Package 'cyra4cm'

March 29, 2024

Type Package

Title Cyber Risk Assessment for Capital Management

Version 0.1.0

Author anonymous

Maintainer anonymous <anonymous@email.com>

Description na

Depends R (>= 4.1.0), mathjaxr

License GPL (>= 2)

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.2

RdMacros mathjaxr

R topics documented:

calcCapitalAllocation 2
calcConditionalExpection 3
calcNormalizer.l1norm 3
calcPenaltyValue 4
calcVaR 4
createInvestTable 5
createWeightsTables 6
cyraSampleData 7
generateDiscretizedProbs 7
generateLossDists 8
generateSyntheticData 9
prepareCyraParameters 10
scaleLogNormal 10

Index **12**

`calcCapitalAllocation` *Develop Capital Allocation Strategy for Cyber Risk Management*

Description

This function outputs capital allocation strategies with discrete investment options according to the holistic principle. This function should only be used when all components of the allocation model are ready.

Usage

```
calcCapitalAllocation(
  invest.table,
  cyra.weights,
  normalizer = "cte",
  cyra.params,
  scale.fun = scaleLogNormal,
  penalty.thres = 0.9,
  total_budget = NA,
  ...
)
```

Arguments

`invest.table`	a data.frame of class `cyraInvestTable` generated by function `createInvestTable()`.
`cyra.weights`	a list of class `cyraWeights` generated by function `createWeightsTables()`.
`normalizer`	either a character value that specifies how the expected deviance should be normalized, or a custom function that returns a vector of normalizing factors.
`cyra.params`	a list of class `cyraParams` generated by function `prepareCyraParameters()`.
`scale.fun`	a function that scales a random variable. `scaleLogNormal()` is used by default, and should be changed accordingly depending on the distribution fitted to losses.
`penalty.thres`	a numeric value between 0 and 1, which is the probability used in the TVaR penalty function. 0.9 is used by default.
`...`	additional arguments for `normalizer` if it is a custom function.

Value

a data.frame that shows all cybersecurity investment options and all corresponding optimal reserving schemes.

- `investj` for `j`$= 1, 2, \ldots, m$: amount spent on control/vulnerability `j`.
- `thetaj` for `j`$= 1, 2, \ldots, m$: theta value after the cybersecurity investment is made on control/vulnerability `j`.
- `Kik` for `i`$= 1, 2, \ldots, l$ and `k`$= 1, 2, \ldots, n$: reserve on threat-asset pair `ik`.
- `obj_tot`: objective function value of the corresponding investment and reserving strategy.
- `obj_c`: component in the objective function that is related to the investment part.
- `obj_r`: component in the objective function that is related to the reserving part.

calcConditionalExpection

Calculate Conditional Expectation

Description

This function calculates the conditional expectation of X, when the Value-at-Risk of X is given

Usage

```
calcConditionalExpection(x, p, q)
```

Arguments

x	a numeric value representing the Value-at-Risk of random variable X
p	a numeric vector of probabilities of the distribution of X
q	a numeric vector of quantiles corresponding to p

calcNormalizer.l1norm *Built-in Weight Normalizers*

Description

These functions are possible choices of the normalizer of each expected deviance term.

Usage

```
calcNormalizer.l1norm(term.pt = "TVaR", h.func, p, q, ...)

calcNormalizer.local(h.func, p, q, ...)

calcNormalizer.cte(p, q, ...)
```

Arguments

term.pt	either a numerical value or the character value "TVaR" (default). This is the terminal point such that the l1-norm of the quadratic function and that of the linear function are equal with in [0, term.pt]. If term.pt = "TVaR", then the value of the terminal point is the TVaR of the loss. This argument is needed only for function calcNormalizer.l1norm.
h.func	a weighted penalty function. See calcPenaltyValue.CTE().
p	a numeric vector representing the probabilities of the loss random variable in the expected deviance.
q	a numeric vector that has the same length as p and represent the quantiles corresponding to p.
...	additional needed arguments in h.func.

