Package ‘cascsim’

January 13, 2020

Title Casualty Actuarial Society Individual Claim Simulator

Version 0.4

Description It is an open source insurance claim simulation engine sponsored by the Casualty Actuarial Society. It generates individual insurance claims including open claims, reopened claims, incurred but not reported claims and future claims. It also includes claim data fitting functions to help set simulation assumptions. It is useful for claim level reserving analysis.


Depends R (>= 3.4.0)

Imports parallel, R2HTML, fitdistrplus, moments, copula, scatterplot3d, methods

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Robert Bear [aut], Kailan Shang [aut, cre], Hai You [aut], Brian Fannin [ctb]

Maintainer Kailan Shang <klshang81@gmail.com>

Repository CRAN

Date/Publication 2020-01-13 09:20:13 UTC

R topics documented:

CDFPlot ................................................................. 4
ChiSqrTest ............................................................ 5
### R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>claimdata</td>
<td>5</td>
</tr>
<tr>
<td>claimFitting</td>
<td>6</td>
</tr>
<tr>
<td>claimSample</td>
<td>8</td>
</tr>
<tr>
<td>claimSimulation</td>
<td>8</td>
</tr>
<tr>
<td>ClaimType-class</td>
<td>10</td>
</tr>
<tr>
<td>copulaDataPlot</td>
<td>11</td>
</tr>
<tr>
<td>copulaFit</td>
<td>12</td>
</tr>
<tr>
<td>copulaFitPlot</td>
<td>13</td>
</tr>
<tr>
<td>CopulaObj-class</td>
<td>14</td>
</tr>
<tr>
<td>copulaPlot</td>
<td>14</td>
</tr>
<tr>
<td>copulaSample</td>
<td>15</td>
</tr>
<tr>
<td>Density</td>
<td>16</td>
</tr>
<tr>
<td>DevFac-class</td>
<td>17</td>
</tr>
<tr>
<td>Distribution-class</td>
<td>18</td>
</tr>
<tr>
<td>doPlot</td>
<td>18</td>
</tr>
<tr>
<td>doSample</td>
<td>19</td>
</tr>
<tr>
<td>dtbeta</td>
<td>20</td>
</tr>
<tr>
<td>dtempirical</td>
<td>21</td>
</tr>
<tr>
<td>dtexp</td>
<td>22</td>
</tr>
<tr>
<td>dtgamma</td>
<td>23</td>
</tr>
<tr>
<td>dtgeom</td>
<td>24</td>
</tr>
<tr>
<td>dtlnorm</td>
<td>25</td>
</tr>
<tr>
<td>dtbinom</td>
<td>26</td>
</tr>
<tr>
<td>dtnorm</td>
<td>27</td>
</tr>
<tr>
<td>dtpareto</td>
<td>28</td>
</tr>
<tr>
<td>dtpois</td>
<td>29</td>
</tr>
<tr>
<td>dweibull</td>
<td>30</td>
</tr>
<tr>
<td>expectZeros</td>
<td>31</td>
</tr>
<tr>
<td>FitDist-class</td>
<td>31</td>
</tr>
<tr>
<td>fitPlot</td>
<td>32</td>
</tr>
<tr>
<td>getCopula</td>
<td>33</td>
</tr>
<tr>
<td>getIndex</td>
<td>34</td>
</tr>
<tr>
<td>getObservation</td>
<td>34</td>
</tr>
<tr>
<td>getTrend</td>
<td>35</td>
</tr>
<tr>
<td>Index-class</td>
<td>36</td>
</tr>
<tr>
<td>KSTest</td>
<td>36</td>
</tr>
<tr>
<td>mpareto</td>
<td>37</td>
</tr>
<tr>
<td>nloglik</td>
<td>38</td>
</tr>
<tr>
<td>observationPlot</td>
<td>38</td>
</tr>
<tr>
<td>PDFPlot</td>
<td>39</td>
</tr>
<tr>
<td>pempirical</td>
<td>40</td>
</tr>
<tr>
<td>plotText</td>
<td>41</td>
</tr>
<tr>
<td>PPPPlot</td>
<td>41</td>
</tr>
<tr>
<td>Probability</td>
<td>42</td>
</tr>
<tr>
<td>QQPlot</td>
<td>43</td>
</tr>
<tr>
<td>Quantile</td>
<td>44</td>
</tr>
<tr>
<td>rreopen</td>
<td>45</td>
</tr>
<tr>
<td>sampleKurtosis</td>
<td>46</td>
</tr>
</tbody>
</table>
R topics documented:

