method for the product time-line. Returns a `timeDate` object

`get_age` get method for the age of the insured

`set_age` set method for the age of the insured get_barrier get method for the state-dependent fee barrier. Returns a positive scalar with the barrier

set_barrier set method for the state-dependent fee barrier. Argument must be a positive scalar.

set_penalty_object the argument penalty is a penalty_class object which is stored in a private field.

get_penalty_object gets the penalty_class object.

set_penalty set method for the penalty applied in case of surrender. The argument must be a scalar between 0 and 1.

get_penalty get method for the surrender penalties. It can be a scalar between 0 and 1 in case the penalty is constant or a numeric vector in case the penalty varies with time.

set_fee set method for the contract fee. The argument is a constant_parameters object with the fee.

set_payoff set method for the payoff_guarantee object of the GMAB rider

set_death_payoff set method for the payoff_guarantee object of the GMDB rider

survival_benefit_times returns a numeric vector with the survival benefit time indexes.

surrender_times returns a numeric vector with the surrender time indexes. Takes as argument a string with the frequency of the decision if surrendering the contract, e.g. "3m" corresponds to a surrender decision taken every 3 months.

times_in_yrs returns the product time-line in fraction of year

cash_flows returns a numeric vector with the cash flows of the product. It takes as argument spot_values a numeric vector which holds the values of the underlying fund and death_time a time index with the time of death

survival_benefit Returns a numeric scalar corresponding to the survival benefit. The arguments are spot_values vector which holds the values of the underlying fund and t the time index of the survival benefit.

get_premium Returns the premium as non negative scalar

References

BMOP2011 *Bacinello A.R., Millossovich P., Olivieri A., Pitacco E., "Variable annuities: a unifying valuation approach." In: Insurance: Mathematics and Economics 49 (2011), pp. 285-297.*

BHM2014 *Bernard C., Hardy M. and Mackay A. "State-dependent fees for variable annuity guarantees." In: Astin Bulletin 44 (2014), pp. 559-585.*

Examples

```
#Sets up the payoff as a roll-up of premiums with roll-up rate 1%

rate <- constant_parameters$new(0.01)

premium <- 100
rollup <- payoff_rollup$new(premium, rate)

#Five years time-line
begin <- timeDate::timeDate("2016-01-01")
```

```
end <- timeDate::timeDate("2020-12-31")
#Age of the insured
age <- 60
# A constant fee of 2% per year (365 days)
fee <- constant_parameters$new(0.02, 365)

#Barrier for a state-dependent fee. The fee will be applied only if
#the value of the account is below the barrier
barrier <- 200

#Withdrawal penalty applied in case the insured surrenders the contract
#It is a constant penalty in this case
penalty <- penalty_class$new(type = 1, 0.01)

#Sets up the GMAB + GMDB with the same payoff for survival and death
#benefits
contract <- GMAB_GMDB$new(rollup, t0 = begin, t = end, age = age, fee =fee,
barrier = barrier, penalty = penalty, death_payoff = rollup)
```

GMDB *Variable Annuity with GMDB guarantee*

Description

Class for VA with Guaranteed Minimum Death Benefit (GMDB). It supports a simple state-dependent fee structure with a single barrier.
See **References** for a description of variable annuities life insurance products, their guarantees and fee structures.

Usage

```
GMDB
```

Format

`R6Class` object.

Value

Object of `R6Class`

Methods

`new` Constructor method with arguments:

- `payoff` `payoff` object of the GMDB guarantee
- `t0` `timeDate` object with the issue date of the contract
- `t` `timeDate` object with the end date of the accumulation period
- `t1` `timeDate` object with the end date of the life benefit payment

age numeric positive scalar with the age of the policyholder

fee constant_parameters object with the fee

barrier numeric positive scalar with the state-dependent fee barrier

penalty penalty_class object with the penalty

get_times get method for the product time-line. Returns a timeDate object

get_age get method for the age of the insured

set_age set method for the age of the insured

get_barrier get method for the state-dependent fee barrier. Returns a positive scalar with the barrier

set_barrier set method for the state-dependent fee barrier. Argument must be a positive scalar.

set_penalty_object the argument penalty is a penalty_class object which is stored in a private field.

get_penalty_object gets the penalty_class object.

set_penalty set method for the penalty applied in case of surrender. The argument must be a scalar between 0 and 1.

get_penalty get method for the surrender penalties. It can be a scalar between 0 and 1 in case the penalty is constant or a numeric vector in case the penalty varies with time.

set_fee set method for the contract fee. The argument is a constant_parameters object with the fee.

survival_benefit_times returns a numeric vector with the survival benefit time indexes.

surrender_times returns a numeric vector with the surrender time indexes. Takes as argument a string with the frequency of the decision if surrendering the contract, e.g. "3m" corresponds to a surrender decision taken every 3 months.

times_in_yrs returns the product time-line in fraction of year

cash_flows returns a numeric vector with the cash flows of the product. It takes as argument spot_values a numeric vector which holds the values of the underlying fund and death_time a time index with the time of death

survival_benefit Returns a numeric scalar corresponding to the survival benefit. The arguments are spot_values vector which holds the values of the underlying fund and t the time index of the survival benefit.

get_premium Returns the premium as non negative scalar

References

BMOP2011 *Bacinello A.R., Millossovich P., Olivieri A., Pitacco E., "Variable annuities: a unifying valuation approach." In: Insurance: Mathematics and Economics 49 (2011), pp. 285-297.*

BHM2014 *Bernard C., Hardy M. and Mackay A. "State-dependent fees for variable annuity guarantees." In: Astin Bulletin 44 (2014), pp. 559-585.*

Examples

```
#Sets up the payoff as a roll-up of premiums with roll-up rate 2%

rate <- constant_parameters$new(0.02)

premium <- 100
rollup <- payoff_rollup$new(premium, rate)

begin <- timeDate::timeDate("2016-01-01")
end <- timeDate::timeDate("2020-12-31")

age <- 60
# A constant fee of 0.02% per year (365 days)
fee <- constant_parameters$new(0.02)

#Barrier for a state-dependent fee. The fee will be applied only if
#the value of the account is below the barrier
barrier <- Inf

#Withdrawal penalty applied in case the insured surrenders the contract
#It is a constant penalty in this case
penalty <- penalty_class$new(type = 1, 0.01)

#Sets up a VA contract with GMDB guarantee. The guaranteed miminum
#is the roll-up of premiums with rate 2%

contract <- GMDB$new(rollup, t0 = begin, t = end, age = age,  fee = fee,
barrier = barrier, penalty = penalty)
```

GMIB *Variable Annuity with GMIB guarantee*

Description

Class for VA with Guaranteed Minimum Income Benefit (GMIB). A GMIB rider provides a lifetime annuity from a specified future time. Types of GMIB supported are a whole-life annuity (Ia), an annuity-certain (Ib) or annuity-certain followed by a deferred whole-life annuity (Ic). It supports a simple state-dependent fee structure with a single barrier.
See **References** for a description of variable annuities life insurance products, their guarantees and fee structures.

Usage

```
GMIB
```

Format

R6Class object.

Details

The annuity payment is assumed to be annual and it's calculated as the annuitization rate by the roll-up or ratchet payoff at the end of the accumulation period `t`.

