Package 'NetSimR'

September 15, 2026

Type Package

Title Actuarial Functions for Non-Life Insurance Modelling

Version 0.3.0

Description Assists actuaries and other insurance modellers in pricing, reserving and capital modelling for non-life insurance and reinsurance modelling. Provides functions that help model excess levels, capping and pure Incurred but not reported claims (pure IBNR).
Includes capped mean, exposure curves and increased limit factor curves (ILFs) for LogNormal, Gamma, Pareto, Sliced LogNormal-Pareto and Sliced Gamma-Pareto distributions.
Includes mean, probability density function (pdf), cumulative probability function (cdf) and inverse cumulative probability function for Sliced LogNormal-Pareto and Sliced Gamma-Pareto distributions.
Includes calculating pure IBNR exposure with LogNormal and Gamma distribution for reporting delay.
Includes three 'shiny' tools, one to simulate insurance claims applying reinsurance structures, fit generalised linear models and fit claims frequency or severity distributions.
Methods used in the package refer to
Free for All by Yiannis Parizas (2023) <https://www.theactuary.com/2023/03/02/free-all>;
Escaping the triangle by Yiannis Parizas (2019) <https://www.theactuary.com/features/2019/06/2019/06/05/escaping-triangle>;
Take to excess by Yiannis Parizas (2019) <https://www.theactuary.com/features/2019/03/2019/03/06/taken-excess>.

License GPL-3

URL https://github.com/NetSimAnalytics/NetSimR

BugReports https://github.com/NetSimAnalytics/NetSimR/issues

Encoding UTF-8

Language en-GB

Imports shiny (>= 1.8.1), base64enc, bslib (>= 0.9.0), future (>= 1.17.0), parallel, methods, stats, utils, graphics, grDevices

Suggests knitr, rmarkdown, crch, testthat, DBI, RMySQL, RODBC, RPostgreSQL, RSQLite

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Yiannis Parizas [aut, cre]

Maintainer Yiannis Parizas <yiannis.parizas@gmail.com>

Repository CRAN

Date/Publication 2026-09-15 11:30:13 UTC

Contents

apply_deductible_limit . . . 3
dSlicedGammaPareto . . . 4
dSlicedLNormPareto . . . 5
erf . . . 6
ExposureCurveGamma . . . 6
ExposureCurveLNorm . . . 7
ExposureCurvePareto . . . 8
ExposureCurveSlicedGammaPareto . . . 8
ExposureCurveSlicedLNormPareto . . . 9
GammaCappedMean . . . 10
IGamma . . . 11
ILFGamma . . . 12
ILFLNorm . . . 13
ILFPareto . . . 14
ILFSlicedGammaPareto . . . 15
ILFSlicedLNormPareto . . . 16
LNormCappedMean . . . 17
ParetoCappedMean . . . 17
ParetoCappedMeanCalc . . . 18
pSlicedGammaPareto . . . 19
pSlicedLNormPareto . . . 20
PureIBNRGamma . . . 21
PureIBNRLNorm . . . 22
qSlicedGammaPareto . . . 23
qSlicedLNormPareto . . . 24
run_shiny_distribution_fitting_tool . . . 25
run_shiny_glm_fitting_tool . . . 26
run_shiny_simulator . . . 26
simulate_claims . . . 27
simulate_function . . . 30

SlicedGammaParetoCappedMean 34
SlicedGammaParetoMean 35
SlicedLNormParetoCappedMean 36
SlicedLNormParetoMean 37

Index **38**

`apply_deductible_limit` *Apply a deductible and limit to claims*

Description

Apply a deductible and limit to claims

Usage

```
apply_deductible_limit(
  gross_claims_data,
  reinsurance_structure,
  deductible,
  limit
)
```

Arguments

`gross_claims_data`	A vector of Claims.
`reinsurance_structure`	The chosen reinsurance structure, a single string. Options are: 'No Reinsurance Structure', 'Unlimited Layer', 'Limited Layer', 'Exclude Layer'; anything else is an error.
`deductible`	The deductible of the reinsurance structure, zero or more. Not used by 'No Reinsurance Structure'.
`limit`	The limit of the reinsurance structure, zero or more. Used only by 'Limited Layer' and 'Exclude Layer'.

Value

A vector with one value per claim: for 'Unlimited Layer' and 'Limited Layer', the amount ceded to the layer; for 'Exclude Layer', the claims with the layer taken out (the claims less what a 'Limited Layer' with the same deductible and limit would cede); for 'No Reinsurance Structure', the claims unchanged. Stops with an error when a deductible or limit the structure uses is negative.

Examples

```
apply_deductible_limit(c(100, 50, 20), 'Limited Layer', 40, 20)
apply_deductible_limit(c(100, 50, 20), 'Limited Layer', 10, 30)
apply_deductible_limit(c(100, 50, 20), 'Exclude Layer', 40, 20)
```

`dSlicedGammaPareto` *The probability density function (pdf) of a Sliced Gamma Pareto severity distribution*

Description

The probability density function (pdf) of a Sliced Gamma Pareto severity distribution

Usage

```
dSlicedGammaPareto(x, GShape, GRate, SlicePoint, PShape)
```

Arguments

`x`	A real number - the claim amount where the probability density function (pdf) will be evaluated. The pdf is 0 for negative `x`.
`GShape`	A positive real number - the shape parameter of the attritional Claim Severity's Gamma distribution.
`GRate`	A positive real number - the rate parameter of the attritional Claim Severity's Gamma distribution.
`SlicePoint`	A positive real number - the slice point and the scale parameter of the tail Claim Severity's Pareto distribution. An infinite slice point gives the Gamma distribution.
`PShape`	A positive real number - the shape parameter of the tail Claim Severity's Pareto distribution.

Value

The value of the probability density function (pdf) at `x` with an attritional claim Gamma distribution with parameters `GShape` and `GRate` and a large claim Pareto distribution with parameters `SlicePoint` and `PShape`. A non-numeric argument or a non-positive parameter is an error; `NA` values give `NA`.

See Also

Other sliced distribution functions: `SlicedGammaParetoMean()`, `SlicedLNormParetoMean()`, `dSlicedLNormPareto()`, `pSlicedGammaPareto()`, `pSlicedLNormPareto()`, `qSlicedGammaPareto()`, `qSlicedLNormPareto()`

Examples

```
dSlicedGammaPareto(3000,1,0.0005,1000,1.2)
dSlicedGammaPareto(1000,1.1,0.0006,2000,1.6)
dSlicedGammaPareto(2000,1.2,0.0004,3000,1.4)
```

`dSlicedLNormPareto` *The probability density function (pdf) of a Sliced LogNormal Pareto severity distribution*

Description

The probability density function (pdf) of a Sliced LogNormal Pareto severity distribution

Usage

```
dSlicedLNormPareto(x, mu, sigma, SlicePoint, shape)
```

Arguments

`x`	A real number - the claim amount where the probability density function (pdf) will be evaluated. The pdf is 0 for negative `x`.
`mu`	A real number - the first parameter of the attritional Claim Severity's LogNormal distribution.
`sigma`	A positive real number - the second parameter of the attritional Claim Severity's LogNormal distribution.
`SlicePoint`	A positive real number - the slice point and the scale parameter of the Claim Severity's Pareto distribution. An infinite slice point gives the LogNormal distribution.
`shape`	A positive real number - the shape parameter of the Claim Severity's Pareto distribution.

Value

The value of the probability density function (pdf) at `x` with an attritional claim LogNormal distribution with parameters `mu` and `sigma` and a large claim Pareto distribution with parameters `SlicePoint` and `shape`. A non-numeric argument or a non-positive `sigma`, `SlicePoint` or `shape` is an error; `NA` values give `NA`.