- sampleMean ......................................................... 47
- sampleSd .......................................................... 47
- sampleSkew .......................................................... 48
- setAnnualizedRate<- .............................................. 48
- setCopulaParam<- .................................................. 49
- setCopulaType<- .................................................... 50
- setDevFac .......................................................... 50
- setDf<- .............................................................. 51
- setDimension<- ..................................................... 52
- setDispstr<- ......................................................... 52
- setEmpirical<- ...................................................... 53
- setFacModel<- ....................................................... 54
- setFitdata ........................................................... 54
- setfitmethod<- ...................................................... 55
- setFittedDist<- ..................................................... 56
- setfreq<- ............................................................ 57
- setFun<- ............................................................. 58
- setID<- .............................................................. 58
- setidate<- ........................................................... 59
- setifreq<- ........................................................... 60
- setIndex ............................................................. 61
- setMarginal<- ......................................................... 62
- setMeanList<- ....................................................... 62
- setMin .............................................................. 63
- setMonthlyIndex<- .................................................. 63
- setObservation<- .................................................... 64
- setParams<- .......................................................... 64
- setParas<- ........................................................... 65
- setprobs<- ........................................................... 66
- setRange<- ........................................................... 66
- setRectangle .......................................................... 67
- setSeasonality<- ..................................................... 68
- setStartDate<- ........................................................ 68
- setTabulate<- ......................................................... 69
- setTrend<- ............................................................ 69
- setTrialDist<- ....................................................... 70
- setTrialDistErr<- .................................................... 70
- setTruncated<- ...................................................... 71
- setUpperKeep ........................................................ 71
- setUpperTriangle ..................................................... 72
- setVolList<- .......................................................... 72
- setXname<- ........................................................... 73
- setYearlyIndex<- .................................................... 73
- shiftIndex .......................................................... 74
- simP0 ............................................................... 74
- simReport ........................................................... 75
- simSummary .......................................................... 75
- simTriangle ........................................................... 76
CDFPlot

Plotting the CDF of data and fitted distribution

Description

Plotting the CDF of data and fitted distribution

Usage

CDFPlot(object, ...)

## S4 method for signature 'FitDist'

CDFPlot(object, n = missing)

Arguments

object

FitDist Object

...

Additional function arguments

n

Number of samples, should not be used in current setting

Examples

library(cascsim)
data(claimdata)

# frequency fitting example
findex <- new("Index", startDate = as.Date("2012-01-01"), tabulate=TRUE, monthlyIndex = c(rep(1,11), cumprod(c(1,rep(1.5*(1/12),11))),cumprod(c(1.5,rep((1.3/1.5)^(1/12),11)))),
    cumprod(c(1.3,rep((1.35/1.3)^((1/12),11))),cumprod(c(1.35,rep((1.4/1.35)^(1/12),11)))),1.4))
rawdata <- as.data.frame(as.Date(claimdata[,"LoB"]=="Auto" &
    claimdata[,"Type"]=="H",]$occurrenceDate))
colnames(rawdata)<-"occurrenceDate"
xFit <- new("FitDist", observation=rawdata, trend=findex,startDate = as.Date("2012-01-01"),
    method="mle",ifreq=TRUE,idate=TRUE, freq="Monthly")
xFit <- setFitdata(xFit)
setTrialDist(xFit) <- new("Poisson")
xFit@output

CDFPlot(xFit)
ChiSqrTest

Chi-Squared Test

Description
Chi-Squared Test

Usage
ChiSqrTest(object, ...)

## S4 method for signature 'FitDist'
ChiSqrTest(object)

Arguments

object FitDist Object
... Additional function arguments

claimdata Sample Claim Data

Description
A dataset containing about 10,000 simulated claim records from 2012 to 2016 for illustration. The variables are as follows:

Usage
data(claimdata)

Format
A data frame with 10030 rows and 15 variables

Details
- ClaimID. Claim ID
- LoB. Line of Business (Auto, Liab, Property)
- Type. Claim Type (N: Normal, H: High)
- status. Current Claim Status (Closed, Open)
- occurrenceDate. Claim Occurrence Date
- reportDate. Claim Report Date
- incurredLoss. Incurred Loss. For closed claim, it is the ultimate loss. For open claim, it is the estimated or booked loss.
- osRatio. Outstanding Ratio
- settlementDate. Claim Settlement Date.
- Paid. Paid Loss by the valuation date. It equals incurredLoss * (1-osRatio)
- totalLoss. Total loss before deductible and limit. If not available, it will be set as incurredLoss and not used for fitting.
- Deductible. Deductible applied to the claim.
- Limit. Limit applied to the claim.
- LAE. Loss adjustment expense at the claim level. It can be omitted if indemnity and LAE are modeled together as incurred loss.
- claimLiability. Indicating whether the claim is invalid and leads to zero payment. It excludes valid claims that are smaller than deductibles.

---

**claimFitting**

**Claim data fitting analysis at line/type/status level**

**Description**

Claim data fitting analysis at line/type/status level

**Usage**

```r
claimFitting(object, claimData, ...)  
```

### S4 method for signature 'Simulation,data.frame'

```r
claimFitting(object, claimData,  
  startDate = as.Date("2012-01-01"),  
  evaluationDate = as.Date("2016-12-31"),  
  lineList = object@lines,  
  typeList = object@types,  
  discreteDist = c("Poisson",  
                   "NegativeBinomial", "Geometric"),  
  continuousDist = c("Normal",  
                     "Lognormal", "Pareto", "Weibull", "Gamma", "Uniform", "Exponential"),  
  copulaList = c("normal"),  
  fReportLag = TRUE,  
  fSettlementLag = TRUE,  
  fFrequency = TRUE,  
  fSeverity = TRUE,  
  fSSRCorrelation = TRUE,  
  fFreqCorrelation = TRUE,  
  copulaTest = TRUE,  
  iTotalLoss = TRUE,  
  fDeductible = TRUE,  
  fLimit = TRUE,  
  check = TRUE)
```

**Arguments**

- `object` Simulation object
- `claimData` claim data including existing claims for RBNER and claim reopening analysis
- `...` Additional parameters that may or may not be used.
- `startDate` Date after which claims are analyzed;
**claimFitting**

