Package 'sdprisk'

July 23, 2025

Type Package

Version 1.1-6

Date 2019-05-01

Title Measures of Risk for the Compound Poisson Risk Process with Diffusion

Maintainer Benjamin Baumgartner <benjamin@baumgrt.com>

Description Based on the compound Poisson risk process that is perturbed by a Brownian motion, saddlepoint approximations to some measures of risk are provided. Various approximation methods for the probability of ruin are also included. Furthermore, exact values of both the risk measures as well as the probability of ruin are available if the individual claims follow a hypo-exponential distribution (i. e., if it can be represented as a sum of independent exponentially distributed random variables with different rate parameters). For more details see Gatto and Baumgartner (2014) <doi:10.1007/s11009-012-9316-5>.

License AGPL-3

Imports numDeriv, PolynomF (>= 2.0-0), rootSolve, utils, stats

NeedsCompilation yes

Author Benjamin Baumgartner [aut, cre],
Riccardo Gatto [ctb, ths],
Sebastian Szugat [ctb]

Repository CRAN

Date/Publication 2019-04-29 20:50:03 UTC

Contents

`adjcoef` *Adjustment Coefficient*

Description

Returns the adjustment coefficient of a risk process with Gaussian diffusion.

Usage

```
adjcoef(process)
```

Arguments

`process`	An object of the class `"riskproc"`.

Details

The moment-generating function of the individual claim amounts is required to be contained within the `"claiminfo"` element of `process` (see `claiminfo`). The only exception to this is the case of hypo-exponentially distributed claim amounts.

Value

The adjustment coefficient of the risk process represented by `process`. If the adjustment coefficient does not exist or cannot be computed for any reason, `NULL` is returned instead.

See Also

`riskproc` and `claiminfo` for more details on how to provide the information necessary to compute the adjustment coefficient.

Examples

```
## Setting up a risk process with hypo-exponentially distributed claims
myprocess <- riskproc(
    claims   = claiminfo(hypoexp = list(rates = c(1, 10))),
    premium  = 2,
    freq     = 1,
    variance = 0.4
)

## Return the adjustment coefficient
adjcoef(myprocess)
```

claiminfo *Distribution Information about Individual Claim Amounts*

Description

Creates or tests for claim information objects.

Usage

```
claiminfo(...)
is.claiminfo(x)
is.hypoexp(x)
```

Arguments

x	An R object
...	various objects determining the individual claim amount distribution. Refer to the details section.

Details

Typicall usages are:

```
claiminfo(mgf, mgf.d1, mgf.d2, pdf, cdf, mean)
```

To be completed.

Value

`claiminfo` returns an object of the class `"claiminfo"` (see details section).

`is.claiminfo` returns `TRUE` if `x` is a `"claiminfo"` object, and `FALSE` otherwise.

`is.hypoexp` returns `TRUE` if `x` is a `"claiminfo"` object describing hypo-exponentially distributed individual claim amounts (see dhypoexp), and `FALSE` otherwise. If `x` is a `riskproc` object, the function is applied to the `"claiminfo"` object contained within it.

Note

To be completed.

See Also

`riskproc`

Examples

```
## For hypo-exponentially distribution claim amounts
claiminfo(hypoexp = list(rates = c(1, 10)))

## A more complicated example
## Not run: claiminfo()
```

---

hypoexp *Hypo-Exponential Distribution*

Description

Density, distribution function, quantile function, random generation and moment-generating function (and its first two derivatives) for the hypo-exponential distribution with rates `rate`.

Usage

```
dhypoexp(x, rate = 1, log = FALSE)
phypoexp(q, rate = 1, lower.tail = TRUE, log.p = FALSE, tailarea = FALSE)
qhypoexp(p, rate, interval = c(0.0, 1.0e+10))
rhypoexp(n = 1, rate = 1)
mgfhypoexp(x, rate = 1, difforder = 0)
```

Arguments

`x`, `q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`difforder`	the order of derivative for the moment-generating function; currently only implemented for 0, 1, 2.
`rate`	vector of (unique) rates.
`lower.tail`	logical; if `TRUE`, probabilities are $\mathbf{P}(X \leq x)$, otherwise $\mathbf{P}(X > x)$.
`log`, `log.p`	logical; if `TRUE`, probabilities p are given as $\log(p)$.
`tailarea`	logical; if `TRUE`, probabilities are given for the integrated tail area distribution.
`interval`	Passed to `uniroot`.