See Also

Other sliced distribution functions: `SlicedGammaParetoMean()`, `SlicedLNormParetoMean()`, `dSlicedGammaPareto()`, `pSlicedGammaPareto()`, `pSlicedLNormPareto()`, `qSlicedGammaPareto()`, `qSlicedLNormPareto()`

Examples

```
dSlicedLNormPareto(1200,6,1.5,1000,1.2)
dSlicedLNormPareto(4000,7,1.6,3000,1.4)
```

erf *Error function*

Description

Error function

Usage

```
erf(x)
```

Arguments

x	A real number.

Value

The value of the error function at x. A non-numeric x is an error.

See Also

LNormCappedMean

Examples

```
erf(0.1)
erf(0.5)
```

ExposureCurveGamma *Exposure Curve from a Gamma severity distribution*

Description

Exposure Curve from a Gamma severity distribution

Usage

```
ExposureCurveGamma(x, shape, rate)
```

Arguments

x	A non-negative real number - the claim amount where the exposure curve will be evaluated.
shape	A positive real number - the shape parameter of the Claim Severity's Gamma distribution.
rate	A positive real number - the rate parameter of the Claim Severity's Gamma distribution.

Value

The value of the Exposure curve at `x` with Claim Severity from a Gamma distribution with parameters `shape` and `rate`.

See Also

Other exposure curve functions: `ExposureCurveLNorm()`, `ExposureCurvePareto()`, `ExposureCurveSlicedGammaPareto(`
`ExposureCurveSlicedLNormPareto()`

Examples

```
ExposureCurveGamma(700,1,0.0005)
ExposureCurveGamma(1000,1.5,0.0006)
```

`ExposureCurveLNorm` *Exposure Curve from LogNormal a severity distribution*

Description

Exposure Curve from LogNormal a severity distribution

Usage

```
ExposureCurveLNorm(x, mu, sigma)
```

Arguments

`x`	A non-negative real number - the claim amount where the exposure curve will be evaluated.
`mu`	A real number - the first parameter of the Claim Severity's LogNormal distribution.
`sigma`	A positive real number - the second parameter of the Claim Severity's LogNormal distribution.

Value

The value of the Exposure curve at `x` with Claim Severity from a LogNormal distribution with parameters `mu` and `sigma`.

See Also

Other exposure curve functions: `ExposureCurveGamma()`, `ExposureCurvePareto()`, `ExposureCurveSlicedGammaPareto(`
`ExposureCurveSlicedLNormPareto()`

Examples

```
ExposureCurveLNorm(2000,6,1.5)
ExposureCurveLNorm(1000,5,1.6)
```

ExposureCurvePareto *Exposure Curve from a Pareto severity distribution*

Description

Exposure Curve from a Pareto severity distribution

Usage

```
ExposureCurvePareto(x, scale, shape)
```

Arguments

x	A non-negative real number - the claim amount where the exposure curve will be evaluated.
scale	A positive real number - the scale parameter of the Claim Severity's Pareto distribution.
shape	A positive real number - the shape parameter of the Claim Severity's Pareto distribution.

Value

The value of the Exposure curve at x with Claim Severity from a Pareto distribution with parameters scale and shape. The exposure curve divides by the mean, which is infinite when shape <= 1; the function returns 0 in that case.

See Also

Other exposure curve functions: ExposureCurveGamma(), ExposureCurveLNorm(), ExposureCurveSlicedGammaPareto()
ExposureCurveSlicedLNormPareto()

Examples

```
ExposureCurvePareto(700,500,1.2)
ExposureCurvePareto(20000,200,1.1)
```

ExposureCurveSlicedGammaPareto *Exposure Curve from a Sliced Gamma Pareto severity distribution*

Description

Exposure Curve from a Sliced Gamma Pareto severity distribution

Usage

```
ExposureCurveSlicedGammaPareto(x, GShape, GRate, SlicePoint, PShape)
```

Arguments

x	A non-negative real number - the claim amount where the exposure curve will be evaluated.
GShape	A positive real number - the shape parameter of the Claim Severity's Gamma distribution.
GRate	A positive real number - the rate parameter of the Claim Severity's Gamma distribution.
SlicePoint	A positive real number - the slice point and the scale parameter of the Claim Severity's Pareto distribution. An infinite slice point gives the Gamma distribution.
PShape	A positive real number - the shape parameter of the Claim Severity's Pareto distribution.

Value

The value of the Exposure curve at x with an attritional claim Gamma distribution with parameters GShape and GRate and a large claim Pareto distribution with parameters SlicePoint and PShape. The exposure curve divides by the mean, which is infinite when PShape <= 1 (and SlicePoint is finite); the function returns 0 in that case.

See Also

Other exposure curve functions: ExposureCurveGamma(), ExposureCurveLNorm(), ExposureCurvePareto(), ExposureCurveSlicedLNormPareto()

Examples

```
ExposureCurveSlicedGammaPareto(3000,1,0.0005,1000,1.2)
ExposureCurveSlicedGammaPareto(1000,1.1,0.0006,2000,1.6)
ExposureCurveSlicedGammaPareto(2000,1.2,0.0004,3000,1.4)
```

ExposureCurveSlicedLNormPareto

Exposure Curve from a Sliced LogNormal Pareto severity distribution

---

Description

Exposure Curve from a Sliced LogNormal Pareto severity distribution

Usage

```
ExposureCurveSlicedLNormPareto(x, mu, sigma, SlicePoint, shape)
```

Arguments

x	A non-negative real number - the claim amount where the exposure curve will be evaluated.
mu	A real number - the first parameter of the attritional Claim Severity's LogNormal distribution.
sigma	A positive real number - the second parameter of the attritional Claim Severity's LogNormal distribution.
SlicePoint	A positive real number - the slice point and the scale parameter of the tail Claim Severity's Pareto distribution. An infinite slice point gives the LogNormal distribution.
shape	A positive real number - the shape parameter of the tail Claim Severity's Pareto distribution.

Value

The value of the Exposure curve at x with an attritional claim LogNormal distribution with parameters mu and sigma and a large claim Pareto distribution with parameters SlicePoint and shape. The exposure curve divides by the mean, which is infinite when shape <= 1 (and SlicePoint is finite); the function returns 0 in that case.

See Also

Other exposure curve functions: ExposureCurveGamma(), ExposureCurveLNorm(), ExposureCurvePareto(), ExposureCurveSlicedGammaPareto()

Examples

```
ExposureCurveSlicedLNormPareto(1200,6,1.5,1000,1.2)
ExposureCurveSlicedLNormPareto(4000,7,1.6,3000,1.4)
```

GammaCappedMean *Gamma capped mean*

Description

Gamma capped mean

Usage

```
GammaCappedMean(cap, shape, rate)
```

Arguments

cap	A non-negative real number - the claim severity cap.
shape	A positive real number - the shape parameter of the Claim Severity's Gamma distribution.
rate	A positive real number - the rate parameter of the Claim Severity's Gamma distribution.

Value

The mean of the claim severity capped at cap with a Gamma distribution with parameters shape and rate. The arguments are recycled to a common length. A non-numeric argument, a negative cap or a non-positive shape or rate is an error; NA values give NA.

See Also

Other capped mean functions: LNormCappedMean(), ParetoCappedMean(), ParetoCappedMeanCalc(), SlicedGammaParetoCappedMean(), SlicedLNormParetoCappedMean()

Examples

```
GammaCappedMean(700,1,0.0005)
GammaCappedMean(1000,1.5,0.0006)
```

IGamma *Upper incomplete gamma function*

Description

Upper incomplete gamma function

Usage

```
IGamma(a, x)
```

Arguments

a	A positive real number - the shape parameter.
x	A non-negative real number.

Value

The value of the upper incomplete gamma function at x with shape parameter a, i.e. gamma(a) * pgamma(x, a, lower.tail = FALSE). The arguments are recycled to a common length. A non-numeric or non-positive a or a negative x is an error.