- **evaluationDate** Date of evaluation for existing claims and IBNR;
- **lineList** List of business lines to be included in claim fitting;
- **typeList** List of claim types to be included in claim fitting;
- **discreteDist** List of discrete distributions to try fitting (report lag, settlement lag, frequency);
- **continuousDist** List of continuous distribution to try fitting (severity);
- **copulaList** List of copula to try fitting;
- **fReportLag** Boolean variable to indicate whether report lag needs to be fitted;
- **fSettlementLag** Boolean variable to indicate whether settlement lag needs to be fitted;
- **fFrequency** Boolean variable to indicate whether monthly frequency needs to be fitted;
- **fSeverity** Boolean variable to indicate whether severity needs to be fitted;
- **fSSRCorrelation** Boolean variable to indicate whether copula among severity, report lag and settlement lag needs to be fitted;
- **fFreqCorrelation** Boolean variable to indicate whether copula among frequencies of business lines needs to be fitted.
- **copulaTest** Whether to test copula. The testing could take a very long time;
- **iTotalLoss** Boolean variable to indicate whether total loss before deductible and limit is available for severity fitting;
- **fDeductible** Boolean variable to indicate whether deductible empirical distribution needs to be fitted;
- **fLimit** Boolean variable to indicate whether limit empirical distribution needs to be fitted;
- **check** Boolean variable to indicate whether graph of each tried distribution fitting needs to be generated and saved.

**Examples**

```r
library(cascsim)
data(claimdata)
lines<-c("Auto")
types<-c("N")
# exposure index
index1 <- new("Index", monthlyIndex=c(rep(1,11), cumprod(c(1,rep(1.5*(1/12),11)))),
cumprod(c(1.5,rep((1.3/1.5)^(1/12),11))),
cumprod(c(1.3,rep((1.35/1.3)^(1/12),11))),cumprod(c(1.35,rep((1.4/1.35)^(1/12),11))),rep(1.4,301)))
# severity index
index2 <- new("Index", monthlyIndex=c(cumprod(c(1,rep(1.03*(1/12),59))),rep(1.03^5,300)))
objan <- new("ClaimType", line="Auto", claimType="N", exposureIndex=index1, severityIndex=index2)
objlist <- list(objan)
simobj <- new("Simulation", lines=lines, types=types, claimobjs=objlist, iFit=TRUE, iCopula=FALSE, iReport=TRUE, workingFolder=tempdir())
simobj <- claimFitting(simobj,claimdata,fSSRCorrelation = FALSE, fSettlementLag = FALSE)
```
claimSample

**Claim simulation at line/type/status level**

**Description**
Claim simulation at line/type/status level

**Usage**

```
claimSample(object, ...)  
```

```r
## S4 method for signature 'ClaimType'
claimSample(object, claimData = data.frame(),
    startDate = as.Date("2012-01-01"),
    evaluationDate = as.Date("2016-12-31"))
```

**Arguments**

- **object**: ClaimType object
- **...**: Additional parameters that may or may not be used.
- **claimData**: claim data including existing claims for RBNER and claim reopening analysis;
- **startDate**: Date from which claim data is included in the analysis;
- **evaluationDate**: Date of evaluation.

**Examples**

```r
# run time is about 12s(>10s) and is commented out here to avoid long waiting time
#library(cascsim)
data(claimdata)
#IBNR simulation
#claimobj <- new("ClaimType", line="Auto",claimType="N",iRBNER=FALSE,iOPEN=FALSE,
#iIBNR=TRUE,iUPR=FALSE,
#IBNRfreqIndex=new("Index",startDate=as.Date("2016-01-01"),
#monthlyIndex=rep(30,12)),iCopula=TRUE)
#ibnrdata <- claimSample(claimobj,claimdata)
#ibnrdata
```

---

claimSimulation

**Claim simulation at line/type/status level**

**Description**
Claim simulation at line/type/status level
Usage

claimSimulation(object, ...)  

## S4 method for signature 'Simulation'
claimSimulation(object, claimData = data.frame(),
startDate = as.Date("2012-01-01"),
evaluationDate = as.Date("2016-12-31"),
futureDate = as.Date("2017-12-31"), append = TRUE)

Arguments

object  
Simulation object

...  
Additional parameters that may or may not be used.

claimData  
claim data including existing claims for RBNER and claim reopneness analysis;

startDate  
Date after which claims are analyzed;

evaluationDate  
Date of evaluation for existing claims and IBNR;

futureDate  
Date of evaluation for UPR (future claims);

append  
Boolean variable to indicate whether existing simulation results need to be kept.