Value

a numeric value of the calculated normalizer.

Functions

- `calcNormalizer.l1norm`: This function calculates the normalizer that equates the l1-norm of the expected deviance to that of the linear reserve term, up to a terminal point.
- `calcNormalizer.local`: This function calculates the normalizer that equates the expected deviance to the linear reserve term at the point where the expected deviance is minimized.
- `calcNormalizer.cte`: This function calculates the normalizer that equates the average marginal decrease in expected deviance to the (marginal) increase in linear reserve term within the interval from no reserve to the amount that minimizes the expected deviance. In that case, the normalizer takes the value of the CTE of the loss random varible.

`calcPenaltyValue` *Weighted Penalty Function - CTE*

Description

This is a the weighted penalty function defined as follows, $\frac{1_{X>x}}{\mathbb{P}(X>x)}$.

Usage

```
calcPenaltyValue(x, p, q)
```

Arguments

`x`	a numeric value representing the parameter x in the weighted penalty function.
`p`	a numeric vector that represents the probabilities of the loss distribution.
`q`	a numeric vector that has the same length as `p` and represents the quantiles corresponding to `p`.

Value

a numeric value representing the penalty

`calcVaR` *Calculate Value-at-Risk of a Discretized Distribution*

Description

This function calculates the Value-at-Risk a discretized version of X

Usage

```
calcVaR(alpha, p, q)
```

Arguments

alpha	a numeric value representing the probability corresponding to the VaR to be found
p	a numeric vector of probabilities of the distribution of X
q	a numeric vector of quantiles corresponding to p

---

createInvestTable *Create Cyber Control Investment Table with Discrete Investment Options*

Description

Assume that there are discrete investment options on individual cybersecurity vulnerabilities, these functions help populates all possible combinations of investment decisions on individual vulnerabilities.

Usage

```
createInvestTable(inv.opt)

cyraInvestOptions(...)

cyraInvestOption(idx, invest, theta)

extractFromInvestTable(invest.table, field)
```

Arguments

inv.opt	a list of class cyraInvestOptions generated by function cyraInvestOptions().
...	objects of class cyraInvestOption
idx	an integer indicating the index of the vulnerability that this investment option is for.
invest	a numeric value representing the amount spent on this vulerability
theta	a numeric value between 0 and 1, indicating the remaining vulnerability after the investment is made.
invest.table	a cyraInvestTable object.
field	a character value, of which the value is either invest or theta

Functions

- cyraInvestOptions: this function creates a list of class cyraInvestOptions, which is used in function createInvestTable().
- cyraInvestOption: this function creates an individual cyraInvestOption item.
- extractFromInvestTable: this is a helper function that extracts invest or theta information from a cyraInvestTable object (see createInvestTable()).

`createWeightsTables` *Prepare Weights for Capital Allocation*

Description

This function can be used to format the user-provided weights into data.frames that are compatible with the risk assessment and capital allocation framework.

Usage

```
createWeightsTables(cyra.params, nu.ik, om.ik, eta.j, nu, om, eta)

createNuOmegaikTable(l, n, nu.ik, om.ik)

createEtaTable(m, eta.j, eta)

createNuOmegaTable(nu, om)
```

Arguments

`nu.ik, om.ik`	numeric vectors specifying standalone reserve weights, nu_ik and omega_ik.
`eta.j`	a numeric vector specifying standalone cybersecurity investment weights.
`nu, om`	numeric values specifying aggregate reserve weights, nu and omega.
`eta`	a numeric value specifying aggregate cybersecurity investment weights.
`l, m, n`	integer values. l: number of threats; m: number of vulnerabilities; n: number of assets.

Value

a list of weights tables

- `nu_om_ik`: weights on reserves (ν_{ik}) and reserve-loss mismatches (ω_{ik}) on individual threat-asset pairs.
- `nu_om`: weights on aggregate reserve (ν) and aggregate reserve-loss mismatch (ω)
- `etas`: weights on cybersecurity investments, including standalone investments on individual controls and the aggregate investment.