Value

Object of `R6Class`

Methods

`new` Constructor method with arguments:

- `payoff` `payoff` object of the GMAB guarantee
- `t0` `timeDate` object with the issue date of the contract
- `t` `timeDate` object with the end date of the accumulation period
- `t1` `timeDate` object with the end date of the life benefit payment
- `age` `numeric` positive scalar with the age of the policyholder
- `fee` `constant_parameters` object with the fee
- `barrier` `numeric` positive scalar with the state-dependent fee barrier
- `penalty` `penalty_class` object with the penalty
- `eta` `numeric` scalar with the market annuitisation rate
- `type` `string` with the income benefit type: it can be 'Ia' for a whole-life annuity, 'Ib' for an annuity-certain with maturity t1, 'Ic' for an annuity certain with maturity t1 followed with a deferred life-annuity if the insured is alive after t1.

`get_times` get method for the product time-line. Returns a `timeDate` object

`get_age` get method for the age of the insured

`set_age` set method for the age of the insured

`get_barrier` get method for the state-dependent fee barrier. Returns a positive scalar with the barrier

`set_barrier` set method for the state-dependent fee barrier. Argument must be a positive scalar.

`set_penalty_object` the argument `penalty` is a `penalty_class` object which is stored in a private field.

`get_penalty_object` gets the `penalty_class` object.

`set_penalty` set method for the penalty applied in case of surrender. The argument must be a scalar between 0 and 1.

`get_penalty` get method for the surrender penalties. It can be a scalar between 0 and 1 in case the penalty is constant or a numeric vector in case the penalty varies with time.

`set_fee` set method for the contract fee. The argument is a `constant_parameters` object with the fee.

`set_payoff` set method for the `payoff_guarantee` object.

`survival_benefit_times` returns a `numeric` vector with the survival benefit time indexes.

`surrender_times` returns a `numeric` vector with the surrender time indexes. Takes as argument a string with the frequency of the decision if surrendering the contract, e.g. "3m" corresponds to a surrender decision taken every 3 months.

`times_in_yrs` returns the product time-line in fraction of year

`cash_flows` returns a `numeric` vector with the cash flows of the product. It takes as argument `spot_values` a `numeric` vector which holds the values of the underlying fund, `death_time` a time index with the time of death and `discounts` a `numeric` vector with the discount factors at time of death. These latest are used to calculate the death benefit for type Ib and Ic.

`survival_benefit` Returns a numeric scalar corresponding to the survival benefit. The arguments are `spot_values` vector which holds the values of the underlying fund and `time` the time index of the survival benefit. The function will return 0 if there's no survival benefit at the specified time

`get_premium` Returns the premium as non negative scalar

References

BMOP2011 *Bacinello A.R., Millossovich P., Olivieri A., Pitacco E., "Variable annuities: a unifying valuation approach." In: Insurance: Mathematics and Economics 49 (2011), pp. 285-297.*

BHM2014 *Bernard C., Hardy M. and Mackay A. "State-dependent fees for variable annuity guarantees." In: Astin Bulletin 44 (2014), pp. 559-585.*

Examples

```
#Sets up the payoff as a roll-up of premiums with roll-up rate 1%

rate <- constant_parameters$new(0.01)

premium <- 100
rollup <- payoff_rollup$new(premium, rate)


t0 <- timeDate::timeDate("2016-01-01")

#Five year accumulation period
t <- timeDate::timeDate("2020-12-31")

#Five year annuity certain period
t1 <- timeDate::timeDate("2025-12-31")

age <- 60

# A constant fee of 2% per year (365 days)
fee <- constant_parameters$new(0.02)

#Barrier for a state-dependent fee. The fee will be applied only if
#the value of the account is below the barrier
barrier <- 200

#Withdrawal penalty applied in case the insured surrenders the contract
#It is a constant penalty in this case
penalty <- penalty_class$new(type = 1, 0.01)

#Sets up a VA contract with GMIB guarantee, whole-life (Ia).
```

```
contract <- GMIB$new(rollup, t0 = t0, t = t, age = age,  fee = fee,
barrier = barrier, penalty = penalty, eta = 0.04)

#Sets up a VA contract with GMIB gurantee annuity-certain with
#maturity t1
contract <- GMIB$new(rollup, t0 = t0, t = t, t1 = t1,  age = age,
fee = fee, barrier = barrier, penalty = penalty, eta = 0.04, type = "Ib")
```

GMWB *Variable Annuity with GMWB guarantee*

Description

Class for a VA product with Guaranteed Minimum Withdrawal Benefit (GMWB). A GMWB rider allows for periodic withdrawals from the policy account. Types of GMWB supported are withdrawals up to a fixed date independent of survival (Wa), withdrawals up to fixed date only if the insured is alive (Wb) or whole life withdrawals (Wc). It supports a simple state-dependent fee structure with a single barrier.
See **References** for a description of variable annuities life insurance products, their guarantees and fee structures.

Usage

```
GMWB
```

Format

`R6Class` object.

Value

Object of `R6Class`

Methods

`new` Constructor method with arguments:

- `payoff` `payoff_GMWB` object with the amount of the periodic withdrawal
- `t0` `timeDate` object with the issue date of the contract
- `t1` `timeDate` object with the end date of the contract
- `age` `numeric` positive scalar with the age of the policyholder
- `fee` `constant_parameters` object with the fee
- `barrier` `numeric` positive scalar with the state-dependent fee barrier
- `penalty` `penalty_class` object with the penalty
- `type` `string` with the GMWB contract type: it can be `'Wa'` for withdrawals up to `t1` independent of survival,`'Wb'` for withdrawals up to `t1` only if the insured is alive, `'Wc'` for whole life withdrawals.

freq string with the frequency of withdrawals expressed in months (e.g. '12m' stands for yearly withdrawals).

get_times get method for the product time-line. Returns a timeDate object

get_age get method for the age of the insured

set_age set method for the age of the insured

get_barrier get method for the state-dependent fee barrier. Returns a positive scalar with the barrier

set_barrier set method for the state-dependent fee barrier. Argument must be a positive scalar.

set_penalty_object the argument penalty is a penalty_class object which is stored in a private field.

get_penalty_object gets the penalty_class object.

set_penalty set method for the penalty applied in case of surrender. The argument must be a scalar between 0 and 1.

get_penalty get method for the surrender penalties. It can be a scalar between 0 and 1 in case the penalty is constant or a numeric vector in case the penalty varies with time.

set_fee set method for the contract fee. The argument is a constant_parameters object with the fee.

set_payoff set method for the payoff_guarantee object.

survival_benefit_times returns a numeric vector with the survival benefit time indexes.

surrender_times returns a numeric vector with the surrender time indexes. Takes as argument a string with the frequency of the decision if surrendering the contract, e.g. "3m" corresponds to a surrender decision taken every 3 months.

times_in_yrs returns the product time-line in fraction of year

cash_flows returns a numeric vector with the cash flows of the product. It takes as argument: spot_values a numeric vector which holds the values of the underlying fund, death_time a time index with the time of death and discounts a numeric vector with the discount factors at time of death. These latest are used to calculate the death benefit for the GMWB of type Wa

survival_benefit Returns a numeric scalar corresponding to the survival benefit. The arguments are: spot_values vector which holds the values of the underlying fund, death_time time index of the time of death and time the time index of the survival benefit. The function will return 0 if there's no survival benefit at the specified time

get_premium Returns the premium as non negative scalar

References

BMOP2011 *Bacinello A.R., Millossovich P., Olivieri A., Pitacco E., "Variable annuities: a unifying valuation approach." In: Insurance: Mathematics and Economics 49 (2011), pp. 285-297.*