Examples

library(cascsim)
data(claimdata)
lines <- c("Auto")
types <- c("N")
AutoN <- new("ClaimType", line = "Auto", claimType = "N")
AutoN@exposureIndex <- setIndex(new("Index", indexID="I1", tabulate= FALSE,
startDate=as.Date("2012-01-01"), annualizedRate = 0))  # level exposure across time
AutoN@frequency <- new("Poisson", p1 =50)
AutoN@severityIndex <- setIndex(new("Index", indexID="I2", tabulate= FALSE,
startDate=as.Date("2012-01-01"), annualizedRate = 0.02))  #assuming a 2% annual inflation
AutoN@severity <- new("Lognormal", p1 =2, p2 =3)
AutoN@deductible <- new("Empirical", empirical=matrix(c(0,1,100,100),2,2))
AutoN@limit <- new("Empirical", empirical=matrix(c(0,1,1e0,1e0),2,2))
AutoN@reportLag <- new("Exponential", p1 =0.1)
AutoN@settlementLag <- new("Exponential", p1 =0.05)
AutoN@Copula <- TRUE #use copula
AutoN@ssrCopula <- new("CopulaObj", type ="normal", dimension = 3,
param = c(0.1,0.2,0.1))#A Gaussian Copula
AutoN@ssrCopula@marginal <- c(AutoN@severity,AutoN@settlementLag,AutoN@reportLag)
AutoN@laeDevFac <- new("DevFac",FacID="F1",FacModel= TRUE,fun="linear",
paras =c(5,1.5,0.005,1.2,3))
AutoN@fIBNER <- new("DevFac",FacID="D1",FacModel= FALSE,
meanList =c(1.2,1.15,1.1,1.05,1),volList =c(0,0,0,0,0))
AutoN@reopen <- new("DevFac",FacID="D2",FacModel= FALSE,
meanList =c(0.02,0.015,0.01,0.005,0),volList =c(0.003, 0.002, 0.001, 0.001, 0))
AutoN@roDevFac <- new("DevFac",FacID="D3",FacModel= FALSE,
meanList =c(0.00589,0.0037,0.00632,0.00815,0))
AutoN@reopenLag <- new("Exponential", p1 =0.01)
AutoN@resettleLag <- new("Exponential", p1 =0.25)
simobj <- new("Simulation", lines=lines, types=types,
claimobjs= list(AutoN),workingFolder=tempdir())
simobj@simNo <- 1
simobj@iRBNER <-FALSE
simobj@iROPEN <-FALSE
simobj@iIBNR <-TRUE
simobj@iUPR <-FALSE
simdata <- claimSimulation(simobj,claimdata, startDate = as.Date("2012-01-01"),
evaluationDate = as.Date("2016-12-31"), futureDate = as.Date("2017-12-31"))

ClaimType-class

An S4 class to represent a claim type.

Description

An S4 class to represent a claim type.

Slots

simno The simulation index.
line A string to identify the business line that the claim belongs to.
claimType A string to identify the type of the claim. It further classifies the claims within a business line. For example, the type could be based on the size of the loss.
iRBNER A Boolean variable to indicate whether RBNER (open claims) should be simulated.
iROPEN A Boolean variable to indicate whether claim reopen should be simulated.
iIBNR A Boolean variable to indicate whether IBNR claims should be simulated.
iUPR A Boolean variable to indicate whether future claims should be simulated.
fIBNER IBNER development factor.
severity Severity distribution.
frequency Frequency distribution.
reportLag Report lag distribution.
settlementLag Settlement lag distribution.
reopen Claim reopen probability based on the number of years after settlement till valuation date.
reopenLag Reopen lag distribution.
resettleLag Resettlement lag distribution.
roDevFac Reopened claim development factor.
ioDevFac A numeric variable to indicate the method of loss development for open claim severity.

    1: Conditional distribution based on paid loss; 2: conditional distribution based on incurred loss; 3: year-to-year development factors
irDevFac  A numeric variable to indicate the method of loss development for claim reopen severity simulation. 1: Conditional distribution based on paid loss; 2: conditional distribution based on incurred loss; 3: year-to-year development factors

freqIndex  Frequency distribution time index.

severityIndex  Severity distribution time index.

exposureIndex  Exposure time index for IBNR or UPR.

iCopula  Whether copula is used to model severity, report lag and settlement lag.

ssrCopula  Copula object used for severity, report lag and settlement lag.

sdata  Indicating whether only closed claims (CLOSED) or closed + open claims (ALL) will be used for severity fitting.

p0  An yearly table that controls the probability of invalid claim, excluding these valid claims less than deductible based on development year. It is based on the DevFac class.

copulaDataPlot  Experience data plotting.

Description

Experience data plotting.

Usage

copulaDataPlot(object, ...)

## S4 method for signature 'CopulaObj'
copulaDataPlot(object)

Arguments

object  Copula Object

...  Additional parameters that may or may not be used

Examples

library(cascsim)
dist1 <- new("Pareto", p1=20, p2=3)
dist2 <- new("Normal", p1=5, p2=3, min=0, max=20, truncated=TRUE)
nom.cop <- new("CopulaObj", param=c(0.5), marginal=list(dist1=dist1, dist2=dist2), dimension=2)
setObservation(nom.cop) <- copulaSample(nom.cop, 100)
copulaDataPlot(nom.cop)
Description

Copula fitting

Usage

copulaFit(object, ...)

## S4 method for signature 'CopulaObj'
copulaFit(object)

Arguments

object Copula Object

... Additional parameters that may or may not be used

Examples

library(cascsim)
#Prepare pseudo observation data
library(copula)
dist1<-new("Pareto",p1=20,p2=3)
dist2<-new("Normal",p1=5,p2=3,min=0,max=20,truncated=TRUE)
dist3<-new("Lognormal",p1=2,p2=1,min=0,max=100,truncated=TRUE)
normal.cop <- normalCopula(c(0.6, 0.36, 0.6), dim=3, dispstr="un")
x <- rCopula(1000, normal.cop)
x[,1]<-Quantile(dist1,x[,1])
x[,2]<-Quantile(dist2,x[,2])
x[,3]<-Quantile(dist3,x[,3])
#Create Copula Object and Fit it to observation data without goodness of fit test
nom.cop <- new("CopulaObj", param=c(0.5,0.5,0.5),marginal=list(dist1=dist1,dist2=dist2,dist3=dist3),
dimension=3,observation=x,fittest=FALSE)
nom.cop <- copulaFit(nom.cop)
nom.cop@coutput
#Create Copula Object and Fit it to observation data with goodness of fit test
clayton.cop <- claytonCopula(c(3), dim=2)
x <- rCopula(1000, clayton.cop)
x[,1]<-Quantile(dist1,x[,1])
x[,2]<-Quantile(dist2,x[,2])
cla.cop <- new("CopulaObj", type="clayton",param=c(3),
marginal=list(dist1=dist1,dist2=dist2),dimension=2,observation=x,fittest=TRUE)
cla.cop <- copulaFit(cla.cop)
cla.cop@coutput
Description

