Package 'imaginator'

June 6, 2017

Title Simulate General Insurance Policies and Losses

Version 0.1.1

Description

Simulate general insurance policies, losses and loss emergence. The package contemplates deterministic and stochastic policy retention and growth scenarios. Retention and growth rates are percentages relative to the expiring portfolio. Claims are simulated for each policy. This is accomplished either be assuming a frequency distribution per development lag or by generating random wait times until claim emergence and settlement. Loss simulation uses standard loss distributions for claim amounts.

License GPL-3

Depends R (>= 3.2.3)

LazyData true

Imports dplyr, magrittr, assertthat, lubridate, stringi

Suggests testthat, knitr, rmarkdown, ggplot2

VignetteBuilder knitr

RoxygenNote 6.0.1

NeedsCompilation no

Author Brian Fannin [aut, cre]

Maintainer Brian Fannin <FanninQED@Yahoo.com>

Repository CRAN

Date/Publication 2017-06-05 23:41:53 UTC

R topics documented:

`BinomialHelper` *BinomialHelper*

Description

This will return a list of poisson random number generator functions.

Usage

```
BinomialHelper(size, prob, SingletonList = FALSE)
```

Arguments

`size`	number of trials
`prob`	probability of success on each trial
`SingletonList`	Boolean indicating whether to return a single function as a list. Default FALSE.

Details

The lambda parameter may be a vector.

Value

A list of functions.

Examples

```
myFuncs <- BinomialHelper(size = 100, prob = c(.9, .1))
myFuncs[[1]](4)
```

`ClaimsByFirstReport` *Claims by first report*

Description

Given a data frame of policies, this will simulate the number of claims- and their initial payment- per policy by the development lag at which they are first reported.

Usage

```
ClaimsByFirstReport(dfPolicy, Frequency, PaymentSeverity, Lags)
```

Arguments

`dfPolicy`	A policy data frame
`Frequency`	A function, or list of functions which will randomly generate the number of claims per policy
`PaymentSeverity`	A function, or list of functions which will randomly generate the payment amount for each claim
`Lags`	A vector of lags as integers

Details

Creates a data frame with randomly generated claim values.

Value

A claims data frame

Examples

```
# This will generate a claim data frame which has 1,000 records
# each of which has a severity of 100
dfPolicy <- NewPolicyYear(100, 2001)
dfClaims <- ClaimsByFirstReport(
               dfPolicy
             , Frequency = FixedHelper(10)
             , PaymentSeverity = FixedHelper(100)
             , Lags = 1)
```

`ClaimsByLinkRatio` *Claims by link ratio*

Description

Given a data frame of claims, this will simulate claim development by applying a (possibly) random link ratio.

Usage

```
ClaimsByLinkRatio(dfClaims, Links, Lags)
```

Arguments

`dfClaims`	A claims data frame
`Links`	A list of functions which dictate how severities change from one evaluation date to the next
`Lags`	A vector of lags

Details

This function will apply the link ratio algorithm at an individual claim level.

Value

A claims data frame

Examples

```
dfPolicy <- NewPolicyYear(10, 2001)
dfClaims <- ClaimsByFirstReport(
               dfPolicy
             , Frequency = FixedHelper(10)
             , PaymentSeverity = FixedHelper(100)
             , Lags = 1)
dfClaims <- ClaimsByLinkRatio(dfClaims
                              , Links = FixedHelper(c(1.25, 1.1, 1.05))
                              , Lags = 1:4)
```

`ClaimsByWaitTime` *ClaimsByWaitTime*

Description

Construct a data frame of claims simulated by time between events.

Usage

```
ClaimsByWaitTime(dfPolicy, ClaimFrequency, PaymentFrequency, OccurrenceWait,
  ReportWait, PayWait, PaySeverity, PayOnlyPositive = TRUE)
```

Arguments

`dfPolicy`	A data frame of policy records
`ClaimFrequency`	A function which will randomly generate the number of claims per policy
`PaymentFrequency`	A function which will determine the number of payments per claim
`OccurrenceWait`	A function which will generate the time until occurrence for each claim
`ReportWait`	A function which will generate the time until report
`PayWait`	A function which will generate the lag time between payments
`PaySeverity`	A function which will randomly generate the severity of each claim payment
`PayOnlyPositive`	Boolean indicating whether to discard negative payments.

Details

This function will generate claim transactions.

Wait times and frequencies will be converted to integers with no message. If wait times or claim frequencies are less than zero, or payment frequencies are less than one, they will be converted with a message.

`FixedHelper` *FixedHelper*

Description

Returns the same, non-stochastic, value

Usage

```
FixedHelper(Fixed, SingletonList = FALSE)
```

Arguments

Fixed	A scalar value
SingletonList	Boolean indicating whether to return a single function as a list. Default FALSE.

Value

A function

GammaHelper *GammaHelper*

Description

This will create a random number generator for the gamma function.

Usage

```
GammaHelper(alpha, beta, SingletonList = FALSE)
```

Arguments

alpha	Double for the α parameter
beta	Double for the β parameter
SingletonList	Boolean indicating whether to return a single function as a list. Default FALSE.