BHM2014 *Bernard C., Hardy M. and Mackay A. "State-dependent fees for variable annuity guarantees." In: Astin Bulletin 44 (2014), pp. 559-585.*

Examples

```
#Sets up the periodic payment.

premium <- 100
beta <- 0.1
GMWB_payment <- payoff_GMWB$new(premium, beta)

#Issue date of the contract
t0 <- timeDate::timeDate("2016-01-01")

#Ten years expiration of the guarantee

t1 <- timeDate::timeDate("2025-12-31")

age <- 60

# A constant fee of 2% per year (365 days)
fee <- constant_parameters$new(0.02)

#Barrier for a state-dependent fee. The fee will be applied only if
#the value of the account is below the barrier
barrier <- 200

#Withdrawal penalty applied in case the insured surrenders the contract
#It is a constant penalty in this case
penalty <- penalty_class$new(type = 1, 0.01)

#Sets up a VA contract with GMWB guarantee type Wa with yearly
#withdrawals for 10 years.

contract <- GMWB$new(GMWB_payment, t0 = t0, t1 = t1, age = age,  fee = fee,
barrier = barrier, penalty = penalty, type = "Wa", freq = "12m")
```

`makeham` *Makeham's intensity of mortality*

Description

Makeham's intensity of mortality

Usage

```
makeham(t, x, A, B, c)
```

Arguments

`t`	time as numeric scalar
`x`	age as numeric scalar

`A`	numeric scalar
`B`	numeric scalar
`c`	numeric scalar

`mc_gatherer` *Monte Carlo gatherer*

Description

Class which defines a gatherer for the Monte Carlo simulated values. It has methods to return the Monte Carlo estimate and Monte Carlo Standard Error of the estimate as well as a convergence table.

Usage

```
mc_gatherer
```

Format

`R6Class` object.

Value

Object of `R6Class`

Methods

`new` Constructor method

`dump_result` Saves the argument `result` which is a numeric scalar

`get_results` Returns the Monte Carlo estimate and the (estimated) Monte Carlo Standard Error of the estimate

`convergence_table` Returns the convergence table

`plot` Plots a Monte Carlo convergence graph at 95% level

mortality_BBM2010 *BBM2010 demographic processes*

Description

List of parameters to initialize a va_sde_engine object to simulate the intensity of mortality process according to the stochastic differential equation specified in BBM2010 - See **References**.

Usage

```
mortality_BBM2010
```

Format

A list with elements:

[[1]] List of parameters for `simulate`

[[2]] List of parameters for `setModel`

[[3]] Vector with indices indicating the intensity of mortality in solve.variable `setModel`

References

BBM2010 *Bacinello A.R., Biffis E. e Millossovich P. "Regression-based algorithms for life insurance contracts with surrender guarantees". In: Quantitative Finance 10.9 (2010), pp. 1077-1090.*

mortality_BMOP2011 *BMOP2011 demographic processes*

Description

List of parameters to initialize a va_sde_engine object to simulate the intensity of mortality process according to the stochastic differential equation specified in BMOP2011 - See **References**.

Usage

```
mortality_BMOP2011
```

Format

A list with elements:

[[1]] List of parameters for `simulate`

[[2]] List of parameters for `setModel`

[[3]] Vector with indices indicating the intensity of mortality in solve.variable `setModel`

References

BMOP2011 *Bacinello A.R., Millossovich P., Olivieri A. e Pitacco E. "Variable annuities: a unifying valuation approach." In: Insurance: Mathematics and Economics 49 (2011), pp. 285-297.*

`mu` *Weibull intensity of mortality*

Description

Weibull intensity of mortality

Usage

```
mu(t, x, c1, c2)
```

Arguments

`t`	time as numeric scalar
`x`	age as numeric scalar
`c1`	numeric scalar
`c2`	numeric scalar

`payoff_GMWB` *GMWB payoff class*

Description

Class providing the periodic payment of a GMWB contract. It is stated as a given percentage of the premium.

Usage

```
payoff_GMWB
```

Format

`R6Class` object.

Value

Object of `R6Class`

Methods

- `new` Initialize method. The arguments are a non negative scalar with the premium and a scalar between 0 and 1 with the percentage.
- `set_premium` Stores the premium in a private field. The argument is a non negative scalar
- `get_premium` Returns the premium as non negative scalar
- `set_beta` Sets the percentage. The argument is a scalar between 0 and 1
- `get_beta` Gets the percentage
- `get_payoff` Gets the payoff

Examples

```
premium <- 100
beta <- 0.15
GMWB_payment <- payoff_GMWB$new(premium, beta)
GMWB_payment$get_payoff()
```

---

payoff_guarantee *Generic guarantee payoff class*

Description

Class providing an interface for guarantee payoff objects. This class shouldn't be instantiated but used as base class for more specialized implementations such as a roll-up or ratchet payoff classes.

Usage

```
payoff_guarantee
```

Format

`R6Class` object.

Value

Object of `R6Class`

Methods

- `new (public)` Initialize method. The argument is a non negative scalar with the premium.
- `set_premium (public)` Stores the premium in a private field. The argument is a non negative scalar
- `get_premium (public)` Returns the premium as non negative scalar
- `get_payoff (public)` Gets a zero payoff in this base class.The arguments are a `numeric` vector with the amounts and a vector of `timeDate` objects to calculate the payoff

`payoff_ratchet` *Ratchet payoff class*

Description

Class providing a ratchet payoff object. The payoff will be the highest account value recorded at some specified times.

Usage

```
payoff_ratchet
```

Format

`R6Class` object.

Value

Object of `R6Class`

Methods

`new` Initialize method. The arguments are a non negative scalar with the premium and the ratchet frequency. Allowed units for the frequency are "m" for 4 weeks, "w" for weeks, "d" for days

`set_premium` Stores the premium in a private field. The argument is a non negative scalar

`get_premium` Returns the premium as non negative scalar

`set_freq` Sets the ratchet frequency. Allowed units for the frequency are "m" for 4 weeks, "w" for weeks, "d" for days

`get_payoff` Gets the payoff. The arguments are a `numeric` vector with the amounts, a vector of `timeDate` objects with the start and end dates for the ratchet and a `numeric` vector with the account values. (see **Examples**)

Examples

```
freq <- "1m"
premium <- 100
ratchet <- payoff_ratchet$new(premium, freq)
t1 <- timeDate::timeDate("2016-01-01")
t2 <- timeDate::timeDate("2016-12-31")
account <- 120 * rnorm(365)
ratchet$get_payoff(c(120,100), c(t1,t2), account)
```

payoff_rollup *Roll-up of premiums payoff class*

Description

Class providing a roll-up of premium payoff object. The payoff is the maximum between the account value and the roll-up of the premium at a given rate.

Usage

```
payoff_rollup
```

Format

`R6Class` object.