See Also

GammaCappedMean

Examples

```
IGamma(1,1)
IGamma(0.1,2)
```

ILFGamma *Increased Limit Factor Curve from a Gamma severity distribution*

Description

Increased Limit Factor Curve from a Gamma severity distribution

Usage

```
ILFGamma(xLow, xHigh, shape, rate)
```

Arguments

xLow	A non-negative real number - the claim amount where the Increased Limit Factor Curve will be evaluated from.
xHigh	A non-negative real number - the claim amount where the Increased Limit Factor Curve will be evaluated to.
shape	A positive real number - the shape parameter of the Claim Severity's Gamma distribution.
rate	A positive real number - the rate parameter of the Claim Severity's Gamma distribution.

Value

The value of the Increased Limit Factor curve from xLow to xHigh with Claim Severity from a Gamma distribution with parameters shape and rate.

See Also

Other ILF functions: ILFLNorm(), ILFPareto(), ILFSlicedGammaPareto(), ILFSlicedLNormPareto()

Examples

```
ILFGamma(700,1000,1,0.0005)
ILFGamma(1000,1200,1.5,0.0006)
```

`ILFLNorm` *Increased Limit Factor Curve from a LogNormal severity distribution*

Description

Increased Limit Factor Curve from a LogNormal severity distribution

Usage

```
ILFLNorm(xLow, xHigh, mu, sigma)
```

Arguments

`xLow`	A non-negative real number - the claim amount where the Increased Limit Factor Curve will be evaluated from.
`xHigh`	A non-negative real number - the claim amount where the Increased Limit Factor Curve will be evaluated to.
`mu`	A real number - the first parameter of the Claim Severity's LogNormal distribution.
`sigma`	A positive real number - the second parameter of the Claim Severity's LogNormal distribution.

Value

The value of the Increased Limit Factor curve from `xLow` to `xHigh` with Claim Severity from a LogNormal distribution with parameters `mu` and `sigma`.

See Also

Other ILF functions: `ILFGamma()`, `ILFPareto()`, `ILFSlicedGammaPareto()`, `ILFSlicedLNormPareto()`

Examples

```
ILFLNorm(1000,2000,6,1.5)
ILFLNorm(1000,1500,5,1.6)
```

`ILFPareto` *Increased Limit Factor Curve from a Pareto severity distribution*

Description

Increased Limit Factor Curve from a Pareto severity distribution

Usage

```
ILFPareto(xLow, xHigh, scale, shape)
```

Arguments

`xLow`	A non-negative real number - the claim amount where the Increased Limit Factor Curve will be evaluated from.
`xHigh`	A non-negative real number - the claim amount where the Increased Limit Factor Curve will be evaluated to.
`scale`	A positive real number - the scale parameter of the Claim Severity's Pareto distribution.
`shape`	A positive real number - the shape parameter of the Claim Severity's Pareto distribution.

Value

The value of the Increased Limit Factor curve from `xLow` to `xHigh` with Claim Severity from a Pareto distribution with parameters `scale` and `shape`.

See Also

Other ILF functions: `ILFGamma()`, `ILFLNorm()`, `ILFSlicedGammaPareto()`, `ILFSlicedLNormPareto()`

Examples

```
ILFPareto(700,1200,500,1.2)
ILFPareto(1200,20000,200,1.1)
```

`ILFSlicedGammaPareto` *Increased Limit Factor Curve from a Sliced Gamma Pareto severity distribution*

Description

Increased Limit Factor Curve from a Sliced Gamma Pareto severity distribution

Usage

```
ILFSlicedGammaPareto(xLow, xHigh, GShape, GRate, SlicePoint, PShape)
```

Arguments

`xLow`	A non-negative real number - the claim amount where the Limit Factor Curve will be evaluated from.
`xHigh`	A non-negative real number - the claim amount where the Limit Factor Curve will be evaluated to.
`GShape`	A positive real number - the shape parameter of the attritional Claim Severity's Gamma distribution.
`GRate`	A positive real number - the rate parameter of the attritional Claim Severity's Gamma distribution.
`SlicePoint`	A positive real number - the slice point and the scale parameter of the tail Claim Severity's Pareto distribution. An infinite slice point gives the Gamma distribution.
`PShape`	A positive real number - the shape parameter of the tail Claim Severity's Pareto distribution.

Value

The value of the Increased Limit Factor curve from `xLow` to `xHigh` with an attritional claim Gamma distribution with parameters `GShape` and `GRate` and a large claim Pareto distribution with parameters `SlicePoint` and `PShape`.

See Also

Other ILF functions: `ILFGamma()`, `ILFLNorm()`, `ILFPareto()`, `ILFSlicedLNormPareto()`

Examples

```
ILFSlicedGammaPareto(2000,3000,1,0.0005,1000,1.2)
ILFSlicedGammaPareto(800,1000,1.1,0.0006,2000,1.6)
ILFSlicedGammaPareto(1200,2000,1.2,0.0004,3000,1.4)
```

`ILFSlicedLNormPareto` *Increased Limit Factor Curve from a Sliced LogNormal Pareto severity distribution*

Description

Increased Limit Factor Curve from a Sliced LogNormal Pareto severity distribution

Usage

```
ILFSlicedLNormPareto(xLow, xHigh, mu, sigma, SlicePoint, shape)
```

Arguments

`xLow`	A non-negative real number - the claim amount where the Limit Factor Curve will be evaluated from.
`xHigh`	A non-negative real number - the claim amount where the Limit Factor Curve will be evaluated to.
`mu`	A real number - the first parameter of the attritional Claim Severity's LogNormal distribution.
`sigma`	A positive real number - the second parameter of the attritional Claim Severity's LogNormal distribution.
`SlicePoint`	A positive real number - the slice point and the scale parameter of the tail Claim Severity's Pareto distribution. An infinite slice point gives the LogNormal distribution.
`shape`	A positive real number - the shape parameter of the tail Claim Severity's Pareto distribution.

Value

The value of the Increased Limit Factor curve from `xLow` to `xHigh` with an attritional claim LogNormal distribution with parameters `mu` and `sigma` and a large claim Pareto distribution with parameters `SlicePoint` and `shape`.

See Also

Other ILF functions: `ILFGamma()`, `ILFLNorm()`, `ILFPareto()`, `ILFSlicedGammaPareto()`

Examples

```
ILFSlicedLNormPareto(800,1200,6,1.5,1000,1.2)
ILFSlicedLNormPareto(2000,4000,7,1.6,3000,1.4)
```

LNormCappedMean *Lognormal capped mean*

Description

Lognormal capped mean

Usage

```
LNormCappedMean(cap, mu, sigma)
```

Arguments

cap	A non-negative real number - the claim severity cap.
mu	A real number - the first parameter of the Claim Severity's LogNormal distribution.
sigma	A positive real number - the second parameter of the Claim Severity's LogNormal distribution.

Value

The mean of the claim severity capped at cap with a LogNormal distribution with parameters mu and sigma. The arguments are recycled to a common length. A non-numeric argument, a negative cap or a non-positive sigma is an error; NA values give NA.

See Also

Other capped mean functions: GammaCappedMean(), ParetoCappedMean(), ParetoCappedMeanCalc(), SlicedGammaParetoCappedMean(), SlicedLNormParetoCappedMean()

Examples

```
LNormCappedMean(2000,6,1.5)
LNormCappedMean(1000,5,1.6)
```

ParetoCappedMean *Pareto capped mean*

Description

Pareto capped mean

Usage

```
ParetoCappedMean(cap, scale, shape)
```

Arguments

cap	A non-negative real number - the claim severity cap.
scale	A positive real number - the scale parameter of the Claim Severity's Pareto distribution.
shape	A positive real number - the shape parameter of the Claim Severity's Pareto distribution.