Visualization Copula fitting

Usage

copulaFitPlot(object, ...)

## S4 method for signature 'CopulaObj'
copulaFitPlot(object)

Arguments

object  
Copula Object
...
Additional parameters that may or may not be used

Examples

library(cascsim)
#Prepare pseudo observation data
library(copula)
dist1<-new("Pareto",p1=20,p2=3)
dist2<-new("Normal",p1=5,p2=3,min=0,max=20, truncated=TRUE)
dist3<-new("Lognormal",p1=2,p2=1,min=0,max=100, truncated=TRUE)
normal.cop <- normalCopula(c(0.6, 0.36, 0.6), dim=3, dispstr="un")
x <- rCopula(1000, normal.cop)
x[,1]<-Quantile(dist1,x[,1])
x[,2]<-Quantile(dist2,x[,2])
x[,3]<-Quantile(dist3,x[,3])
#Create Copula Object and Fit it to observation data without goodness of fit test
nom.cop <- new("CopulaObj", param=c(0.5,0.5,0.5),marginal=list(dist1=dist1,dist2=dist2,dist3=dist3),
dimension=3,observation=x,fittest=FALSE)
nom.cop <- copulaFit(nom.cop)
copulaFitPlot(nom.cop)
#Create Copula Object and Fit it to observation data with goodness of fit test
clayton.cop <- claytonCopula(c(3), dim=2)
x <- rCopula(1000, clayton.cop)
x[,1]<-Quantile(dist1,x[,1])
x[,2]<-Quantile(dist2,x[,2])
cla.cop <- new("CopulaObj", type="clayton",param=c(3),marginal=list(dist1=dist1,dist2=dist2),
dimension=2,observation=x,fittest=TRUE)
cla.cop <- copulaFit(cla.cop)
copulaFitPlot(cla.cop)
CopulaObj-class

An S4 class to represent a copula object to model the correlation.

Description

An S4 class to represent a copula object to model the correlation.

Slots

type The type of the copula object.
para A numeric vector that contains copula parameter(s).
marginal A list of Distribution objects.
dispstr The format of symmetric positive definite matrix used by elliptical copula (Normal Copula, t Copula). The default is "un" for unstructured. Other choices include "ex" for exchangeable, "ar1" for AR(1), and "toep" for Toeplitz (toeplitz).
df The number of degrees of freedom used in t Copula.
observation A matrix that contains the experience data for copula fitting.
fitmethod The method of copula fitting. Default is "mpl": maximum pseudo-likelihood estimator. Others include "ml": maximum likelihood assuming it is the true distribution; "itau": inversion of Kendall's tau estimator; "irho": inversion of Spearman's rho estimator.
fittest Whether to run goodness of fit test for copula fitting. Goodness of fit test could take a long time to finish.
fitsucc Whether a copula fitting is successful.
coutput Goodness of fit results.
info A character string that contains additional information of the copula to identify line/type/frequency/time lag/severity.

copulaPlot

Copula plotting. Only for 2 or 3 variables

Description

Copula plotting. Only for 2 or 3 variables

Usage

copulaPlot(object, ...)

## S4 method for signature 'CopulaObj'
copulaPlot(object)
**copulaSample**

**Arguments**

\[
\begin{align*}
\text{object} & : \text{Copula Object} \\
\ldots & : \text{Additional parameters that may or may not be used}
\end{align*}
\]

**Examples**

```r
library(cascsim)
dist1 <- new("Pareto", p1=20, p2=3)
dist2 <- new("Normal", p1=5, p2=3, min=0, max=20, truncated=TRUE)
nom.cop <- new("CopulaObj", param=c(0.5), marginal=list(dist1=dist1, dist2=dist2), dimension=2)
copulaPlot(nom.cop)
```

copulaSample

*Copula sampling. It will generate correlated variables or percentiles when marginal distributions are not specified.*

**Description**

Copula sampling. It will generate correlated variables or percentiles when marginal distributions are not specified.

**Usage**

```r
copulaSample(object, n, ...)
```

```r
## S4 method for signature 'CopulaObj,numeric'
copulaSample(object, n)
```

**Arguments**

\[
\begin{align*}
\text{object} & : \text{Copula Object} \\
\text{n} & : \text{Number of samples} \\
\ldots & : \text{Additional parameters that may or may not be used}
\end{align*}
\]

**Examples**

```r
library(cascsim)
dist1 <- new("Pareto", p1=20, p2=3)
dist2 <- new("Normal", p1=5, p2=3, min=0, max=20, truncated=TRUE)
nom.cop <- new("CopulaObj", param=c(0.5), marginal=list(dist1=dist1, dist2=dist2), dimension=2)
copulaSample(nom.cop, 100)
```
Density function.