Details

The sum of n independent exponentially distributed random variables X_i with rate parameters λ_i has a hypo-exponential distribution with rate vector $(\lambda_1, \ldots, \lambda_n)$.

The hypo-exponential distribution is a generalization of the Erlang distribution (a Gamma distribution with an integer-valued shape parameter) and a special case of the phase-type distribution (see References section).

The quantile function is computed by numeric inversion (using `uniroot`).

Value

dhypoexp gives the density, phypoexp gives the distribution function (or the integrated tail area distribution function), qhypoexp gives the quantile function, rhypoexp generates random deviates and mgfhypoexp gives the moment-generating function (or its derivative up to the second order).

Note

If length(rate) == 1, dhypoexp, phypoexp and rhypoexp are equivalent to dexp, pexp and rexp with rate parameter rate and should, in fact, be replaced by the latter ones for computation speed.

References

Neuts, M. F. (1981) *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach*, reprinted and corrected.

See Also

dexp, dgamma

Examples

```
## Random generation
rhypoexp(10, c(3, 5))

## Mean
mu <- mgfhypoexp(0, c(3, 5), difforder = 1)

## Variance
mgfhypoexp(0, c(3, 5), difforder = 2) - mu^2

## Quantile
qhypoexp(0.5, c(3, 5))
```

---

riskproc *Compound Poisson Risk Process with Diffusion*

Description

Creates an R object representing a compound Poisson risk process with Gaussian diffusion, which contains some or all information necessary for further processing.

Usage

```
riskproc(claims, premium, freq, variance)
is.riskproc(x)
```

Arguments

claims	a claiminfo object.
premium	premium rate.
freq	claim frequency.
variance	squared volatility of the Wiener component; currently only implemented for variance > 0.
x	an R object.

Details

Given the arguments, most prominently claims, various auxialiary parameters and functions associated with the risk process to be represented are calculated.

Value

riskproc returns an object of the class "riskproc". Internally, this is a list containing various elements (depending on the information provided in the arguments).

is.riskproc returns TRUE if x is a "riskproc" object, and FALSE otherwise.

See Also

claiminfo for more details about passing on information about the distribution of the individual claim amounts.

Examples

```
## A risk process with hypo-exponentially distributed individual claim amounts
riskproc(
    claims   = claiminfo(hypoexp = list(rates = c(1, 10))),
    premium  = 2,
    freq     = 1,
    variance = 0.4
)
```

---

ruinprob *Calculation or Approximation of the Probability of Ruin*

Description

This functions provide various approximation methods for the (total) probability of ruin, the probability of ruin due to oscillation and the probability of ruin due to a claim. Exact calculations are possible in the case of hypo-exponentially distrubuted claim amounts.

Usage

```
ruinprob(process, method = c("saddlepoint", "fft", "bounds", "hypoexp", "lundberg"), ...)
boundsRuinprob(process, interval, maxreserve, richardson = TRUE, use.splines = FALSE)
fftRuinprob(process, interval, maxreserve, n, use.splines = FALSE)
hypoexpRuinprob(process)
saddlepointRuinprob(process, jensen = FALSE, normalize = TRUE)
```

Arguments

process	a "riskproc" object.
method	character string indicating the method used for approximation or calculation.
interval	interval width for the discretization of the claim distribution.
maxreserve	maximal value of the initial reserve for which the approximation can be calculated.
n	Length of the probability vectors resulting from the discretization.
richardson	logical; if TRUE, Richardson extrapolation is used for the approximation of the probability of ruin due to oscillation.
use.splines	logical; if TRUE, a cubic spline interpolation is used instead of step functions.
jensen	logical; if TRUE, the formulae of *Jensen (1992)* are used instead of the ones by *Lugannani and Rice (1980)* and *Daniels (1954)* (see references).
normalize	logical; if TRUE, the saddlepoint approximations based on densities are re-normalized such that those densities integrate to 1.
...	further arguments that are passed on to boundsRuinprob, fftRuinprob, hypoexpRuinprob or saddlepointRuinprob, depending on the value of method.

Details

ruinprob is a wrapper function for the other ones given here.

Value

psi	the total probability of ruin (as a function of the initial reserve).
psi.1	the probability of ruin due to oscillation (as a function of the initial reserve).
psi.2	the probability of ruin due to a claim (as a function of the initial reserve).
...	

References

Daniels, H. E. (1954) Saddlepoint Approximations in Statistics. *Annals of Mathematical Statistics* **25**(4), pp. 631–650.