Value

The mean of the claim severity capped at cap with a Pareto distribution with parameters scale and shape. A cap at or below scale is returned unchanged, as no claim is smaller than scale. The arguments are recycled to a common length. A non-numeric argument, a negative cap or a non-positive scale or shape is an error; NA values give NA.

See Also

Other capped mean functions: GammaCappedMean(), LNormCappedMean(), ParetoCappedMeanCalc(), SlicedGammaParetoCappedMean(), SlicedLNormParetoCappedMean()

Examples

```
ParetoCappedMean(600,200,1.2)
ParetoCappedMean(800,100,1)
ParetoCappedMean(1000,500,0.8)
ParetoCappedMean(50,100,2)
```

ParetoCappedMeanCalc *Pareto capped mean intermediary calculation*

Description

Pareto capped mean intermediary calculation

Usage

```
ParetoCappedMeanCalc(cap, scale, shape)
```

Arguments

cap	A non-negative real number - the claim severity cap.
scale	A positive real number - the scale parameter of the Claim Severity's Pareto distribution.
shape	A positive real number - the shape parameter of the Claim Severity's Pareto distribution.

Value

An interim calculation for the mean of the claim severity capped at `cap` with a Pareto distribution with parameters `scale` and `shape`. It is the closed form for `cap >= scale` and `shape != 1`; use `ParetoCappedMean` for the capped mean itself. The arguments are recycled to a common length.

See Also

Other capped mean functions: `GammaCappedMean()`, `LNormCappedMean()`, `ParetoCappedMean()`, `SlicedGammaParetoCappedMean()`, `SlicedLNormParetoCappedMean()`

Examples

```
ParetoCappedMeanCalc(800,100,1.1)
ParetoCappedMeanCalc(1000,500,0.9)
```

---

pSlicedGammaPareto *The cumulative density function (cdf) of a Sliced Gamma-Pareto severity distribution*

Description

The cumulative density function (cdf) of a Sliced Gamma-Pareto severity distribution

Usage

```
pSlicedGammaPareto(x, GShape, GRate, SlicePoint, PShape)
```

Arguments

`x`	A real number - the claim amount where the cumulative density function (cdf) will be evaluated. The cdf is 0 for negative `x`.
`GShape`	A positive real number - the shape parameter of the attritional Claim Severity's Gamma distribution.
`GRate`	A positive real number - the rate parameter of the attritional Claim Severity's Gamma distribution.
`SlicePoint`	A positive real number - the slice point and the scale parameter of the tail Claim Severity's Pareto distribution. An infinite slice point gives the Gamma distribution.
`PShape`	A positive real number - the shape parameter of the tail Claim Severity's Pareto distribution.

Value

The value of the cumulative density function (cdf) at `x` with an attritional claim Gamma distribution with parameters `GShape` and `GRate` and a large claim Pareto distribution with parameters `SlicePoint` and `PShape`. A non-numeric argument or a non-positive parameter is an error; `NA` values give `NA`.

See Also

Other sliced distribution functions: SlicedGammaParetoMean(), SlicedLNormParetoMean(), dSlicedGammaPareto(), dSlicedLNormPareto(), pSlicedLNormPareto(), qSlicedGammaPareto(), qSlicedLNormPareto()

Examples

```
pSlicedGammaPareto(3000,1,0.0005,1000,1.2)
pSlicedGammaPareto(1000,1.1,0.0006,2000,1.6)
pSlicedGammaPareto(2000,1.2,0.0004,3000,1.4)
```

---

pSlicedLNormPareto *The cumulative density function (cdf) of a Sliced LogNormal Pareto severity distribution*

Description

The cumulative density function (cdf) of a Sliced LogNormal Pareto severity distribution

Usage

```
pSlicedLNormPareto(x, mu, sigma, SlicePoint, shape)
```

Arguments

x	A real number - the claim amount where the cumulative density function (cdf) will be evaluated. The cdf is 0 for negative x.
mu	A real number - the first parameter of the attritional Claim Severity's LogNormal distribution.
sigma	A positive real number - the second parameter of the attritional Claim Severity's LogNormal distribution.
SlicePoint	A positive real number - the slice point and the scale parameter of the tail Claim Severity's Pareto distribution. An infinite slice point gives the LogNormal distribution.
shape	A positive real number - the shape parameter of the tail Claim Severity's Pareto distribution.

Value

The value of the cumulative density function (cdf) at x with an attritional claim LogNormal distribution with parameters mu and sigma and a large claim Pareto distribution with parameters SlicePoint and shape. A non-numeric argument or a non-positive sigma, SlicePoint or shape is an error; NA values give NA.

See Also

Other sliced distribution functions: SlicedGammaParetoMean(), SlicedLNormParetoMean(), dSlicedGammaPareto(), dSlicedLNormPareto(), pSlicedGammaPareto(), qSlicedGammaPareto(), qSlicedLNormPareto()

Examples

```
pSlicedLNormPareto(1200,6,1.5,1000,1.2)
pSlicedLNormPareto(4000,7,1.6,3000,1.4)
```

`PureIBNRGamma` *Pure IBNR exposure from a Gamma reporting delay distribution*

Description

Durations are counted in days on each date's own calendar and clock: a `POSIXct` time of day counts as a fraction of a day, and daylight saving changes do not add fractions of a day, so `Date` and `POSIXct` dates (or a mix of them) give the same results.

Usage

```
PureIBNRGamma(IncDate, ExpDate, ValDate, shape, rate)
```

Arguments

`IncDate`	A `Date` or `POSIXct` vector - the inception dates of the periods. Numbers and character strings are not accepted; convert them with `as.Date()` first.
`ExpDate`	A `Date` or `POSIXct` vector - the expiry dates of the periods. Must not be before the inception dates.
`ValDate`	A `Date` or `POSIXct` date - the valuation date.
`shape`	A positive real number - the shape parameter of the reporting delay's Gamma distribution, with the delay measured in days.
`rate`	A positive real number - the rate parameter (per day) of the reporting delay's Gamma distribution.

Value

A data frame with the unearned and pure IBNR exposure of each period in days (`UnearnedDuration`, and `PureIBNRDuration` rounded to 2 decimals) and as proportions between 0 and 1 of the period's duration (`UnearnedDurationRatio` and `PureIBNRDurationRatio`, rounded to 5 decimals), where the reporting delay has a Gamma distribution with parameters `shape` and `rate`. The dates and parameters are recycled to a common length, one row each; lengths that do not recycle are an error.

See Also

Other pure IBNR functions: `PureIBNRLNorm()`

Examples

```
Dates = data.frame(
    inceptionDate = c("01/01/2006", "01/07/2006", "01/01/2007")
    ,expiryDate = c("31/12/2006", "30/06/2007", "31/12/2007")
)

Dates$inceptionDate<-as.POSIXct(Dates$inceptionDate, format="%d/%m/%Y")

Dates$expiryDate<-as.POSIXct(Dates$expiryDate, format="%d/%m/%Y")

ValuationDate<-as.POSIXct("30/10/2007", format="%d/%m/%Y")

PureIBNRGamma(Dates$inceptionDate,Dates$expiryDate,ValuationDate,7,0.15)
```

`PureIBNRLNorm` *Pure IBNR exposure from a LogNormal reporting delay distribution*

Description

Durations are counted in days on each date's own calendar and clock: a `POSIXct` time of day counts as a fraction of a day, and daylight saving changes do not add fractions of a day, so `Date` and `POSIXct` dates (or a mix of them) give the same results.