Functions

- `createNuOmegaikTable`: create a table for standalone reserve weights, nu_ik and omega_ik
- `createEtaTable`: create a compatible table for investment weights, eta
- `createNuOmegaTable`: create a compatible table for aggregate reserve weights, nu and omega

cyraSampleData *cyra Sample Data for Demonstration and Testing*

Description

This is a dataset generated by function `generateSyntheticData()`

Usage

```
data(cyraSampleData)
```

Format

An object of class `data.frame` with 1778 rows and 6 columns.

generateDiscretizedProbs *Discretize Continuous Loss Distributions*

Description

This function discretizes a continuous distribution by allocating the probability within each interval to the beginning of that interval

Usage

```
generateDiscretizedProbs(q, pdist, ...)
```

Arguments

`q`	a numeric vector of quantiles, for which the discretized probabilites are calculated
`pdist`	a character value, indicating the name of the continuous distribution, e.g., "lnorm", "weibull", etc.
`...`	a named numeric vector of parameters of the distribution.

Value

a numeric vector of probabilities that corresponds to the input quantiles `q`.

`generateLossDists` *Aggregate Loss Distribution Functions*

Description

The function generates discretized probability distributions of various losses arising from a cyber system.

Usage

```
generateLossDists(th, cyra.params, scale.fun = scaleLogNormal)
```

Arguments

`th`	a numeric vector, of which the length equals the number of vulnerabilities. Elements in the vector range from 0 to 1. `th` (Vector θ) represents the scaling of loss random variables when cybersecurity controls are present. If losses are unmitigated, then `th` should be a vector of one's.
`cyra.params`	a list of S3 class cyraParams, generated by function `prepareCyraParameters()`.
`scale.fun`	a function that returns the new parameters of a probability distribution of a random variable after it is scaled. For more details, see built-in functions `scaleLogNormal()` (default) and `scaleWeibull()`. A compatible customer function can also be provided.

Value

This function returns a list of discretized probability distributions that are represented by quantile-probability tables. Items in the list are as follows,

- X_dist: the distribution of X_{ijk}, the loss random variable on each attack path defined by a threat-vulnerability-asset tuple.
- Zik_dist: the distribution of Z_{ik}, the loss random variable associated with each threat-asset pair.
- Zi_dist: the distribution of Z_i, the loss random variable associated with each threat.
- L_dist: the distribution of L, the random variable of the total loss in one incident.
- S_dist: the distribution of S, the random variable of the aggregate loss over a period.
- Sik_dist: the distribution of S_{ik}, the random variable associated with the aggregate loss on each threat-asset pair over a period.

generateSyntheticData *Generate Synthetic Industry Cyber Incident Data*

Description

This function generates synthetic industry cyber incident data by simulation. The generated dataset is compatible with the risk assessment and capital allocation framework proposed by Chong, et al. and is for demonstration and testing purposes only.

Usage

```
generateSyntheticData(attack.paths, lambda, years, prob.com, seed = 199)
```

Arguments

attack.paths	a list of attack path information, see Details.
lambda	a number, which is parameter of the Poisson distribution used for generating yearly number of incidents in this industry.
years	a numeric vector that represents the time span of the sample.
prob.comp	a number, which represents probability of the incident occurring to the company of interest.
seed	a number, seed of random number generator.

Details

To be added

Value

This function returns a cyraIncidents class data.frame of synthetic cyber incidents. Its structure is compatible with the risks assessment and capital allocation framework in this package.

