Package ‘bootruin’

July 22, 2025

Type Package

Title A Bootstrap Test for the Probability of Ruin in the Classical Risk Process

Version 1.2-4

Date 2016-12-30

Author
Benjamin Baumgartner <benjamin@baumgrt.com>, Riccardo Gatto <gatto@stat.unibe.ch>

Maintainer Benjamin Baumgartner <benjamin@baumgrt.com>

Description We provide a framework for testing the probability of ruin in the classical (compound Poisson) risk process. It also includes some procedures for assessing and comparing the performance between the bootstrap test and the test using asymptotic normality.

License AGPL-3

Imports stats, utils

NeedsCompilation yes

Repository CRAN

Date/Publication 2016-12-30 19:39:15

Contents

`bootruin-package`	*A Bootstrap Test for the Probability of Ruin in the Compound Poisson Risk process*

Description

This package provides a bootstrap test for the probability of ruin in the classical (compound Poisson) risk process and some procedures for comparing the performance of the bootstrap test and the test using the asymptotic normal approximation.

Details

See the reference for more information.

Author(s)

Benjamin Baumgartner <`benjamin@baumgrt.com`>,
Riccardo Gatto <`gatto@stat.unibe.ch`>

References

Baumgartner, B. and Gatto, R. (2010) *A Bootstrap Test for the Probability of Ruin in the Compound Poisson Risk Process*. ASTIN Bulletin, **40**(1), pp. 241–255.

`pvaldens`	*Density Estimation of Data in the Unit Interval*

Description

This function computes density estimators for densities with the unit interval as support. One example of data with such a density are p-values. Currently, two methods are implemented that differ in the kernel function used for estimation.

Usage

```
pvaldens(x, bw, rho, method = c("jh", "chen"))
```

Arguments

`x`	a numeric vector of data points between 0 and 1.
`bw`	a number indicating the bandwidth used for the density estimation.
`rho`	a number determining the correlation coefficient, only used if `method = "jh"`
`method`	a character string determining the kernel function that is used, see Details.

Details

Depending on which `method` is selected, a different kernel function is used for the estimation. Since the support of the estimated function is bounded, those kernel functions are location-dependent.

If `method = "jh"`, a Gaussian copula-based kernel function according to Jones and Henderson (2007) is used. In this case the bandwidth can either be specified directly or as correlation coefficient: if $\rho > 0$ denotes the correlation coefficient and $h > 0$ the bandwidth, then $h^2 = 1 - \rho$. Note that `rho` and `bw` are mutually exclusive.

For `method = "chen"`, the kernel function is based on a beta density, according to Chen (1999).

See the cited articles for more details.

Value

A function with a single vector-valued argument that returns the estimated density at any given point(s).

References

Jones, M. C. and Henderson, D. A. (2007) *Kernel-Type Density Estimation on the Unit Interval*. Biometrika, **94**(4), pp. 977–984.

Chen, S. X. (1999) *A Beta Kernel Estimation for Density Functions*. Computational Statistics and Data Analysis, **31**(2), pp. 131–145.

See Also

`density`

Examples

```
require(graphics)

x <- rbeta(100, 2, 5)
fhat <- pvaldens(x, rho = 0.9, method = "jh")

hist(x, freq = FALSE, xlim = c(0, 1))
curve(fhat(x), from = 0, to = 1, add = TRUE, col = 2)
box()
```

`pvaldistance` *Distance Measures of Empirical Probability Functions*

Description

This function provides a framework to evaluate various measures of distance between an empirical distribution (induced by the dataset provided) and a theoretical probability distribution.

Usage

```
pvaldistance(x, method = c("ks", "cvm"), dist.to = c("uniform"))
```

Arguments

`x`	a numeric vector containing a data sample.
`method`	a character string indicating which measure of distance is computed.
`dist.to`	a character string determining the (theoretical) probability distribution that is used as a reference.

Details

`method = "ks"` gives the Kolmogorov-Smirnov distance.

`method = "cvm"` yields the Cramér-von-Mises criterion (scaled with the sample size).

Value

A positive real number giving the distance measure.

Note

At the moment, `dist.to = "uniform"` (the uniform distribution on the unit interval) is the only valid option for the theoretical distribution, and hence the members of `x` have to lie in the unit interval.

See Also

See `ks.test` for the Kolmogorov-Smirnov test.

Examples

```
# A sample from the standard uniform distribution
x <- runif(100, 0, 1)

# Distance to uniformity should be small
pvaldistance(x, "ks")
pvaldistance(x, "cvm")

# A sample from the Beta(2, 7) distribution
y <- rbeta(100, 2, 7)

# Distance to uniformity should be much larger here
pvaldistance(y, "ks")
pvaldistance(y, "cvm")
```

ruinprob — *The Probability of Ruin in the Classical Risk Process*

Description

This function calculates or estimates the probability of ruin in the classical (compund Poisson) risk process using several different methods.

Usage

```
ruinprob(x, param.list, compmethod = c("dg", "exp"),
    flmethod = c("nonp", "exp", "lnorm", "custom"),
    reserve, loading, fl = NA, interval = 0.5,
    implementation = c("R", "C"), ...)
```

Arguments

`x`	a numeric vector, matrix or array of individual claims.
`param.list`	a named list of parameters. It might contain any of the arguments except `x` and `...`
`compmethod`	a character string determining the algorithm for the computation.
`flmethod`	a character string indicating what cumulative probability distribution function is used for the increments of the running maximum of the aggregate loss process if `compmethod = "dg"`, see also Details and References.
`reserve`	a number indicating the initial surplus.
`loading`	a number determining the relative security loading.
`fl`	a function that is used as custom cumulative probability distribution to be used for the discretization if `flmethod = "custom"`.
`interval`	a number determining the approximation precision, viz. the mesh width of the discretization if `compmethod = "dg"`, see Details and References.
`implementation`	a character string determining whether to use the native implementation in R or the one in C.
`...`	further arguments are passed to `fl`.