Usage

```
PureIBNRLNorm(IncDate, ExpDate, ValDate, mu, sigma)
```

Arguments

`IncDate`	A `Date` or `POSIXct` vector - the inception dates of the periods. Numbers and character strings are not accepted; convert them with `as.Date()` first.
`ExpDate`	A `Date` or `POSIXct` vector - the expiry dates of the periods. Must not be before the inception dates.
`ValDate`	A `Date` or `POSIXct` date - the valuation date.
`mu`	A real number - the first parameter of the reporting delay's LogNormal distribution, with the delay measured in days.
`sigma`	A positive real number - the second parameter of the reporting delay's LogNormal distribution.

Value

A data frame with the unearned and pure IBNR exposure of each period in days (`UnearnedDuration`, and `PureIBNRDuration` rounded to 2 decimals) and as proportions between 0 and 1 of the period's duration (`UnearnedDurationRatio` and `PureIBNRDurationRatio`, rounded to 5 decimals), where the reporting delay has a LogNormal distribution with parameters `mu` and `sigma`. The dates and parameters are recycled to a common length, one row each; lengths that do not recycle are an error.

See Also

Other pure IBNR functions: `PureIBNRGamma()`

Examples

```
Dates = data.frame(
    inceptionDate = c("01/01/2006", "01/07/2006", "01/01/2007")
    ,expiryDate = c("31/12/2006", "30/06/2007", "31/12/2007")
)

Dates$inceptionDate<-as.POSIXct(Dates$inceptionDate, format="%d/%m/%Y")

Dates$expiryDate<-as.POSIXct(Dates$expiryDate, format="%d/%m/%Y")

ValuationDate<-as.POSIXct("30/10/2007", format="%d/%m/%Y")

PureIBNRLNorm(Dates$inceptionDate,Dates$expiryDate,ValuationDate,4,1.5)
```

---

qSlicedGammaPareto *The inverse cumulative density function of a Sliced Gamma Pareto severity distribution*

Description

The inverse cumulative density function of a Sliced Gamma Pareto severity distribution

Usage

```
qSlicedGammaPareto(q, GShape, GRate, SlicePoint, PShape)
```

Arguments

`q`	A real number between 0 and 1 - the probability where the inverse cumulative density function will be evaluated. Values outside [0, 1] give `NaN`, as in `qgamma()`.
`GShape`	A positive real number - the shape parameter of the attritional Claim Severity's Gamma distribution.
`GRate`	A positive real number - the rate parameter of the attritional Claim Severity's Gamma distribution.
`SlicePoint`	A positive real number - the slice point and the scale parameter of the tail Claim Severity's Pareto distribution. An infinite slice point gives the Gamma distribution.
`PShape`	A positive real number - the shape parameter of the tail Claim Severity's Pareto distribution.

Value

The value of the inverse cumulative density function at q with an attritional claim Gamma distribution with parameters GShape and GRate and a large claim Pareto distribution with parameters SlicePoint and PShape. A non-numeric argument or a non-positive parameter is an error; NA values give NA.

See Also

Other sliced distribution functions: SlicedGammaParetoMean(), SlicedLNormParetoMean(), dSlicedGammaPareto(), dSlicedLNormPareto(), pSlicedGammaPareto(), pSlicedLNormPareto(), qSlicedLNormPareto()

Examples

```
qSlicedGammaPareto(0.5,1,0.0005,1000,1.2)
qSlicedGammaPareto(0.2,1.1,0.0006,2000,1.6)
qSlicedGammaPareto(0.8,1.2,0.0004,3000,1.4)
```

---

qSlicedLNormPareto *The inverse cumulative density function of a Sliced LogNormal Pareto severity distribution*

Description

The inverse cumulative density function of a Sliced LogNormal Pareto severity distribution

Usage

```
qSlicedLNormPareto(q, mu, sigma, SlicePoint, shape)
```

Arguments

q	A real number between 0 and 1 - the probability where the inverse cumulative density function will be evaluated. Values outside [0, 1] give NaN, as in qlnorm().
mu	A real number - the first parameter of the attritional Claim Severity's LogNormal distribution.
sigma	A positive real number - the second parameter of the attritional Claim Severity's LogNormal distribution.
SlicePoint	A positive real number - the slice point and the scale parameter of the tail Claim Severity's Pareto distribution. An infinite slice point gives the LogNormal distribution.
shape	A positive real number - the shape parameter of the tail Claim Severity's Pareto distribution.

Value

The value of the inverse cumulative density function at q with an attritional claim LogNormal distribution with parameters mu and sigma and a large claim Pareto distribution with parameters SlicePoint and shape. A non-numeric argument or a non-positive sigma, SlicePoint or shape is an error; NA values give NA.

See Also

Other sliced distribution functions: SlicedGammaParetoMean(), SlicedLNormParetoMean(), dSlicedGammaPareto(), dSlicedLNormPareto(), pSlicedGammaPareto(), pSlicedLNormPareto(), qSlicedGammaPareto()

Examples

```
qSlicedLNormPareto(0.5,6,1.5,1000,1.2)
qSlicedLNormPareto(0.7,7,1.6,3000,1.4)
```

---

run_shiny_distribution_fitting_tool

A function to run the distribution fitting tool application

Description

A function to run the distribution fitting tool application

Usage

```
run_shiny_distribution_fitting_tool()
```

Value

A shiny app object. Printing it, as happens when the function is called at the console, opens the application; it can also be passed to shiny::runApp().

Examples

```
if (interactive()) {
  run_shiny_distribution_fitting_tool()
}
```

run_shiny_glm_fitting_tool *A function to run the GLM fitting tool application*

Description

A function to run the GLM fitting tool application

Usage

```
run_shiny_glm_fitting_tool()
```

Value

A shiny app object. Printing it, as happens when the function is called at the console, opens the application; it can also be passed to shiny::runApp().

Examples

```
if (interactive()) {
  run_shiny_glm_fitting_tool()
}
```

run_shiny_simulator *A function to run the shiny simulator application*

Description

A function to run the shiny simulator application

Usage

```
run_shiny_simulator()
```

Value

A shiny app object (class `shiny.appobj`). Printing it, as happens when `run_shiny_simulator()` is called at the console, opens the app; pass it to `shiny::runApp()` to choose options such as the port.

See Also

`simulate_claims` and `simulate_function`, which run the same model from R code.

Examples

```
if (interactive()) {
  run_shiny_simulator()
}
```

`simulate_claims` *Simulate claims with a frequency-severity model*

Description

A simpler interface to `simulate_function`, with short argument names and defaults for everything optional. Leaving an option out (NULL) switches the feature off: no seed, no Pareto slices, no cap, no reinstatement limit.

Usage

```
simulate_claims(
  n_sims,
  frequency,
  frequency_params,
  severity,
  severity_params,
  seed = NULL,
  truncate_at_zero = FALSE,
  pareto_thresholds = NULL,
  pareto_alphas = NULL,
  severity_cap = NULL,
  eel_layer = "none",
  eel_deductible = NULL,
  eel_limit = NULL,
  eel_reinstatements = NULL,
  agg_layer = "none",
  agg_deductible = NULL,
  agg_limit = NULL,
  parallel = FALSE,
  chunk_size = NULL,
  gross = TRUE,
  shortcuts = TRUE,
  progress = NULL
)
```

Arguments

`n_sims`	Number of simulations (e.g. years).
`frequency`	Name of the claim count distribution; see Details.

`frequency_params`
: Parameters of the claim count distribution, either all unnamed in the order of Details or all named.

`severity`
: Name of the claim size distribution; see Details.

`severity_params`
: Parameters of the claim size distribution, either all unnamed in the order of Details or all named.

`seed`
: A whole number for a reproducible run. NULL (the default) uses the current random number stream, so `set.seed()` before the call also makes the run reproducible.

`truncate_at_zero`
: TRUE to draw Normal claim sizes from the Normal distribution truncated at zero, so no claim is negative. Only used with the Normal severity.