Gatto, R. and Mosimann, M. (2012) Four Approaches to Compute the Probability of Ruin in the Compound Poisson Risk Process with Diffusion. *Mathematical and Computer Modelling* **55**(3–4), pp. 1169–1185

Jensen, J. L. (1992) The Modified Signed Likelihood Statistic and Saddlepoint Approximations. *Biometrika* **79**(4), pp. 693–703.

Lugannani, R. and Rice, S. (1980) Saddle Point Approximation for the Distribution of the Sum of Independent Random Variables. *Advances in Applied Probability* **12**(2), pp. 475–490.

See Also

riskproc, claiminfo

`sensitivity` *Sensitivity of the Value and Tail Value at Ruin*

Description

The sensitivities of both the value and the tail value at ruin are defined as their respective derivatives with respect to the probability level.

Usage

```
sensitivity(process, method = c("saddlepoint", "hypoexp"), ...)
hypoexpSensitivity(process)
saddlepointSensitivity(process, ...)
```

Arguments

`process`	a riskproc object.
`method`	character string indicating the calculation or approximation method.
`...`	further arguments that are passed on to `saddlepointTvaru`.

Value

`varu`	a function returning the sensitivity of the value at ruin.
`tvaru`	a function returning the sensitivity of the tail value at ruin.

`tvaru` *Tail Value at Ruin*

Description

The tail value at ruin for a given probability level ε is defined as the conditional expectation of the maximal aggregate loss given that it is above the value at ruin of level ε.

Usage

```
tvaru(process, method = c("saddlepoint", "hypoexp"), ...)
hypoexpTvaru(process)
saddlepointTvaru(process, type = c("tail", "density"), ...)
```

Arguments

process	a "riskproc" object.
method	character string indicating the calculation or approximation method.
type	character string indicating which function is to be used for the approximation.
...	further arguments that are passed on to saddlepointRuinprob, depending on the value of method.

Details

tvaru is a wrapper function for hypoexpTvaru and saddlepointTvaru.

hypoexpTvaru calculates the tail value at ruin in the case of hypo-exponentially distributed claim amounts by numerical integration of the probability of ruin, which can be computed exactly.

saddlepointTvaru uses saddlepoint techniques for the approximation of the tail value at ruin. More precisely, the saddlepoint approximation to the probability of is numerically integrated in the frequency domain, and implicitly also the saddlepoint approximation to the value at ruin (see varu) is used. If type = "tail" the integrand is the probability of ruin (as function in the frequency domain), otherwise (type = "density") it is essentially a re-scaled version of the probability of ruin due to claims. The former requires fewer calculations and seems to produce slightly more accurate results.

Value

A function returning the tail value at ruin of a given probability level is returned.

If method = "saddlepoint" or if saddlepointTvaru is used, the returned function has an additional second argument giving the number of iterations used for the approximation of the value at ruin (i. e., the lower integration limit), see varu.

See Also

varu

---

varu *Value at Ruin*

Description

The value at ruin at a given probability level ε is defined as the minimal capital that is required in order to have a ruin probability of at most ε. This is equivalent to the $(1-\varepsilon)$-quantile of the maximal aggregate loss.

Usage

```
varu(process, method = c("saddlepoint", "hypoexp"), ...)
hypoexpVaru(process)
saddlepointVaru(process, type = 2)
```

Arguments

process	a "riskproc" object.
method	character string indicating the calculation or approximation method.
type	number indicating the type of approximation; possible choices are 1 and 2.
...	further arguments that are passed on to saddlepointVaru, depending on the value of method.

Details

varu is a wrapper function for hypoexpVaru and saddlepointVaru.

hypoexpVaru calculates the value at ruin in the case of hypo-exponentially distributed claim amounts by numerical inversion of the probability of ruin, which can be computed exactly.

saddlepointVaru uses saddlepoint techniques for the approximation of the value at ruin, more specifically, the inversion algorithms provided by *Wang (1995)*. The first one (type = 1) is only given for completeness (or comparison purposes), because, due to repeatedly switching back and forth between the monetary domain the frequency (saddlepoint) domain, it is much slower than the second one (type = 2), which is performed entirely in the frequency domain. Refer to the references given below for more details.

Value

A function returning the value at ruin of a given probability level is returned.

If method = "saddlepoint" or if saddlepointVaru is used, the returned function has an additional second argument giving the number of iterations.

References

Wang, Suojin (1995) One-Step Saddlepoint Approximations for Quantiles. *Computational Statistics and Data Analysis* **20**(1), pp. 65–74.

Index