Examples

```
# create attack paths information
path_info <- list(
list("T" = 1, "V" = 1,"A" = 1, "Xzero" = 0.90, "Xdist" = "lnorm", "Xparams" = c("meanlog" = 10, "sdlog" = 5), 'pro
list("T" = 2, "V" = 2,"A" = 2, "Xzero" = 0.92, "Xdist" = "lnorm", "Xparams" = c("meanlog" = 2, "sdlog" = 10), 'pro
list("T" = 2, "V" = 3,"A" = 2, "Xzero" = 0.85, "Xdist" = "lnorm", "Xparams" = c("meanlog" = 5, "sdlog" = 1), 'prob
)
lambda <- 100
years <- 2001:2008
prob_comp <- 0.1
sample_data <- generateSyntheticData(path_info = path_info, lambda = lambda, years = years, prob_comp = prob_co
```

prepareCyraParameters *Estimate Parameters for Risk Assessment and Capital Allocation*

Description

Based on the dataset of industry cyber incidents, this functions estimates all parameters that are needed for cyber risk assessment and capital allocation.

Usage

```
prepareCyraParameters(
  ind.data,
  loss.dist.name = "lnorm",
  grid.ub.prob = 0.95,
  grid.num = 1000L
)
```

Arguments

ind.data	A cyraIncidents class data.frame.
loss.dist.name	A character value, specifies the name of the distribution that will be fitted to the loss data. Common choices are "lnorm" (by defauly, lognormal distribution) and "weibull" (Weibull distribution).
grid.ub.prob	A numeric value, specifies the probability corresponding to the largest quantile that will be used in the process of discretizing loss distributions.
grid.num	An integer value, specifies the number of grids when discretizing loss distributions.

Value

This function returns a list of parameters with their values, which will be used for risk assessment and capital allocation.

Examples

```
risk_params <- generateRiskParameters(ind.data = sample_data)
```

scaleLogNormal *Scale a Random Variable*

Description

Here are two built-in functions for scaling lognormal-distributed and Weibull-distributed random variables.

Usage

```
scaleLogNormal(th, meanlog, sdlog)

scaleWeibull(th, shape, scale)
```

Arguments

`th`	a numeric value representing the scaling factor.
`meanlog, sdlog`	numeric values, which are parameters of a lognormal distribution before scaling.
`shape, scale`	numeric values, which are parameters of a Weibull distribution before scaling.

Value

a name vector of new parameters.

Functions

- `scaleLogNormal`: scales a lognormal-distributed random variable, i.e., $Y = \theta X$, where X follows a lognormal distribution with parameters `meanlog` and `sdlog`. The function outputs the parameters of Y.
- `scaleWeibull`: scales a Weibull-distributed random variable, i.e., $Y = \theta X$, where X follows a Weibull distribution with parameters `scale` and `shape`. The function outputs the parameters of Y.

Index

∗ **datasets**
cyraSampleData, 7

calcCapitalAllocation, 2
calcConditionalExpection, 3
calcNormalizer.cte *(*calcNormalizer.l1norm*)*, 3
calcNormalizer.l1norm, 3
calcNormalizer.local *(*calcNormalizer.l1norm*)*, 3
calcPenaltyValue, 4
calcPenaltyValue.CTE(), *3*
calcVaR, 4
createEtaTable *(*createWeightsTables*)*, 6
createInvestTable, 5
createInvestTable(), *2*, *5*
createNuOmegaikTable *(*createWeightsTables*)*, 6
createNuOmegaTable *(*createWeightsTables*)*, 6
createWeightsTables, 6
createWeightsTables(), *2*
cyraInvestOption *(*createInvestTable*)*, 5
cyraInvestOptions *(*createInvestTable*)*, 5
cyraInvestOptions(), *5*
cyraSampleData, 7

extractFromInvestTable *(*createInvestTable*)*, 5

generateDiscretizedProbs, 7
generateLossDists, 8
generateSyntheticData, 9
generateSyntheticData(), *7*

prepareCyraParameters, 10
prepareCyraParameters(), *2*, *8*

scaleLogNormal, 10
scaleLogNormal(), *2*, *8*
scaleWeibull *(*scaleLogNormal*)*, 10
scaleWeibull(), *8*