`pareto_thresholds`
: Increasing claim sizes above which the severity tail is replaced by Pareto slices, one per slice (at most six). NULL (the default) for no slices.

`pareto_alphas`
: The Pareto alpha of each slice, one per threshold.

`severity_cap`
: The largest amount a single claim can reach, or NULL (the default) for no cap.

`eel_layer`
: The each-and-every-loss layer: "none" (the default), "unlimited", "limited" or "exclude".

`eel_deductible`
: The deductible of the each-and-every-loss layer.

`eel_limit`
: The limit of a "limited" or "exclude" each-and-every-loss layer.

`eel_reinstatements`
: The number of reinstatements of a "limited" each-and-every-loss layer, so it pays at most `(eel_reinstatements + 1) * eel_limit` per simulation. NULL (the default) for unlimited reinstatements.

`agg_layer`
: The aggregate layer: "none" (the default), "unlimited", "limited" or "exclude".

`agg_deductible`
: The deductible of the aggregate layer.

`agg_limit`
: The limit of a "limited" or "exclude" aggregate layer.

`parallel`
: TRUE to run the chunks of simulations on parallel workers. Results are the same as a sequential run.

`chunk_size`
: The number of simulations per vectorised batch; NULL (the default) chooses it from the expected number of claims, aiming at about a million claims per batch but never fewer than 100 simulations, so batches are larger when the mean frequency exceeds 10,000. Results with a fixed seed depend on it.

`gross`
: TRUE (the default) to return the gross totals before the layers. FALSE allows a much faster run with an "unlimited" or "limited" each-and-every-loss layer, by drawing only the claims that reach it.

`shortcuts`
: TRUE (the default) to use exact shortcuts where the settings allow; see `simulate_function`.

`progress`
: An optional function called after each chunk of a sequential run with the fraction done and a short description.

Details

Distributions are chosen by name; case, spaces and underscores are ignored, so "Negative Binomial", "negative_binomial" and "Negative_Binomial" are the same. Their parameters are given in the order below, or named with these names:

Distribution	Type	Parameters
Poisson	frequency	lambda
Negative_Binomial	frequency	r, beta (Gamma shape and scale of the Poisson mean)
Binomial	frequency	n, p
Fixed_number_of_Counts	frequency	count
Normal	severity	mean, sd
LogNormal	severity	meanlog, sdlog
Gamma	severity	shape, scale
Exponential	severity	rate
Pareto	severity	alpha, x_m (minimum)
Fixed_Severity	severity	amount

Layers are "none", "unlimited" (everything above the deductible), "limited" (the limit excess of the deductible) or "exclude" (the claims with that layer removed). The each-and-every-loss layer applies to every claim; the aggregate layer applies to each simulation's total after the each-and-every-loss layer. The order is: severity cap -> each-and-every-loss layer per claim -> sum over the simulation -> aggregate deductible -> aggregate limit and reinstatement capacity. So with a "limited" each-and-every-loss layer and an "unlimited" or "limited" aggregate layer, the ceded total is `min(max(S - agg_deductible, 0), agg_limit, (eel_reinstatements + 1) * eel_limit)`, where S is the sum of the layer's recoveries; three claims of 100 through a layer of 100 excess of 0 with no reinstatements and an aggregate deductible of 50 cede 100. An "exclude" aggregate layer is taken out of the recoveries after the reinstatement capacity has capped them. See `simulate_function` for the details.

Value

A data frame with one row per simulation, at full precision: `claim_counts`, `total_claims` (after the layers), `gross_claims` (before them, unless `gross = FALSE`) and, with limited reinstatements, `number_of_reinstatements_used`: the layer's recoveries in the simulation (after the aggregate deductible and limit, if any) divided by `eel_limit`, capped at `eel_reinstatements`.

See Also

`simulate_function`, which this calls, and `run_shiny_simulator` for the same model in an app.

Examples

```
# 10,000 years of Poisson claim counts with Log-Normal claim sizes
claims <- simulate_claims(
  10000, frequency = "Poisson", frequency_params = 3,
  severity = "LogNormal", severity_params = c(meanlog = 8, sdlog = 1.5), seed = 1
)
summary(claims$total_claims)
```

```
# a Pareto tail above 100,000, and a layer of 50,000 excess of 20,000 on each
# claim with two reinstatements
ceded <- simulate_claims(
  10000, "Poisson", 3, "LogNormal", c(8, 1.5), seed = 1,
  pareto_thresholds = 100000, pareto_alphas = 1.5,
  eel_layer = "limited", eel_deductible = 20000, eel_limit = 50000,
  eel_reinstatements = 2
)
mean(ceded$total_claims)
```

`simulate_function` *Simulate insurance claims with reinsurance structures*

Description

A function to simulate frequency - severity of insurance claims using chunked vectorisation. The function applies severity cap, reinsurance structure for each and every loss claim, reinsurance structure for aggregate claims, and allows for piecewise Pareto slices

Usage

```
simulate_function(
  numOfSimulations,
  freq_params,
  sev_params,
  seedSetBinary = !is.null(seedValue),
  seedValue = NULL,
  freqDistr,
  sevDistr,
  paretoSlice = FALSE,
  pareto_slice_times = NULL,
  slice_pareto_alphas = NULL,
  slice_pareto_x_ms = NULL,
  sevCapBinary = FALSE,
  sev_cap_amount = NULL,
  reinsuranceStructureEEL = "No Reinsurance Structure",
  reinsurance_structure_eel_dedctible_amount = NULL,
  reinsurance_structure_eel_limit_amount = NULL,
  reinsuranceStructureAL = "No Reinsurance Structure",
  reinsurance_structure_al_dedctible_amount = NULL,
  reinsurance_structure_al_limit_amount = NULL,
  reinsuranceStructureLimitedReinstatements = FALSE,
  reinsuranceStructureReinstatementLimit = NULL,
  multiprocessing = FALSE,
  sevTruncateAtZero = FALSE,
  chunk_size = NULL,
  gross = TRUE,
  shortcuts = TRUE,
```

```
  progress = NULL
)
```

Arguments

numOfSimulations	The number of simulations to run.
freq_params	A vector of the frequency distribution parameters.
sev_params	A vector of the severity distribution parameters.
seedSetBinary	True if there is a fixed seed (seedValue), otherwise false. Defaults to TRUE when a seedValue is given and FALSE otherwise, so a seedValue on its own makes the run reproducible; an explicit FALSE ignores seedValue.
seedValue	The seed value, a whole number between -.Machine$integer.max and .Machine$integer.max, or NULL (the default) for no fixed seed.
freqDistr	The frequency distribution. Options are as per the freq_dist_options.
sevDistr	The severity distribution. Options are as per the sev_dist_options.
paretoSlice	True if there is Pareto slicing.
pareto_slice_times	The number of Pareto slices.
slice_pareto_alphas	A vector of Pareto slices' alpha parameters.
slice_pareto_x_ms	A vector of Pareto slices' x_m parameters.
sevCapBinary	True if there is a severity cap.
sev_cap_amount	The severity cap amount.
reinsuranceStructureEEL	The chosen reinsurance structure for each and every loss claim.
reinsurance_structure_eel_dedctible_amount	The deductible for each and every loss reinsurance structure.
reinsurance_structure_eel_limit_amount	The limit for each and every loss reinsurance structure.
reinsuranceStructureAL	The chosen reinsurance structure for aggregate claims.
reinsurance_structure_al_dedctible_amount	The deductible for aggregate reinsurance structure.
reinsurance_structure_al_limit_amount	The limit for aggregate reinsurance structure.
reinsuranceStructureLimitedReinstatements	True if there is a limit in reinstatements, otherwise false.
reinsuranceStructureReinstatementLimit	The reinstatement limit.