Details

The classical risk process, also called Cramér-Lundberg risk process, is a stochastic model for an insurer's surplus over time and, for any $t \geq 0$, it is given by

$$Y_t = r_0 + ct - Z_t,$$

where Z_t is a compund Poisson process, $r_0 \geq 0$ is the initial surplus and $c > 0$ is the constant premium rate.

This function calculates, approximates or estimates (depending on what options are given) the probability of ruin in the infinite time horizon, i.e. the probability that Y_t ever falls below 0.

Currently there are two options for the `compmethod` argument. If `compmethod = "exp"`, the claims are assumed to be from an exponential distribution. In that case, the probability of ruin is given by

$$\frac{1}{1+\beta}\exp\left\{-\frac{\beta}{1+\beta}\frac{r_0}{\mu}\right\},$$

where μ is the mean claim size (estimated from `x`) and β is the relative security loading.

For `compmethod = "dg"`, the recursive algorithm due to Dufresne and Gerber (1989) is used. In this case, the parameter `flmethod` determines what cumulative distribution function is used for the discretization. The possible choices are either a non-parametric estimator, parametric estimators for exponential or log-normal claims, or a user-supplied function (in which the argument `fl` must be specified). See the reference for more details on how this algorithm works.

Value

The estimated or calculated probability of ruin. The shape and dimension of the output depends on the specifics of the claim data `x`. If `x` is a vector, the output is a single numeric value. In general, the dimension of the output is one less than that of `x`. More precisely, if `x` is an array, then the output value is an array of dimension `dim(x)[-1]`, see the note below.

Note

If `x` is an array rather than a vector, the function acts as if it was called through `apply` with `MARGIN = 2:length(dim(x))`

If an option is given both explicitly and as part of the `param.list` argument, then the value given explicitly takes precedence. This way the parameter list, saved as a variable, can be reused, but modifications of one or more parameter values are still possible.

References

Dufresne, F. and Gerber, H.-U. (1989) *Three Methods to Calculate the Probability of Ruin*. ASTIN Bulletin, **19**(1), pp. 71–90.

See Also

`ruinprob.test`

Examples

```
# Claims have an exponential distribution with mean 10
x <- rexp(10, 0.1)
print(x)

# The estimated probability of ruin
ruinprob(x, reserve = 100, loading = 0.2, interval = 0.25)

# The true probability of ruin of the risk process
ruinprob(
    10, reserve = 100, loading = 0.2,
    flmethod = "exp", compmethod = "exp"
)
```

`ruinprob.test`	*A Bootstrap Test for the Probability of Ruin in the Classical Risk Process*

Description

This function provides a testing framework for the probability of ruin in the classical, compound Poisson risk process. The test can be performed using the bootstrap method or using normal approximation.

Usage

```
ruinprob.test(x, prob.null, type = c("bootstrap", "normal"),
    nboot, bootmethod = c("nonp", "exp", "lnorm"), ...)
```

Arguments

`x`	a numeric vector of data values (claims)
`prob.null`	a number indicating the hypothesized true probability of ruin.
`type`	a character string determining the type of test that is performed.
`nboot`	a number indicating the number of bootstrap replications.
`bootmethod`	a character string determining how the bootstrap replications are created.
`...`	further arguments to be passed to `ruinprob`.

Details

The null hypothesis is that the probability of ruin is equal to `prob.null` versus the one-sided alternative that probability of ruin is smaller than `prob.null`.

If `type = "bootstrap"`, a bootstrap test is performed. The arguments `nboot` and `bootmethod` have to be specified. `bootmethod` determines the kind of bootstrap: `"nonp"` creates the usual nonparametric bootstrap replications, while `"exp"` and `"lnorm"` create parametric bootstrap replications, the former assuming exponentially distributed claims, the latter log-normally distributed ones.

`type = "normal"` makes use of an asymptotic normal approximation. The computations are a lot faster, but from a theoretical point of view the bootstrap method is more accurate, see References.

For details about the necessary and valid arguments that might have to be supplied for `...`, see `ruinprob`.

Value

A list with class `"htest"` containing the following components:

`statistic`	the value of the studentized probability of ruin, i.e. the test statistic.
`parameter`	additional parameters.
`p.value`	the p-value for the test.

`estimate`	the estimated probability of ruin.
`null.value`	the specified hypothesized value of the probability of ruin.
`alternative`	a character string describing the alternative hypothesis.
`method`	a character string indicating what type of test was performed.
`data.name`	a character string giving the name of the data.

Note

Using the bootstrap method is computationally intensive. Values for `nboot` should not be too large, usually numbers between 50 and 200 are reasonable choices.

References

Baumgartner, B. and Gatto, R. (2010) *A Bootstrap Test for the Probability of Ruin in the Compound Poisson Risk Process*. ASTIN Bulletin, **40**(1), pp. 241–255.

See Also

`ruinprob`

Examples

```
# Generating a sample of 50 exponentially distributed claims with mean 10
x <- rexp(50, 0.1)

## Not run:
# Given this sample, test whether the probability of ruin is smaller than
# 0.1 using a bootstrap test with 100 bootstrap replications.
ruinprob.test(
    x = x, prob.null = 0.10, type = "bootstrap",
    loading = 0.2, reserve = 100, interval = 1,
    bootmethod = "nonp", nboot = 100
)

## End(Not run)

# The same test using normal approximation. This is a lot faster.
ruinprob.test(
    x = x, prob.null = 0.15, type = "normal",
    loading = 0.2, reserve = 100, interval = 1
)
```

Index