Value

Object of `R6Class`

Methods

- `new` Initialize method. The arguments are a non negative scalar with the premium and a `constant_parameters` object with the roll-up rate.
- `set_premium` Stores the premium in a private field. The argument is a non negative scalar
- `get_premium` Returns the premium as non negative scalar
- `set_rate` Sets the roll-up rate into a private field. The argument is a `constant_parameters` object
- `get_payoff` Gets the payoff. The arguments are a `numeric` vector with the amounts and a vector of `timeDate` objects with the start and end dates to calculate the roll-up amount (see **Examples**)

Examples

```
rate <- constant_parameters$new(0.01)
premium <- 100
rollup <- payoff_rollup$new(premium, rate)
t1 <- timeDate::timeDate("2016-01-01")
t2 <- timeDate::timeDate("2016-12-31")
rollup$get_payoff(c(120,100), c(t1,t2))
```

penalty_class *Surrender penalty class*

Description

Class providing a surrender charge. It supports a constant surrender charge (type 1) and two surrender charges decreasing with time, (type 2 and type 3).

Usage

```
penalty_class
```

Format

`R6Class` object.

Value

Object of `R6Class`

Methods

- `new` Initialization methods with arguments:
  - `type` type of the surrender charge. It can be 1 (constant) or 2 or 3 (decreasing with time).
  - `const` positive integer between 0 and 1 with the maximum surrender charge.
  - `T` Positive integer with expiry of the VA product.
- `get` get the surrender penalty. Argument is `time` a scalar in [0, T].
- `set` set the maximum surrender penalty.
- `get_type` get the type of the surrender penalty

Examples

```
#Sets a constant penalty
penalty <- penalty_class$new(type = 1, const = 0.03)
penalty$get()
penalty$set(0.04)
penalty$get()
#Sets a time decreasing penalty of type 2
penalty <- penalty_class$new(type = 2, const = 0.08, T = 10)
penalty$get(time = 0)
penalty$get(time = 2)
penalty$set(0.05)
penalty$get(time = 0)
#Sets a time decreasing penalty of type 3
penalty <- penalty_class$new(type = 3, const = 0.08, T = 10)
penalty$get(time = 0)
penalty$get(time = 2)
penalty$set(0.05)
penalty$get(time = 0)
```

sq — *Square root utility function*

Description

Takes square root if positive otherwise returns zero. To be used with mean reverting squared root processes (CIR SDE)

Usage

```
sq(x)
```

Arguments

x	numeric scalar

va_bs_engine — *Variable Annuity pricing engine with GBM*

Description

Class providing a variable annuity pricing engine with the underlying reference risk neutral fund modeled as a Geometric Brownian Motion and the intensity of mortality modeled by the Weibull intensity of mortality. The value of the VA contract is estimated by means of the Monte Carlo method if the policyholder cannot surrender (the so called "static" approach), and by means of Least Squares Monte Carlo in case the policyholder can surrender the contract (the "mixed" approach).
See **References** -`[BMOP2011]` for a description of the mixed and static approaches and the algorithm implemented by this class, `[LS2001]` for Least Squares Monte Carlo.

Usage

```
va_bs_engine
```

Format

`R6Class` object.

Value

Object of `R6Class`

Methods

new Constructor method with arguments:

product va_product object

interest constant_parameters object with the interest rate

c1 numeric scalar argument of the intensity of mortality function mu

c2 numeric scalar argument of the intensity of mortality function mu

spot numeric scalar with the initial fund price

volatility constant_parameters object with the volatility

dividends constant_parameters object with the dividend rate

death_time Returns the time of death index. If the death doesn't occur during the product time-line it returns the last index of the product time-line

simulate_financial_paths Simulates npaths paths of the underlying fund of the VA contract and the discount factors (interest rate) and saves them into private fields for later use.

simulate_mortality_paths Simulates npaths paths of the intensity of mortality and saves them into private fields for later use.

get_fund Gets the i-th path of the underlying fund where i goes from 1 to npaths

do_static Estimates the VA contract value by means of the static approach (Monte Carlo), see **References**. It takes as arguments:

the_gatherer gatherer object to hold the point estimates

npaths positive integer with the number of paths to simulate

simulate boolean to specify if the paths should be simulated from scratch, default is TRUE.

do_mixed Estimates the VA contract by means of the mixed approach (Least Squares Monte Carlo), see **References**. It takes as arguments:

the_gatherer gatherer object to hold the point estimates

npaths positive integer with the number of paths to simulate

degree positive integer with the maximum degree of the weighted Laguerre polynomials used in the least squares by LSMC

freq string which contains the frequency of the surrender decision. The default is "3m" which corresponds to deciding every three months if surrendering the contract or not.

simulate boolean to specify if the paths should be simulated from scratch, default is TRUE.

get_discount Arguments are i,j. Gets the j-th discount factor corresponding to the i-th simulated path of the discount factors. This method must be implemented by sub-classes.

fair_fee Calculates the fair fee for a contract using the bisection method. Arguments are:

fee_gatherer data_gatherer object to hold the point estimates

npaths numeric scalar with the number of MC simulations to run

lower numeric scalar with the fee corresponding to the lower end of the bisection interval

upper numeric scalar with the fee corresponding to the upper end of the bisection interval

mixed boolean specifying if the mixed method has to be used. The default is FALSE

tol numeric scalar with the tolerance of the bisection algorithm. Default is 1e-4

nmax positive integer with the maximun number of iterations of the bisection algorithm

simulate boolean specifying if financial and mortality paths should be simulated.

References

BMOP2011 *Bacinello A.R., Millossovich P., Olivieri A. ,Pitacco E., "Variable annuities: a unifying valuation approach." In: Insurance: Mathematics and Economics 49 (2011), pp. 285-297.*

LS2001 *Longstaff F.A. e Schwartz E.S. Valuing american options by simulation: a simple least-squares approach. In: Review of Financial studies 14 (2001), pp. 113-147*

Examples

```
#Sets up the payoff as a roll-up of premiums with roll-up rate 1%

rate <- constant_parameters$new(0.01)

premium <- 100
rollup <- payoff_rollup$new(premium, rate)

#Ten years time-line
begin <- timeDate::timeDate("2016-01-01")
end <- timeDate::timeDate("2025-12-31")

#Age of the policyholder.
age <- 60
# A constant fee of 4% per year (365 days)
fee <- constant_parameters$new(0.04)

#Barrier for a state-dependent fee. The fee will be applied only if
#the value of the account is below the barrier
barrier <- Inf
#Withdrawal penalty applied in case the insured surrenders the contract
#It is a constant penalty in this case
penalty <- penalty_class$new(type = 1, 0.01)
#Sets up the contract with GMAB guarantee
contract <- GMAB$new(rollup, t0 = begin, t = end, age = age, fee = fee,
barrier = barrier, penalty = penalty)

#Interest rate
r <- constant_parameters$new(0.03)
#Initial value of the underlying fund
spot <- 100
#Volatility
vol <- constant_parameters$new(0.2)
#Dividend rate
div <- constant_parameters$new(0.0)
#Gatherer for the MC point estimates
the_gatherer <- mc_gatherer$new()
#Number of paths to simulate
no_of_paths <- 1e2

#Sets up the pricing engine specifying the va_contract, the interest rate
#the parameters of the Weibull intensity of mortality, the initial fund
#value, the volatility and dividends rate
engine <- va_bs_engine$new(contract, r, c1=90.43, c2=10.36, spot,
volatility=vol, dividends=div)
```

```
#Estimates the contract value by means of the static approach.

engine$do_static(the_gatherer, no_of_paths)
the_gatherer$get_results()

#Estimates the contract value by means of the mixed approach.
#To compare with the static approach we won't simulate the underlying
#fund paths again.

the_gatherer_2 <- mc_gatherer$new()

engine$do_mixed(the_gatherer_2, no_of_paths, degree = 3,
freq = "3m", simulate = FALSE)
the_gatherer_2$get_results()
```

`va_engine` *Generic Variable Annuity pricing engine*

Description

Class providing an interface for a generic VA pricing engine.
This class shouldn't be instantiated but used as base class for variable annuity pricing engines. The value of the VA contract is estimated by means of the Monte Carlo method if the policyholder cannot surrender (the so called "static" approach), and by means of Least Squares Monte Carlo in case the policyholder can surrender the contract (the "mixed" approach).
See **References** -`[BMOP2011]` for a description of the mixed and static approaches and the algorithm implemented by this class, `[LS2001]` for Least Squares Monte Carlo.