`multiprocessing`
: True to run the chunks in parallel with the future package, otherwise false. A future plan with more than one worker that the caller has already set is reused and left running. Otherwise the call starts a multisession plan with one worker per available core (`parallelly::availableCores()`), shuts those workers down when it finishes and restores the caller's plan, so every such call pays the start-up cost again. To choose the number of workers and reuse them across calls, set a plan first, e.g. `future::plan(future::multisession, workers = 4)`.

`sevTruncateAtZero`
: True to draw Normal severities from the Normal distribution truncated at zero, so that no claim is negative. Ignored for other severity distributions. Defaults to FALSE.

`chunk_size`
: The number of simulations processed per vectorised batch. By default (NULL) it is chosen from the expected number of claims per simulation, so that a batch holds about a million claims (between 100 and 10,000 simulations). Because of the floor of 100 simulations, a batch holds more than a million claims when the mean frequency exceeds 10,000 claims per simulation (about 100 million at a mean of a million), and memory use grows with it; give a smaller `chunk_size` to keep batches small. Results with a fixed seed depend on the chunk size.

`gross`
: True (the default) to return the gross total claims before reinsurance. Set it to FALSE when only the totals after the structures are needed: with an each-and-every-loss layer this allows drawing only the claims that reach the layer, which is much faster.

`shortcuts`
: True (the default) to use exact shortcuts where the settings allow: when no layer, cap, Pareto slice or truncation acts on individual claims, each simulation's total is drawn in one step for the Normal, Gamma, Exponential and fixed severities; with `gross = FALSE` and a layer, only the claims above the deductible are drawn. The results follow the same distribution as without shortcuts. Set it to FALSE to simulate every claim.

`progress`
: An optional function called after each chunk of a sequential run with the fraction done and a short description, e.g. to update a progress bar.

Details

Order of the calculations, for each simulation (a period, e.g. a year): claims are drawn from the severity distribution (with its Pareto slices) and capped at the severity cap; the each-and-every-loss (EEL) structure applies to each claim; the results are summed over the period; then the aggregate deductible comes off that sum, and finally the aggregate limit and the reinstatement capacity cap what is left. In short: cap -> EEL layer per claim -> annual sum -> aggregate deductible -> aggregate limit and reinstatement capacity.

The reinstatement capacity applies to a 'Limited Layer' EEL structure with limited reinstatements, which pays at most `(reinstatements + 1) * limit` in a period. With an aggregate 'Unlimited Layer' or 'Limited Layer', the ceded total is `min(max(S - aggregate deductible, 0), aggregate limit, (reinstatements + 1) * limit)`, where S is the sum of the period's EEL recoveries before any capacity (the market convention for an annual aggregate deductible). For example, three claims of 100 through a layer of 100 excess of 0 with no reinstatements and an aggregate deductible of 50 cede `min(300 - 50, 100) = 100`. Without an aggregate structure, the capacity caps S. With an

aggregate 'Exclude Layer', the capacity caps S first and the aggregate layer is then taken out of the capped amount. An unlimited EEL layer, or limited reinstatements switched off, has no capacity cap. (Before version 0.2.0 the capacity was applied before the aggregate deductible, which gave smaller ceded totals when the two were combined.)

Totals are returned at full precision; round them only for display.

Random numbers: each chunk of simulations uses its own L'Ecuyer-CMRG random stream, derived from one seed, so a run gives the same results whether or not it runs in parallel. With `seedSetBinary = TRUE` (the default when a `seedValue` is given) the run is reproducible from `seedValue` and the caller's random number stream is left unchanged; otherwise the seed is drawn from the caller's stream, so `set.seed()` before the call also makes it reproducible. The streams always use Inversion for normal draws and Rejection sampling, so a seed gives the same results whatever the caller's `RNGkind()`, which is restored afterwards. Results depend on the chunk size, which by default adapts to the expected number of claims per simulation.

Value

A data frame with one row per simulation, at full precision: `claim_counts`, the claim count; `total_claims`, the total claims after the reinsurance structures; `gross_claims`, the gross total claims before them (after Pareto slices and the severity cap; unless `gross = FALSE`); and, when reinstatements are limited, `number_of_reinstatements_used`: the EEL layer's recoveries in the period (after the aggregate deductible and limit, when there is an aggregate layer) divided by the EEL limit, capped at the number of reinstatements, so reinstatements are counted pro rata to the amount recovered. Stops with an error that names any required setting that is missing or invalid.

See Also

`simulate_claims`, a simpler interface with short argument names, and `run_shiny_simulator` for the same model in an app.

Examples

```
# 1,000 simulated years of Poisson claim counts with Normal claim sizes, no reinsurance
results <- simulate_function(
  numOfSimulations = 1000, freq_params = 3, sev_params = c(1000, 200),
  seedSetBinary = TRUE, seedValue = 1, freqDistr = "Poisson", sevDistr = "Normal"
)
summary(results$total_claims)

# the same claims ceded to a layer of 1,500 excess of 800 on each claim
layer <- simulate_function(
  numOfSimulations = 1000, freq_params = 3, sev_params = c(1000, 200),
  seedSetBinary = TRUE, seedValue = 1, freqDistr = "Poisson", sevDistr = "Normal",
  reinsuranceStructureEEL = "Limited Layer",
  reinsurance_structure_eel_dedctible_amount = 800,
  reinsurance_structure_eel_limit_amount = 1500
)
mean(layer$total_claims)
```

SlicedGammaParetoCappedMean *Sliced Gamma Pareto capped mean*

Description

Sliced Gamma Pareto capped mean

Usage

```
SlicedGammaParetoCappedMean(cap, GShape, GRate, SlicePoint, PShape)
```

Arguments

cap	A non-negative real number - the claim severity cap.
GShape	A positive real number - the shape parameter of the attritional Claim Severity's Gamma distribution.
GRate	A positive real number - the rate parameter of the attritional Claim Severity's Gamma distribution.
SlicePoint	A positive real number - the slice point and the scale parameter of the tail Claim Severity's Pareto distribution. An infinite slice point gives the Gamma distribution.
PShape	A positive real number - the shape parameter of the tail Claim Severity's Pareto distribution.

Value

The mean of the claim severity capped at cap with an attritional claim Gamma distribution with parameters GShape and GRate and a large claim Pareto distribution with parameters SlicePoint and PShape. A non-numeric argument, a negative cap or a non-positive parameter is an error; NA values give NA.

See Also

Other capped mean functions: GammaCappedMean(), LNormCappedMean(), ParetoCappedMean(), ParetoCappedMeanCalc(), SlicedLNormParetoCappedMean()

Examples

```
SlicedGammaParetoCappedMean(3000,1,0.0005,1000,1.2)
SlicedGammaParetoCappedMean(1000,1.1,0.0006,2000,1.6)
SlicedGammaParetoCappedMean(2000,1.2,0.0004,3000,1.4)
```

`SlicedGammaParetoMean` *Sliced Gamma Pareto mean*

Description

Sliced Gamma Pareto mean

Usage

```
SlicedGammaParetoMean(GShape, GRate, SlicePoint, PShape)
```

Arguments

`GShape`	A positive real number - the shape parameter of the attritional Claim Severity's Gamma distribution.
`GRate`	A positive real number - the rate parameter of the attritional Claim Severity's Gamma distribution.
`SlicePoint`	A positive real number - the slice point and the scale parameter of the tail Claim Severity's Pareto distribution. An infinite slice point gives the Gamma distribution.
`PShape`	A positive real number - the Shape parameter of the tail Claim Severity's Pareto distribution.

Value

The mean of the claim severity with an attritional claim Gamma distribution with parameters `GShape` and `GRate` and a large claim Pareto distribution with parameters `SlicePoint` and `PShape`. A non-numeric or non-positive parameter is an error; `NA` values give `NA`.