Description
Density function.

Usage
Density(object, x, ...)

## S4 method for signature 'Normal'
Density(object, x, log = FALSE)

## S4 method for signature 'Beta'
Density(object, x, log = FALSE)

## S4 method for signature 'Exponential'
Density(object, x, log = FALSE)

## S4 method for signature 'Gamma'
Density(object, x, log = FALSE)

## S4 method for signature 'Geometric'
Density(object, x, log = FALSE)

## S4 method for signature 'Lognormal'
Density(object, x, log = FALSE)

## S4 method for signature 'NegativeBinomial'
Density(object, x, log = FALSE)

## S4 method for signature 'Pareto'
Density(object, x, log = FALSE)

## S4 method for signature 'Poisson'
Density(object, x, log = FALSE)

## S4 method for signature 'Uniform'
Density(object, x, log = FALSE)

## S4 method for signature 'Weibull'
Density(object, x, log = FALSE)

## S4 method for signature 'Empirical'
Density(object, x, log = FALSE)
DevFac-class

Arguments

object Distribution Object
x Variable value
... Additional function arguments
log Boolean variable to indicate whether to return log of probability

Examples

xPareto <- new("Pareto",p1=20,p2=3)
Density(xPareto,50)

DevFac-class An S4 class to represent a loss development schedule.

Description

An S4 class to represent a loss development schedule.

Slots

FacID A character string to identify the loss development schedule.
FacModel A boolean to indicate whether the loss development schedule is described as a model (TRUE) or a list of value (FALSE).
fun A character string that indicates the model format in link function. Currently identity(linear), inverse(reciprocal linear), log(exponential), and exponential(loglinear) link functions(models) are supported. It is only used when model == TRUE.
distType A character string that indicates the distribution of development factors. Currently normal, lognormal, and gamma distributions are supported. It is only used when model == FALSE.
xname A vector that includes the names of explanatory variables. They will have to be matched exactly to the claim data file. It is only used when model == TRUE.
paras A vector that contains the parameters of the model. It is only used when model == TRUE.
meanList A vector that contains the mean yearly development factor if distribution type is Normal. It is mu for Lognormal distribution and shape for Gamma distribution. It is only used when model == FALSE.
volList A vector that contains the volatility of yearly development factor if distribution type is Normal. It is sigma for Lognormal distribution and scale for Gamma distribution. It is used for simulating IBNER factors. It is only used when model == FALSE.
Distribution-class  

An S4 class to represent a distribution, either parametric or non-parametric.

Description

An S4 class to represent a distribution, either parametric or non-parametric.

Slots

- p1  A number for the value of the first parameter (default: 0.8).
- p2  A number for the value of the second parameter (default: 1).
- p3  A number for the value of the third parameter (default: 0).
- empirical A matrix that defines an empirical distribution with values and probabilities.
- min A number that defines the minimum value of the variable (default: 1e-8 considering it is used for frequency and severity modeling).
- max A number that defines the maximum value of the variable (default: 1e8).
- fitsucc Whether a distribution fitting is successful.
- info A character string that contains additional information of the distribution to identify line/type/frequency or severity.

doPlot

Plot function.

Description

Plot function.

Usage

doPlot(object, ...)

## S4 method for signature 'Distribution'
doPlot(object)

Arguments

- object  Object
- ...  Additional function arguments

Examples

```r
xPareto <- new("Pareto",p1=20,p2=3)
doPlot(xPareto)
```
doSample

Sampling from the distribution.

Description

Sampling from the distribution.

Usage

doSAMPLE(object, n, ...)

## S4 method for signature 'Normal,numeric'
doSample(object, n)

## S4 method for signature 'Beta,numeric'
doSample(object, n)

## S4 method for signature 'Exponential,numeric'
doSample(object, n)

## S4 method for signature 'Gamma,numeric'
doSample(object, n)

## S4 method for signature 'Lognormal,numeric'
doSample(object, n)

## S4 method for signature 'Pareto,numeric'
doSample(object, n)

## S4 method for signature 'Poisson,numeric'
doSample(object, n)

## S4 method for signature 'NegativeBinomial,numeric'
doSample(object, n)

## S4 method for signature 'Geometric,numeric'
doSample(object, n)

## S4 method for signature 'Uniform,numeric'
doSample(object, n)

## S4 method for signature 'Weibull,numeric'
doSample(object, n)

## S4 method for signature 'Empirical,numeric'
doSample(object, n)
Arguments

- **object**: A Distribution Object
- **n**: Number of samples
- **...**: Additional function arguments

Examples

```r
xPareto <- new("Pareto", p1=20, p2=3)
doSample(xPareto, 10000)
```

---

**dtbeta**

*Density function of Truncated Beta Distribution*

Description

- Density function of Truncated Beta Distribution
- Cumulative probability function of Truncated Beta Distribution
- Quantile function of Truncated Beta Distribution $\max(0, \min(\text{claim}, \text{limit}) - \text{deductible})$
- Random generation of Truncated Beta Distribution $\max(0, \min(\text{claim}, \text{limit}) - \text{deductible})$

Usage

- `dtbeta(x, shape1, shape2, ncp = 0, min = 0, max = 1)`
- `ptbeta(q, shape1, shape2, ncp = 0, min = 0, max = 1)`
- `qtbeta(p, shape1, shape2, ncp = 0, min = 0, max = 1)`
- `rtbeta(n, shape1, shape2, ncp = 0, min = 0, max = 1)`