GrowPolicies *Simulate policy growth*

Description

Given a policy data frame, this will generate new policies in subsequent policy years.

Usage

```
GrowPolicies(dfPolicy, Growth)
```

Arguments

dfPolicy	Data frame of policy data
Growth	Scalar value greater than or equal to zero

`imaginator` *imaginator*

Description

Simulate general insurance policies, losses and loss emergence. The package contemplates deterministic and stochastic policy retention and growth scenarios. Retention and growth rates are percentages relative to the expiring portfolio. Claims are simulated for each policy. This is accomplished either be assuming a frequency distribution per development lag or by generating random wait times until claim emergence and settlement. Loss simulation uses standard loss distributions for claim amounts.

`IncrementPolicyYear` *Incremental a policy year*

Description

Given a policy data frame, this will combine the `GrowPolicies` and `RenewPolicies` functions to produce a subsequent policy year.

Usage

```
IncrementPolicyYear(dfPolicy, Retention, Growth)
```

Arguments

`dfPolicy`	A policy data frame
`Retention`	Scalar renewal rate
`Growth`	Scalar growth rate

Value

Policy data frame

LognormalHelper *LognormalHelper*

Description

This will create a random number generator for the lognormal function

Usage

```
LognormalHelper(meanlog, sdlog, SingletonList = FALSE)
```

Arguments

meanlog	Mean on the log scale
sdlog	Standard deviation on the log scale
SingletonList	Boolean indicating whether to return a single function as a list. Default FALSE.

NewPolicyYear *Simulate a new policy year*

Description

This will generate a data frame of policy data. This may be used to construct renewal and growth data frames for subsequent policy years.

Usage

```
NewPolicyYear(N, PolicyYear, Exposure = 1, StartID = 1, AdditionalColumns)
```

Arguments

N	The number of policies to generate
PolicyYear	Scalar integer indicating the policy year to generate
Exposure	Vector of exposures
StartID	Integer of the first number in the policy ID sequence
AdditionalColumns	A list of addtional column names and values

Details

Effective dates are uniformly distributed throughout the year.

When providing additional columns, each element of the list must be a scalar and be named.

Value

Data frame of policy data

`NormalHelper` *NormalHelper*

Description

Returns a normal distribution with optional bounds on the returned values

Usage

```
NormalHelper(mean, sd, lowerBound, upperBound, SingletonList = FALSE)
```

Arguments

`mean`	The mean
`sd`	The standard deviation
`lowerBound`	Lower boundary of the return values
`upperBound`	Upper bound of the returned values
`SingletonList`	Boolean indicating whether to return a single function as a list. Default FALSE.

Value

A function

`PoissonHelper` *PoissonHelper*

Description

This will return a list of poisson random number generator functions.

Usage

```
PoissonHelper(lambda, SingletonList = FALSE)
```

Arguments

`lambda`	Expected value of poisson function
`SingletonList`	Boolean indicating whether to return a single function as a list. Default FALSE.

Details

The function is vectorised in the sense that one may pass in a vector of function parameters and receive a list of functions with the same length.

If only one parameter is supplied and the user does not ask for a list to be returned, this will return a function.

Value

A list of functions or a single function (see Details).

Examples

```
myFuncs <- PoissonHelper(c(10, 20))
myFuncs[[1]](10)
myFuncs[[2]](10)

myFunc <- PoissonHelper(10)
is.function(myFunc)
myFunc(10)

myFunc <- PoissonHelper(10, SingletonList = TRUE)
is.list(myFunc)
myFunc[[1]](10)
```

RenewPolicies *Simulate policy renewal*

Description

Given a policy data frame, this will construct renewal data frames. The number of policies which renew is governed by the the `Retention` parameter.

Usage

```
RenewPolicies(dfPolicy, Retention)
```

Arguments

`dfPolicy`	Data frame of policy data
`Retention`	Scalar value greater than or equal to zero

SimulatePolicies *Simulate a data frame of policies*

Description

Given a starting number of policies, this function will generate additional years of policy data.

Growth is given as a the positive rate of growth of new policies. This may be set to zero.

Retention is given as the portion of expiring policies which will renew.

Usage

```
SimulatePolicies(N, PolicyYears, NumYears, Exposure = 1, Retention = 1,
  Growth = 0, StartID = 1, AdditionalColumns)
```

Arguments

N	An integer giving the number of policies in the first year
PolicyYears	A vector of integers in sequence
NumYears	The number of years to simulate. If 'PolicyYears' is given, this is ignored.
Exposure	Exposure per policy
Retention	A vector indicating loss of policies
Growth	A vector indicating the rate of growth of policies
StartID	Integer of the first number in the policy ID sequence
AdditionalColumns	A list of addtional column names and values

Value

A data frame of policy data

UniformHelper *UniformHelper*

Description

Returns a uniform distribution function

Usage

```
UniformHelper(min, max, SingletonList = FALSE)
```

Arguments

min	The minimum value
max	The maximum value
SingletonList	Boolean indicating whether to return a single function as a list. Default FALSE.

Value

A function

Index