Usage

```
va_engine
```

Format

`R6Class` object.

Value

Object of `R6Class`

Methods

`new` Constructor method

`death_time` Returns the time of death index. If the death doesn't occur during the product time-line it returns the last index of the product time-line plus one.

`simulate_financial_paths` Simulates `npaths` paths of the underlying fund of the VA contract and the discount factors (interest rate) and saves them into private fields for later use.

simulate_mortality_paths Simulates npaths paths of the intensity of mortality and saves them into private fields for later use.

get_fund Gets the i-th path of the underlying fund where i goes from 1 to npaths

do_static Estimates the VA contract value by means of the static approach (Monte Carlo), see **References**. It takes as arguments:

- the_gatherer gatherer object to hold the point estimates
- npaths positive integer with the number of paths to simulate
- simulate boolean to specify if the paths should be simulated from scratch, default is TRUE.

do_mixed Estimates the VA contract by means of the mixed approach (Least Squares Monte Carlo), see **References**. It takes as arguments:

- the_gatherer gatherer object to hold the point estimates
- npaths positive integer with the number of paths to simulate
- degree positive integer with the maximum degree of the weighted Laguerre polynomials used in the least squares by LSMC
- freq string which contains the frequency of the surrender decision. The default is "3m" which corresponds to deciding every three months if surrendering the contract or not.
- simulate boolean to specify if the paths should be simulated from scratch, default is TRUE.

get_discount Arguments are i,j. Gets the j-th discount factor corresponding to the i-th simulated path of the discount factors. This method must be implemented by sub-classes.

fair_fee Calculates the fair fee for a contract using the bisection method. Arguments are:

- fee_gatherer data_gatherer object to hold the point estimates
- npaths numeric scalar with the number of MC simulations to run
- lower numeric scalar with the fee corresponding to the lower end of the bisection interval
- upper numeric scalar with the fee corresponding to the upper end of the bisection interval
- mixed boolean specifying if the mixed method has to be used. The default is FALSE
- tol numeric scalar with the tolerance of the bisection algorithm. Default is 1e-4
- nmax positive integer with the maximun number of iterations of the bisection algorithm
- simulate boolean specifying if financial and mortality paths should be simulated.

References

BMOP2011 *Bacinello A.R., Millossovich P., Olivieri A. ,Pitacco E., "Variable annuities: a unifying valuation approach." In: Insurance: Mathematics and Economics 49 (2011), pp. 285-297.*

LS2001 *Longstaff F.A. e Schwartz E.S. Valuing american options by simulation: a simple least-squares approach. In: Review of Financial studies 14 (2001), pp. 113-147*

va_mkh_engine *Variable Annuity pricing engine with GBM and Makeham*

Description

Class providing a variable annuity pricing engine with the underlying reference risk neutral fund modeled as a Geometric Brownian Motion and the intensity of mortality modeled by the Makeham intensity of mortality. The value of the VA contract is estimated by means of the Monte Carlo method if the policyholder cannot surrender (the so called "static" approach), and by means of Least Squares Monte Carlo in case the policyholder can surrender the contract (the "mixed" approach).
See **References** -[BMOP2011] for a description of the mixed and static approaches and the algorithm implemented by this class, [LS2001] for Least Squares Monte Carlo.

Usage

```
va_mkh_engine
```

Format

R6Class object.

Value

Object of R6Class

Methods

new Constructor method with arguments:

- product va_product object
- interest constant_parameters object with the interest rate
- A numeric scalar argument of the intensity of mortality function makeham
- B numeric scalar argument of the intensity of mortality function makeham
- spot numeric scalar with the initial fund price
- volatility constant_parameters object with the volatility
- dividends constant_parameters object with the dividend rate
- c numeric scalar argument of the intensity of mortality function makeham

death_time Returns the time of death index. If the death doesn't occur during the product time-line it returns the last index of the product time-line

simulate_financial_paths Simulates npaths paths of the underlying fund of the VA contract and the discount factors (interest rate) and saves them into private fields for later use.

simulate_mortality_paths Simulates npaths paths of the intensity of mortality and saves them into private fields for later use.

get_fund Gets the i-th path of the underlying fund where i goes from 1 to npaths

do_static Estimates the VA contract value by means of the static approach (Monte Carlo), see **References**. It takes as arguments:

the_gatherer gatherer object to hold the point estimates

npaths positive integer with the number of paths to simulate

simulate boolean to specify if the paths should be simulated from scratch, default is TRUE.

do_mixed Estimates the VA contract by means of the mixed approach (Least Squares Monte Carlo), see **References**. It takes as arguments:

the_gatherer gatherer object to hold the point estimates

npaths positive integer with the number of paths to simulate

degree positive integer with the maximum degree of the weighted Laguerre polynomials used in the least squares by LSMC

freq string which contains the frequency of the surrender decision. The default is "3m" which corresponds to deciding every three months if surrendering the contract or not.

simulate boolean to specify if the paths should be simulated from scratch, default is TRUE.

get_discount Arguments are i,j. Gets the j-th discount factor corresponding to the i-th simulated path of the discount factors. This method must be implemented by sub-classes.

fair_fee Calculates the fair fee for a contract using the bisection method. Arguments are:

fee_gatherer data_gatherer object to hold the point estimates

npaths numeric scalar with the number of MC simulations to run

lower numeric scalar with the fee corresponding to the lower end of the bisection interval

upper numeric scalar with the fee corresponding to the upper end of the bisection interval

mixed boolean specifying if the mixed method has to be used. The default is FALSE

tol numeric scalar with the tolerance of the bisection algorithm. Default is 1e-4

nmax positive integer with the maximun number of iterations of the bisection algorithm

simulate boolean specifying if financial and mortality paths should be simulated.