Arguments

- **x**: Value of the variable after deductible and limit $\max(0, \min(\text{claim}, \text{limit}) - \text{deductible})$
- **shape1**: distribution parameter
- **shape2**: distribution parameter
- **ncp**: non-centrality parameter (Default: 0)
- **min**: Left truncation deductible
- **max**: Right truncation limit
- **q**: Value of the variable after deductible and limit $\max(0, \min(\text{claim}, \text{limit}) - \text{deductible})$
- **p**: Value of the probability
- **n**: Number of samples
**Examples**

- `dtbeta(0.6,1,2)`
- `ptbeta(0.5,1,2)`
- `qtbeta(0.5,1,2)`
- `rtbeta(100,1,2)`

---

**dtempirical**

*Density function of truncated empirical distribution*

---

**Description**

**Density function of truncated empirical distribution**

**Cumulative probability function of truncated empirical distribution**

**Quantile function of truncated empirical distribution** $\max(0, \min(\text{claim}, \text{limit}) - \text{deductible})$

**Random generation of Truncated empirical distribution** $\max(0, \min(\text{claim}, \text{limit}) - \text{deductible})$

**Usage**

- `dtempirical(x, cdf, min = 0, max = 1e+09)`
- `ptempirical(q, cdf, min = 0, max = 1e+05)`
- `qtempirical(p, cdf, min = 0, max = 1e+05)`
- `rtempirical(n, cdf, min = 0, max = 1e+05)`

**Arguments**

- **x**: Value of the variable after deductible and limit $\max(0, \min(\text{claim}, \text{limit}) - \text{deductible})$
- **cdf**: Empirical distribution (cdf for continuous distribution and pmf for discrete distribution)
- **min**: Left truncation deductible
- **max**: Right truncation limit
- **q**: Value of the variable after deductible and limit $\max(0, \min(\text{claim}, \text{limit}) - \text{deductible})$
- **p**: Value of the probability
- **n**: Number of samples

**Examples**

- #discrete distribution
  
  ```r
  dtempirical(3,matrix(c(0.1,0.2,0.3,0.05,0.05,0.2,0.1,1:6,10),7,2),3,100)
  ```

- #continuous distribution
  
  ```r
  dtempirical(30,matrix(c(seq(0.01,1,0.01),qnorm(seq(0.01,1,0.01),30,20)),100,2),200,10000000)
  ```

- #discrete distribution
  
  ```r
  ptempirical(c(3,5,10),matrix(c(0.1,0.2,0.3,0.05,0.05,0.2,0.1,1:6,10),7,2),3,100)
  ```
#continuous distribution
ptempirical(350,matrix(c(seq(0.01,1,0.01),cumprod(c(1,rep(1.199))))),100,2),200,10000000)
#discrete distribution
qtempirical(c(0.3,0.65,1),matrix(c(0.1,0.2,0.3,0.05,0.05,0.2,0.1,1:6,10),7,2),3,100)
#continuous distribution
qtempirical(c(0.3,0.65,0.8),matrix(c(seq(0.01,1,0.01)),
cumprod(c(1,rep(1.1,99)))),100,2),200,10000000)
#discrete distribution
rtempirical(100,matrix(c(0.1,0.2,0.3,0.05,0.05,0.2,0.1,1:6,10),7,2),3,100)
#continuous distribution
rtempirical(100,matrix(c(seq(0.01,1,0.01)),cumprod(c(1,rep(1.1,99)))),100,2),200,10000000)

dtexp

Density function of Truncated Exponential Distribution

Description
Density function of Truncated Exponential Distribution
Cumulative probability function of Truncated Exponential Distribution
Quantile function of Truncated Exponential Distribution max(0,min(claim,limit)-deductible)
Random generation of Truncated Exponential Distribution max(0,min(claim,limit)-deductible)

Usage
dtexp(x, rate, min = 0, max = 1e+09)

ptexp(q, rate, min = 0, max = 1e+09)

qtexp(p, rate, min = 0, max = 1e+09)

rtexp(n, rate, min = 0, max = 1e+09)

Arguments
x
rate
min
max
q
p
n
Value of the variable after deductible and limit max(0,min(claim,limit)-deductible)
Distribution parameter
Left truncation deductible
Right truncation limit
Value of the variable after deductible and limit max(0,min(claim,limit)-deductible)
Value of the probability
Number of samples

Examples
dtexp(5,0.1)
ptexp(5,0.1)
qtexp(0.5,0.1)
rtexp(100,0.1)
The density function of Truncated Gamma Distribution is denoted by `dtgamma`. The cumulative probability function of Truncated Gamma Distribution is denoted by `ptgamma`. The quantile function of Truncated Gamma Distribution is denoted by `qtgamma`. Random generation of Truncated Gamma Distribution is denoted by `rtgamma`.