See Also

Other sliced distribution functions: `SlicedLNormParetoMean()`, `dSlicedGammaPareto()`, `dSlicedLNormPareto()`, `pSlicedGammaPareto()`, `pSlicedLNormPareto()`, `qSlicedGammaPareto()`, `qSlicedLNormPareto()`

Examples

```
SlicedGammaParetoMean(1,0.0005,1000,1.2)
SlicedGammaParetoMean(1.1,0.0006,2000,1.6)
SlicedGammaParetoMean(1.2,0.0004,3000,1.4)
```

`SlicedLNormParetoCappedMean`
Sliced LogNormal Pareto capped mean

Description

Sliced LogNormal Pareto capped mean

Usage

```
SlicedLNormParetoCappedMean(cap, mu, sigma, SlicePoint, shape)
```

Arguments

`cap`	A non-negative real number - the claim severity cap.
`mu`	A real number - the first parameter of the attritional Claim Severity's LogNormal distribution.
`sigma`	A positive real number - the second parameter of the attritional Claim Severity's LogNormal distribution.
`SlicePoint`	A positive real number - the slice point and the scale parameter of the tail Claim Severity's Pareto distribution. An infinite slice point gives the LogNormal distribution.
`shape`	A positive real number - the shape parameter of the tail Claim Severity's Pareto distribution.

Value

The mean of the claim severity capped at `cap` with an attritional claim LogNormal distribution with parameters `mu` and `sigma` and a large claim Pareto distribution with parameters `SlicePoint` and `shape`. A non-numeric argument, a negative `cap` or a non-positive `sigma`, `SlicePoint` or `shape` is an error; `NA` values give `NA`.

See Also

Other capped mean functions: `GammaCappedMean()`, `LNormCappedMean()`, `ParetoCappedMean()`, `ParetoCappedMeanCalc()`, `SlicedGammaParetoCappedMean()`

Examples

```
SlicedLNormParetoCappedMean(1200,6,1.5,1000,1.2)
SlicedLNormParetoCappedMean(2500,6.5,1.4,2000,1.6)
SlicedLNormParetoCappedMean(4000,7,1.6,3000,1.4)
```

SlicedLNormParetoMean *Sliced LogNormal Pareto mean*

Description

Sliced LogNormal Pareto mean

Usage

```
SlicedLNormParetoMean(mu, sigma, SlicePoint, shape)
```

Arguments

mu	A real number - the first parameter of the attritional Claim Severity's LogNormal distribution.
sigma	A positive real number - the second parameter of the attritional Claim Severity's LogNormal distribution.
SlicePoint	A positive real number - the slice point and the scale parameter of the tail Claim Severity's Pareto distribution. An infinite slice point gives the LogNormal distribution.
shape	A positive real number - the shape parameter of the tail Claim Severity's Pareto distribution.

Value

The mean of the claim severity with an attritional claim LogNormal distribution with parameters mu and sigma and a large claim Pareto distribution with parameters SlicePoint and shape. A non-numeric argument or a non-positive sigma, SlicePoint or shape is an error; NA values give NA.

See Also

Other sliced distribution functions: SlicedGammaParetoMean(), dSlicedGammaPareto(), dSlicedLNormPareto(), pSlicedGammaPareto(), pSlicedLNormPareto(), qSlicedGammaPareto(), qSlicedLNormPareto()

Examples

```
SlicedLNormParetoMean(6,1.5,1000,1.2)
SlicedLNormParetoMean(6.5,1.4,2000,1.6)
SlicedLNormParetoMean(7,1.6,3000,1.4)
```

Index

∗ **ILF functions**
ILFGamma, 12
ILFLNorm, 13
ILFPareto, 14
ILFSlicedGammaPareto, 15
ILFSlicedLNormPareto, 16
∗ **capped mean functions**
GammaCappedMean, 10
LNormCappedMean, 17
ParetoCappedMean, 17
ParetoCappedMeanCalc, 18
SlicedGammaParetoCappedMean, 34
SlicedLNormParetoCappedMean, 36
∗ **exposure curve functions**
ExposureCurveGamma, 6
ExposureCurveLNorm, 7
ExposureCurvePareto, 8
ExposureCurveSlicedGammaPareto, 8
ExposureCurveSlicedLNormPareto, 9
∗ **pure IBNR functions**
PureIBNRGamma, 21
PureIBNRLNorm, 22
∗ **sliced distribution functions**
dSlicedGammaPareto, 4
dSlicedLNormPareto, 5
pSlicedGammaPareto, 19
pSlicedLNormPareto, 20
qSlicedGammaPareto, 23
qSlicedLNormPareto, 24
SlicedGammaParetoMean, 35
SlicedLNormParetoMean, 37

apply_deductible_limit, 3

dSlicedGammaPareto, 4
dSlicedGammaPareto(), *5*, *20*, *24, 25*, *35*, *37*
dSlicedLNormPareto, 5
dSlicedLNormPareto(), *4*, *20*, *24, 25*, *35*, *37*

erf, 6

ExposureCurveGamma, 6
ExposureCurveGamma(), *7–10*
ExposureCurveLNorm, 7
ExposureCurveLNorm(), *7–10*
ExposureCurvePareto, 8
ExposureCurvePareto(), *7*, *9, 10*
ExposureCurveSlicedGammaPareto, 8
ExposureCurveSlicedGammaPareto(), *7, 8*, *10*
ExposureCurveSlicedLNormPareto, 9
ExposureCurveSlicedLNormPareto(), *7–9*

GammaCappedMean, 10, *11*
GammaCappedMean(), *17–19*, *34*, *36*

IGamma, 11
ILFGamma, 12
ILFGamma(), *13–16*
ILFLNorm, 13
ILFLNorm(), *12*, *14–16*
ILFPareto, 14
ILFPareto(), *12, 13*, *15, 16*
ILFSlicedGammaPareto, 15
ILFSlicedGammaPareto(), *12–14*, *16*
ILFSlicedLNormPareto, 16
ILFSlicedLNormPareto(), *12–15*

LNormCappedMean, *6*, 17
LNormCappedMean(), *11*, *18, 19*, *34*, *36*

ParetoCappedMean, 17, *19*
ParetoCappedMean(), *11*, *17*, *19*, *34*, *36*
ParetoCappedMeanCalc, 18
ParetoCappedMeanCalc(), *11*, *17, 18*, *34*, *36*
pSlicedGammaPareto, 19
pSlicedGammaPareto(), *4, 5*, *20*, *24, 25*, *35*, *37*
pSlicedLNormPareto, 20
pSlicedLNormPareto(), *4, 5*, *20*, *24, 25*, *35*, *37*

PureIBNRGamma, 21
PureIBNRGamma(), *23*
PureIBNRLNorm, 22
PureIBNRLNorm(), *21*

qSlicedGammaPareto, 23
qSlicedGammaPareto(), *4, 5*, *20*, *25*, *35*, *37*
qSlicedLNormPareto, 24
qSlicedLNormPareto(), *4, 5*, *20*, *24*, *35*, *37*

run_shiny_distribution_fitting_tool, 25
run_shiny_glm_fitting_tool, 26
run_shiny_simulator, 26, *29*, *33*

simulate_claims, *26*, 27, *33*
simulate_function, *26–29*, 30
SlicedGammaParetoCappedMean, 34
SlicedGammaParetoCappedMean(), *11*, *17–19*, *36*
SlicedGammaParetoMean, 35
SlicedGammaParetoMean(), *4, 5*, *20*, *24, 25*, *37*
SlicedLNormParetoCappedMean, 36
SlicedLNormParetoCappedMean(), *11*, *17–19*, *34*
SlicedLNormParetoMean, 37
SlicedLNormParetoMean(), *4, 5*, *20*, *24, 25*, *35*