References

BMOP2011 *Bacinello A.R., Millossovich P., Olivieri A. ,Pitacco E., "Variable annuities: a unifying valuation approach." In: Insurance: Mathematics and Economics 49 (2011), pp. 285-297.*

LS2001 *Longstaff F.A. e Schwartz E.S. Valuing american options by simulation: a simple least-squares approach. In: Review of Financial studies 14 (2001), pp. 113-147*

Examples

```
#Sets up the payoff as a roll-up of premiums with roll-up rate 1%

rate <- constant_parameters$new(0.01)

premium <- 100
rollup <- payoff_rollup$new(premium, rate)

#Ten years time-line
begin <- timeDate::timeDate("2016-01-01")
end <- timeDate::timeDate("2025-12-31")

#Age of the policyholder.
age <- 60
```

```
# A constant fee of 4% per year (365 days)
fee <- constant_parameters$new(0.04)

#Barrier for a state-dependent fee. The fee will be applied only if
#the value of the account is below the barrier
barrier <- Inf
#Withdrawal penalty applied in case the insured surrenders the contract
#It is a constant penalty in this case
penalty <- penalty_class$new(type = 1, 0.01)
#Sets up the contract with GMAB guarantee
contract <- GMAB$new(rollup, t0 = begin, t = end, age = age, fee = fee,
barrier = barrier, penalty = penalty)

#Interest rate
r <- constant_parameters$new(0.03)
#Initial value of the underlying fund
spot <- 100
#Volatility
vol <- constant_parameters$new(0.2)
#Dividend rate
div <- constant_parameters$new(0.0)
#Gatherer for the MC point estimates
the_gatherer <- mc_gatherer$new()
#Number of paths to simulate
no_of_paths <- 1e2

#Sets up the pricing engine specifying the va_contract, the interest rate
#the parameters of the Makeham intensity of mortality, the initial fund
#value, the volatility and dividends rate
engine <- va_mkh_engine$new(contract, r, A = 0.0001, B = 0.00035, spot,
volatility = vol, dividends = div, c = 1.075)

#Estimates the contract value by means of the static approach.

engine$do_static(the_gatherer, no_of_paths)
the_gatherer$get_results()

#Estimates the contract value by means of the mixed approach.
#To compare with the static approach we won't simulate the underlying
#fund paths again.

the_gatherer_2 <- mc_gatherer$new()

engine$do_mixed(the_gatherer_2, no_of_paths, degree = 3,
freq = "3m", simulate = FALSE)
the_gatherer_2$get_results()
```

---

va_product *Generic Variable Annuity product class*

Description

Class providing an interface for a generic VA product object. This class shouldn't be instantiated but used as base class for implementing products with contract riders such as GMAB, GMIB, etc. It supports a simple state-dependent fee structure with a single barrier.
See **References** for a description of variable annuities life insurance products, their guarantees and fee structures.

Usage

```
va_product
```

Format

`R6Class` object.

Value

Object of `R6Class`

Methods

`new` Constructor method with arguments:

- `payoff` `payoff` object of the GMAB guarantee
- `t0` `timeDate` object with the issue date of the contract
- `t` `timeDate` object with the end date of the accumulation period
- `t1` `timeDate` object with the end date of the life benefit payment
- `age` `numeric` positive scalar with the age of the policyholder
- `fee` `constant_parameters` object with the fee
- `barrier` `numeric` positive scalar with the state-dependent fee barrier
- `penalty` `penalty_class` object with the penalty

`get_times` get method for the product time-line. Returns a `timeDate` object

`get_age` get method for the age of the insured

`set_age` set method for the age of the insured

`get_barrier` get method for the state-dependent fee barrier. Returns a positive scalar with the barrier

`set_barrier` set method for the state-dependent fee barrier. Argument must be a positive scalar.

`set_penalty_object` the argument `penalty` is a `penalty_class` object which is stored in a private field.

`get_penalty_object` gets the `penalty_class` object.

`set_penalty` set method for the penalty applied in case of surrender. The argument must be a scalar between 0 and 1.

`get_penalty` get method for the surrender penalties. It can be a scalar between 0 and 1 in case the penalty is constant or a numeric vector in case the penalty varies with time.

`set_fee` set method for the contract fee. The argument is a `constant_parameters` object with the fee.

`set_payoff` set method for the `payoff_guarantee` object.

`survival_benefit_times` returns a `numeric` vector with the survival benefit time indexes.

`surrender_times` returns a `numeric` vector with the surrender time indexes. Takes as argument a string with the frequency of the decision if surrendering the contract, e.g. "3m" corresponds to a surrender decision taken every 3 months.

`times_in_yrs` returns the product time-line in fraction of year

`cash_flows` returns a `numeric` vector with the cash flows of the product. It takes as argument `spot_values` a `numeric` vector which holds the values of the underlying fund this method will calculate the cash flows from

`survival_benefit` Returns a numeric scalar corresponding to the survival benefit. The arguments are `spot_values` vector which holds the values of the underlying fund and `t` the time index of the survival benefit. The function will return 0 if there's no survival benefit at the specified time

`get_premium` Returns the premium as non negative scalar

References

BMOP2011 *Bacinello A.R., Millossovich P., Olivieri A., Pitacco E., "Variable annuities: a unifying valuation approach." In: Insurance: Mathematics and Economics 49 (2011), pp. 285-297.*

BHM2014 *Bernard C., Hardy M. and Mackay A. "State-dependent fees for variable annuity guarantees." In: Astin Bulletin 44 (2014), pp. 559-585.*

`va_sde_engine` *General Variable Annuity pricing engine*

Description

Class providing a variable annuity pricing engine where the underlying reference fund and the intensity of mortality are specified by an arbitrary system of stochastic differential equations. The simulation is done by means of the yuima package.
The value of the VA contract is estimated by means of the Monte Carlo method if the policyholder cannot surrender (the so called "static" approach), and by means of Least Squares Monte Carlo in case the policyholder can surrender the contract (the "mixed" approach).
See **References** -`[BMOP2011]` for a description of the mixed and static approaches and the algorithm implemented by this class, `[LS2001]` for Least Squares Monte Carlo and `[YUIMA2014]` for `yuima`.

Usage

```
va_sde_engine
```

Format

`R6Class` object.

Value

Object of R6Class

Methods

new Constructor method with arguments:

- product A va_product object with the VA product.
- financial_parms A list of parameters specifying the financial processes. See financials_BMOP2011 for an example.
- mortality_parms A list of parameters specifying the demographic processes. See mortality_BMOP2011 for an example.

death_time Returns the time of death index. If the death doesn't occur during the product time-line it returns the last index of the product time-line plus one.

simulate_financial_paths Simulates npaths paths of the underlying fund of the VA contract and the discount factors (interest rate) and saves them into private fields for later use.

simulate_mortality_paths Simulates npaths paths of the intensity of mortality and saves them into private fields for later use.

get_fund Gets the i-th path of the underlying fund where i goes from 1 to npaths.

do_static Estimates the VA contract value by means of the static approach (Monte Carlo), see **References**. It takes as arguments:

- the_gatherer gatherer object to hold the point estimates
- npaths positive integer with the number of paths to simulate
- simulate boolean to specify if the paths should be simulated from scratch, default is TRUE.

do_mixed Estimates the VA contract by means of the mixed approach (Least Squares Monte Carlo), see **References**. It takes as arguments:

- the_gatherer gatherer object to hold the point estimates
- npaths positive integer with the number of paths to simulate
- degree positive integer with the maximum degree of the weighted Laguerre polynomials used in the least squares by LSMC
- freq string which contains the frequency of the surrender decision. The default is "3m" which corresponds to deciding every three months if surrendering the contract or not.
- simulate boolean to specify if the paths should be simulated from scratch, default is TRUE.

get_discount Arguments are i,j. Gets the j-th discount factor corresponding to the i-th simulated path of the discount factors.

fair_fee Calculates the fair fee for a contract using the bisection method. Arguments are:

- fee_gatherer data_gatherer object to hold the point estimates
- npaths numeric scalar with the number of MC simulations to run
- lower numeric scalar with the fee corresponding to the lower end of the bisection interval
- upper numeric scalar with the fee corresponding to the upper end of the bisection interval
- mixed boolean specifying if the mixed method has to be used. The default is FALSE
- tol numeric scalar with the tolerance of the bisection algorithm. Default is 1e-4
- nmax positive integer with the maximun number of iterations of the bisection algorithm
- simulate boolean specifying if financial and mortality paths should be simulated.

