

Package ‘imaginator’

October 13, 2022

Title Simulate General Insurance Policies and Losses

Version 1.0.0

Maintainer Brian Fannin <bfannin@casact.org>

Description

Simulate general insurance policies, losses and loss emergence. The functions contemplate 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 by 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 MPL-2.0 | file LICENSE

Depends R (>= 3.2.3)

Imports assertthat, checkmate, distributions3, dplyr, lubridate, magrittr, rlang, stringi, tibble

Suggests testthat, knitr, rmarkdown, ggplot2

VignetteBuilder knitr

RoxygenNote 7.1.2

Encoding UTF-8

URL <https://github.com/casact/imaginator>

NeedsCompilation no

Author Brian Fannin [aut, cre]

Repository CRAN

Date/Publication 2022-01-27 08:40:02 UTC

R topics documented:

claims_by_first_report	2
claims_by_link_ratio	3

claims_by_wait_time	4
imaginator	5
policies_grow	5
policies_renew	6
policies_simulate	7
policy_year_increment	8
policy_year_new	8

Index	10
--------------	-----------

claims_by_first_report
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

```
claims_by_first_report(tbl_policy, frequency, payment_severity, lags)
```

Arguments

tbl_policy	A policy data frame.
frequency	A list of the same length as ‘lags’ of number of claims per policy or their distributions.
payment_severity	A list of the same length as ‘lags’ of payment amount for each claim or their distributions.
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
tbl_policy <- policy_year_new(100, 2001)
tbl_claims <- claims_by_first_report(
  tbl_policy,
  frequency = 10,
  payment_severity = 100,
  lags = 1)
```

claims_by_link_ratio *Claims by link ratio*

Description

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

Usage

```
claims_by_link_ratio(tbl_claims, links, lags)
```

Arguments

tbl_claims	A claims data frame
links	A vector of the same length as 'lags' of factors, or their distributions, determining 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

```
tbl_policy <- policy_year_new(10, 2001)
tbl_claims <- claims_by_first_report(
  tbl_policy,
  frequency = 10,
  payment_severity = 100,
  lags = 1)
```

```
tbl_claims <- claims_by_link_ratio(
  tbl_claims,
  links = c(1.25, 1.1, 1.05),
  lags = 1:4)
```

```
claims_by_wait_time  claims_by_wait_time
```

Description

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

Usage

```
claims_by_wait_time(
  tbl_policy,
  claim_frequency,
  payment_frequency,
  occurrence_wait,
  report_wait,
  pay_wait,
  pay_severity,
  pay_only_positive = TRUE
)
```

Arguments

tbl_policy	A data frame of policy records
claim_frequency	Number of claims per policy; can be a distribution.
payment_frequency	Number of payments per claim; can be a distribution.
occurrence_wait	Time until occurrence for each claim; can be a distribution
report_wait	Time until report; can be a distribution.
pay_wait	Lag time between payments; can be a distribution.
pay_severity	Severity of each claim payment; can be a distribution.
pay_only_positive	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.

Value

A data frame, as follows:

- policy_effective_date** Date
- policy_expiration_date** Date
- exposure** double
- policyholder_id** integer
- claim_id** integer
- occurrence_date** Date
- report_date** Date
- number_of_payments** integer
- payment_date** Date
- payment_amount** double

iminator	<i>iminator</i>
----------	-----------------

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 by 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.

policies_grow	<i>Simulate policy growth</i>
---------------	-------------------------------

Description

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

Usage

```
policies_grow(tbl_policy, growth)
```

Arguments

- tbl_policy Data frame of policy data
- growth Scalar value greater than or equal to zero

Value

A data frame, as follows:

policy_effective_date Date
policy_expiration_date Date
exposure double
policyholder_id integer

policies_renew	<i>Simulate policy renewal</i>
----------------	--------------------------------

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

```
policies_renew(tbl_policy, retention)
```

Arguments

tbl_policy Data frame of policy data
retention Scalar value greater than or equal to zero

Value

A data frame, as follows:

policy_effective_date Date
policy_expiration_date Date
exposure double
policyholder_id integer

policies_simulate *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

```
policies_simulate(  
  n,  
  policy_years,  
  num_years,  
  exposure = 1,  
  retention = 1,  
  growth = 0,  
  start_id = 1,  
  additional_columns  
)
```

Arguments

n	An integer giving the number of policies in the first year
policy_years	A vector of integers in sequence
num_years	The number of years to simulate. If 'policy_years' 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
start_id	Integer of the first number in the policy ID sequence
additional_columns	A list of additional column names and values

Value

A data frame of policy data

policy_year_increment *Incremental a policy year*

Description

Given a policy data frame, this will combine the policies_grow and policies_renew functions to produce a subsequent policy year.

Usage

```
policy_year_increment(tbl_policy, retention, growth)
```

Arguments

tbl_policy	A policy data frame
retention	Scalar renewal rate
growth	Scalar growth rate

Value

Policy data frame

policy_year_new *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

```
policy_year_new(n, policy_year, exposure = 1, start_id = 1, additional_columns)
```

Arguments

n	The number of policies to generate
policy_year	Scalar integer indicating the policy year to generate
exposure	Vector of exposures
start_id	Integer of the first number in the policy ID sequence
additional_columns	A list of additional 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

Index

claims_by_first_report, 2

claims_by_link_ratio, 3

claims_by_wait_time, 4

imaginator, 5

policies_grow, 5

policies_renew, 6

policies_simulate, 7

policy_year_increment, 8

policy_year_new, 8