References

BMOP2011 *Bacinello A.R., Millossovich P., Olivieri A. ,Pitacco E., "Variable annuities: a unifying valuation approach." In: Insurance: Mathematics and Economics 49 (2011), pp. 285-297.*

LS2001 *Longstaff F.A. e Schwartz E.S. Valuing american options by simulation: a simple least-squares approach. In: Review of Financial studies 14 (2001), pp. 113-147*

YUIMA2014 *Alexandre Brouste, Masaaki Fukasawa, Hideitsu Hino, Stefano M. Iacus, Kengo Kamatani, Yuta Koike, Hiroki Masuda, Ryosuke Nomura, Teppei Ogihara, Yasutaka Shimuzu, Masayuki Uchida, Nakahiro Yoshida (2014). The YUIMA Project: A Computational Framework for Simulation and Inference of Stochastic Differential Equations. Journal of Statistical Software, 57(4), 1-51. URL http://www.jstatsoft.org/v57/i04/.*

Examples

```
#Sets up the payoff as a roll-up of premiums with roll-up rate 2%

rate <- constant_parameters$new(0.02)

premium <- 100
rollup <- payoff_rollup$new(premium, rate)

#Five years time-line
begin <- timeDate::timeDate("2016-01-01")
end <- timeDate::timeDate("2020-12-31")

#Age of the policyholder.
age <- 60
# A constant fee of 2% per year (365 days)
fee <- constant_parameters$new(0.02)

#Barrier for a state-dependent fee. The fee will be applied only if
#the value of the account is below the barrier
barrier <- 200
#Withdrawal penalty applied in case the insured surrenders the contract
#It is a constant penalty in this case
penalty <- penalty_class$new(type = 1, 0.02)
#Sets up the contract with GMAB guarantee
contract <- GMAB$new(rollup, t0 = begin, t = end, age = age, fee = fee,
barrier = barrier, penalty = penalty)

#Sets up a gatherer of the MC point estimates
the_gatherer  <- mc_gatherer$new()
no_of_paths <- 10

#Sets up the pricing engine
engine <- va_sde_engine$new(contract, financials_BMOP2011,
mortality_BMOP2011)

#Estimates the contract value by means of the static approach

engine$do_static(the_gatherer, no_of_paths)
the_gatherer$get_results()
```

```
#Estimates the contract value by means of the mixed approach
#To compare with the static approach we don't simulate the underlying
#fund paths again.

the_gatherer_2 <- mc_gatherer$new()

engine$do_mixed(the_gatherer_2, no_of_paths, degree = 3, freq = "3m",
simulate = FALSE)
the_gatherer_2$get_results()
```

---

va_sde_engine2 *Variable Annuity pricing engine with general financial processes and Weibull mortality*

Description

Class providing a variable annuity pricing engine where the underlying reference fund and interest rates are specified by an arbitrary system of stochastic differential equations. In contrast the intensity of mortality is determinitic and given by the Weibull function. The financial paths are simulated by means of the yuima package.
The value of the VA contract is estimated by means of the Monte Carlo method if the policyholder cannot surrender (the so called "static" approach), and by means of Least Squares Monte Carlo in case the policyholder can surrender the contract (the "mixed" approach).
See **References** -`[BMOP2011]` for a description of the mixed and static approaches and the algorithm implemented by this class, `[LS2001]` for Least Squares Monte Carlo and `[YUIMA2014]` for `yuima`.

Usage

```
va_sde_engine2
```

Format

`R6Class` object.

Value

Object of `R6Class`

Methods

`new` Constructor method with arguments:

- `product` A `va_product` object with the VA product.
- `financial_parms` A list of parameters specifying the financial processes. See `financials_BZ2016` for an example.
- `c1` `numeric` scalar argument of the intensity of mortality function `mu`
- `c2` `numeric` scalar argument of the intensity of mortality function `mu`

death_time Returns the time of death index. If the death doesn't occur during the product time-line it returns the last index of the product time-line plus one.

simulate_financial_paths Simulates npaths paths of the underlying fund of the VA contract and the discount factors (interest rate) and saves them into private fields for later use.

simulate_mortality_paths Simulates npaths paths of the intensity of mortality and saves them into private fields for later use.

get_fund Gets the i-th path of the underlying fund where i goes from 1 to npaths.

do_static Estimates the VA contract value by means of the static approach (Monte Carlo), see **References**. It takes as arguments:

- the_gatherer gatherer object to hold the point estimates
- npaths positive integer with the number of paths to simulate
- simulate boolean to specify if the paths should be simulated from scratch, default is TRUE.

do_mixed Estimates the VA contract by means of the mixed approach (Least Squares Monte Carlo), see **References**. It takes as arguments:

- the_gatherer gatherer object to hold the point estimates
- npaths positive integer with the number of paths to simulate
- degree positive integer with the maximum degree of the weighted Laguerre polynomials used in the least squares by LSMC
- freq string which contains the frequency of the surrender decision. The default is "3m" which corresponds to deciding every three months if surrendering the contract or not.
- simulate boolean to specify if the paths should be simulated from scratch, default is TRUE.

get_discount Arguments are i,j. Gets the j-th discount factor corresponding to the i-th simulated path of the discount factors.

fair_fee Calculates the fair fee for a contract using the bisection method. Arguments are:

- fee_gatherer data_gatherer object to hold the point estimates
- npaths numeric scalar with the number of MC simulations to run
- lower numeric scalar with the fee corresponding to the lower end of the bisection interval
- upper numeric scalar with the fee corresponding to the upper end of the bisection interval
- mixed boolean specifying if the mixed method has to be used. The default is FALSE
- tol numeric scalar with the tolerance of the bisection algorithm. Default is 1e-4
- nmax positive integer with the maximun number of iterations of the bisection algorithm
- simulate boolean specifying if financial and mortality paths should be simulated.

References

BMOP2011 *Bacinello A.R., Millossovich P., Olivieri A. ,Pitacco E., "Variable annuities: a unifying valuation approach." In: Insurance: Mathematics and Economics 49 (2011), pp. 285-297.*

LS2001 *Longstaff F.A. e Schwartz E.S. Valuing american options by simulation: a simple least-squares approach. In: Review of Financial studies 14 (2001), pp. 113-147*

YUIMA2014 *Alexandre Brouste, Masaaki Fukasawa, Hideitsu Hino, Stefano M. Iacus, Kengo Kamatani, Yuta Koike, Hiroki Masuda, Ryosuke Nomura, Teppei Ogihara, Yasutaka Shimuzu, Masayuki Uchida, Nakahiro Yoshida (2014). The YUIMA Project: A Computational Framework for Simulation and Inference of Stochastic Differential Equations. Journal of Statistical Software, 57(4), 1-51. URL http://www.jstatsoft.org/v57/i04/.*

Examples

```
#Sets up the payoff as a roll-up of premiums with roll-up rate 2%

rate <- constant_parameters$new(0.02)

premium <- 100
rollup <- payoff_rollup$new(premium, rate)

#Five years time-line
begin <- timeDate::timeDate("2016-01-01")
end <- timeDate::timeDate("2020-12-31")

#Age of the policyholder.
age <- 50
# A constant fee of 2% per year (365 days)
fee <- constant_parameters$new(0.02)

#Barrier for a state-dependent fee. The fee will be applied only if
#the value of the account is below the barrier
barrier <- 200
#Withdrawal penalty applied in case the insured surrenders the contract
#It is a constant penalty in this case
penalty <- penalty_class$new(type = 1, 0.02)
#Sets up the contract with GMAB guarantee
contract <- GMAB$new(rollup, t0 = begin, t = end, age = age, fee = fee,
barrier = barrier, penalty = penalty)

#Sets up a gatherer of the MC point estimates
the_gatherer  <- mc_gatherer$new()
no_of_paths <- 10

#Sets up the pricing engine
engine <- va_sde_engine2$new(contract, financials_BMOP2011)

#Estimates the contract value by means of the static approach

engine$do_static(the_gatherer, no_of_paths)
the_gatherer$get_results()


#Estimates the contract value by means of the mixed approach
#To compare with the static approach we don't simulate the underlying
#fund paths again.

the_gatherer_2 <- mc_gatherer$new()

engine$do_mixed(the_gatherer_2, no_of_paths, degree = 3, freq = "3m",
simulate = FALSE)
the_gatherer_2$get_results()
```

va_sde_engine3 *Variable Annuity pricing engine with general fund processes and Weibull mortality*

Description

Class providing a variable annuity pricing engine where the underlying reference fund is specified by an arbitrary system of stochastic differential equations. In contrast, the interest rates is constant and the intensity of mortality is determinitic and given by the Weibull function. The fund paths are simulated by means of the yuima package.
The value of the VA contract is estimated by means of the Monte Carlo method if the policyholder cannot surrender (the so called "static" approach), and by means of Least Squares Monte Carlo in case the policyholder can surrender the contract (the "mixed" approach).
See **References** -`[BMOP2011]` for a description of the mixed and static approaches and the algorithm implemented by this class, `[LS2001]` for Least Squares Monte Carlo and `[YUIMA2014]` for `yuima`.

Usage

```
va_sde_engine3
```

Format

`R6Class` object.

Value

Object of `R6Class`

Methods

`new` Constructor method with arguments:

- `product` A `va_product` object with the VA product.
- `financial_parms` A list of parameters specifying the financial processes. See `financials_BZ2016bis` for an example.
- `interest` `constant_parameters` object with the constant interest rate
- `c1` `numeric` scalar argument of the intensity of mortality function `mu`
- `c2` `numeric` scalar argument of the intensity of mortality function `mu`

`death_time` Returns the time of death index. If the death doesn't occur during the product time-line it returns the last index of the product time-line plus one.

`simulate_financial_paths` Simulates `npaths` paths of the underlying fund of the VA contract and the discount factors (interest rate) and saves them into private fields for later use.

`simulate_mortality_paths` Simulates `npaths` paths of the intensity of mortality and saves them into private fields for later use.

`get_fund` Gets the `i`-th path of the underlying fund where `i` goes from 1 to `npaths`.

`do_static` Estimates the VA contract value by means of the static approach (Monte Carlo), see **References**. It takes as arguments:

- `the_gatherer` `gatherer` object to hold the point estimates
- `npaths` positive integer with the number of paths to simulate
- `simulate` boolean to specify if the paths should be simulated from scratch, default is TRUE.

`do_mixed` Estimates the VA contract by means of the mixed approach (Least Squares Monte Carlo), see **References**. It takes as arguments:

- `the_gatherer` `gatherer` object to hold the point estimates
- `npaths` positive integer with the number of paths to simulate
- `degree` positive integer with the maximum degree of the weighted Laguerre polynomials used in the least squares by LSMC
- `freq` string which contains the frequency of the surrender decision. The default is `"3m"` which corresponds to deciding every three months if surrendering the contract or not.
- `simulate` boolean to specify if the paths should be simulated from scratch, default is TRUE.

`get_discount` Arguments are `i,j`. Gets the `j`-th discount factor corresponding to the `i`-th simulated path of the discount factors.

`fair_fee` Calculates the fair fee for a contract using the bisection method. Arguments are:

- `fee_gatherer` `data_gatherer` object to hold the point estimates
- `npaths` `numeric` scalar with the number of MC simulations to run
- `lower` `numeric` scalar with the fee corresponding to the lower end of the bisection interval
- `upper` `numeric` scalar with the fee corresponding to the upper end of the bisection interval
- `mixed` `boolean` specifying if the mixed method has to be used. The default is `FALSE`
- `tol` `numeric` scalar with the tolerance of the bisection algorithm. Default is `1e-4`
- `nmax` positive `integer` with the maximun number of iterations of the bisection algorithm
- `simulate` boolean specifying if financial and mortality paths should be simulated.

References

BMOP2011 *Bacinello A.R., Millossovich P., Olivieri A. ,Pitacco E., "Variable annuities: a unifying valuation approach." In: Insurance: Mathematics and Economics 49 (2011), pp. 285-297.*

LS2001 *Longstaff F.A. e Schwartz E.S. Valuing american options by simulation: a simple least-squares approach. In: Review of Financial studies 14 (2001), pp. 113-147*

YUIMA2014 *Alexandre Brouste, Masaaki Fukasawa, Hideitsu Hino, Stefano M. Iacus, Kengo Kamatani, Yuta Koike, Hiroki Masuda, Ryosuke Nomura, Teppei Ogihara, Yasutaka Shimuzu, Masayuki Uchida, Nakahiro Yoshida (2014). The YUIMA Project: A Computational Framework for Simulation and Inference of Stochastic Differential Equations. Journal of Statistical Software, 57(4), 1-51. URL http://www.jstatsoft.org/v57/i04/.*

Examples

```
#Sets up the payoff as a roll-up of premiums with roll-up rate 2%

rate <- constant_parameters$new(0.02)

premium <- 100
```

```
rollup <- payoff_rollup$new(premium, rate)

#constant interest rate
r <- constant_parameters$new(0.03)

#Five years time-line
begin <- timeDate::timeDate("2016-01-01")
end <- timeDate::timeDate("2020-12-31")

#Age of the policyholder.
age <- 50
# A constant fee of 2% per year (365 days)
fee <- constant_parameters$new(0.02)

#Barrier for a state-dependent fee. The fee will be applied only if
#the value of the account is below the barrier
barrier <- 200
#Withdrawal penalty applied in case the insured surrenders the contract
#It is a constant penalty in this case
penalty <- penalty_class$new(type = 1, 0.02)
#Sets up the contract with GMAB guarantee
contract <- GMAB$new(rollup, t0 = begin, t = end, age = age, fee = fee,
barrier = barrier, penalty = penalty)

#Sets up a gatherer of the MC point estimates
the_gatherer  <- mc_gatherer$new()
no_of_paths <- 10

#Sets up the pricing engine
engine <- va_sde_engine3$new(contract, financials_BZ2016bis, interest = r)

#Estimates the contract value by means of the static approach

engine$do_static(the_gatherer, no_of_paths)
the_gatherer$get_results()


#Estimates the contract value by means of the mixed approach
#To compare with the static approach we don't simulate the underlying
#fund paths again.

the_gatherer_2 <- mc_gatherer$new()

engine$do_mixed(the_gatherer_2, no_of_paths, degree = 3, freq = "3m",
simulate = FALSE)
the_gatherer_2$get_results()
```

yr_fractions *Normalizes a timeDate sequence into year fractions*

Description

Normalizes a timeDate sequence into year fractions

Usage

```
yr_fractions(times)
```

Arguments

times	A timeDate sequence

Index