**Usage**

- `dtgamma(x, shape, scale, min = 0, max = 1e+09)`
- `ptgamma(q, shape, scale, min = 0, max = 1e+09)`
- `qtgamma(p, shape, scale, min = 0, max = 1e+09)`
- `rtgamma(n, shape, scale, min = 0, max = 1e+09)`

**Arguments**

- `x`: Value of the variable after deductible and limit max(0,min(claim,limit)-deductible)
- `shape`: Shape parameter
- `scale`: Scale parameter
- `min`: Left truncation deductible
- `max`: Right truncation limit
- `q`: Value of the variable after deductible and limit max(0,min(claim,limit)-deductible)
- `p`: Value of the probability
- `n`: Number of samples

**Examples**

```r
dtgamma(2,3,2)
ptgamma(2,3,2)
qtgamma(0.5,3,2)
rtgamma(100,3,2)
```
**dtgeom**

**Density function of Truncated Geometric Distribution**

**Cumulative probability function of Truncated Geometric Distribution**

**Quantile function of Truncated Geometric Distribution** \( \max(0, \min(\text{claim, limit}) - \text{deductible}) \)

**Random generation of Truncated Geometric Distribution** \( \max(0, \min(\text{claim, limit}) - \text{deductible}) \)

**Usage**

\[
\begin{align*}
dtgeom(x, \text{prob}, \text{min} = 0, \text{max} = 1e+09) \\
ptgeom(q, \text{prob}, \text{min} = 0, \text{max} = 1e+09) \\
qtgeom(p, \text{prob}, \text{min} = 0, \text{max} = 1e+09) \\
rtgeom(n, \text{prob}, \text{min} = 0, \text{max} = 1e+09)
\end{align*}
\]

**Arguments**

- **x** Value of the variable after deductible and limit \( \max(0, \min(\text{claim, limit}) - \text{deductible}) \)
- **prob** Distribution parameter
- **min** Left truncation deductible
- **max** Right truncation limit
- **q** Value of the variable after deductible and limit \( \max(0, \min(\text{claim, limit}) - \text{deductible}) \)
- **p** Value of the probability
- **n** Number of samples

**Examples**

\[
\begin{align*}
dtgeom(3, 0.3) \\
ptgeom(3, 0.3) \\
qtgeom(0.7, 0.3) \\
rtgeom(100, 0.3)
\end{align*}
\]
**dtlnorm**

**Density function of Truncated Lognormal Distribution**

**Description**

Density function of Truncated Lognormal Distribution

Cumulative probability function of Truncated Lognormal Distribution

Quantile function of Truncated Lognormal Distribution max(0,min(claim,limit)-deductible)

Random generation of Truncated Lognormal Distribution max(0,min(claim,limit)-deductible)

**Usage**

```r
dtlnorm(x, meanlog, sdlog, min = 0, max = 1e+09)
ptlnorm(q, meanlog, sdlog, min = 0, max = 1e+09)
qtlnorm(p, meanlog, sdlog, min = 0, max = 1e+09)
rtlnorm(n, meanlog, sdlog, min = 0, max = 1e+09)
```

**Arguments**

- **x**: Value of the variable after deductible and limit max(0,min(claim,limit)-deductible)
- **meanlog**: Mean of the log of the distribution
- **sdlog**: Standard deviation of the log of the distribution
- **min**: Left truncation deductible
- **max**: Right truncation limit
- **q**: Value of the variable after deductible and limit max(0,min(claim,limit)-deductible)
- **p**: Value of the probability
- **n**: Number of samples

**Examples**

```r
dtlnorm(20, 3, 0.5)
ptlnorm(20, 3, 0.5)
qtlnorm(0.5, 3, 0.5)
rtlnorm(100, 3, 0.5)
```
**dtnbinom**

Density function of Truncated Negative Binomial Distribution

Description

Density function of Truncated Negative Binomial Distribution
Cumulative probability function of Truncated Negative Binomial Distribution
Quantile function of Truncated Negative Binomial Distribution max(0,min(claim,limit)-deductible)
Random generation of Truncated Negative Binomial Distribution max(0,min(claim,limit)-deductible)

Usage

```r
  dtnbinom(x, size, prob, min = 0, max = 1e+09)
  ptnbinom(q, size, prob, min = 0, max = 1e+09)
  qtnbinom(p, size, prob, min = 0, max = 1e+09)
  rtnbinom(n, size, prob, min = 0, max = 1e+09)
```

Arguments

- `x`: Value of the variable after deductible and limit max(0,min(claim,limit)-deductible)
- `size`: Number of successful trials
- `prob`: Probability of success in each trial
- `min`: Left truncation deductible
- `max`: Right truncation limit
- `q`: Value of the variable after deductible and limit max(0,min(claim,limit)-deductible)
- `p`: Value of the probability
- `n`: Number of samples

Examples

```r
  dtnbinom(230,100,0.3)
  ptnbinom(230,100,0.3)
  qtnbinom(0.5,100,0.3)
  rtnbinom(500,100,0.3)
```


**dt NOR M**

**Density function of Truncated Normal Distribution**

**Description**
Density function of Truncated Normal Distribution
Cumulative probability function of Truncated Normal Distribution
Quantile function of Truncated Normal Distribution max(0,min(claim,limit)-deductible)
Random generation of Truncated Normal Distribution max(0,min(claim,limit)-deductible)

**Usage**
dtnorm(x, mean, sd, min = 0, max = 1e+09)
ptnorm(q, mean, sd, min = 0, max = 1e+09)
qtnorm(p, mean, sd, min = 0, max = 1e+09)
rtnorm(n, mean, sd, min = 0, max = 1e+09)

**Arguments**
x Value of the variable after deductible and limit max(0,min(claim,limit)-deductible)
mean Mean of the untruncated Normal distribution
sd Standard deviation of the untruncated Normal distribution
min Left truncation (like deductible)
max Right truncation (like limit)
q Value of the variable after deductible and limit max(0,min(claim,limit)-deductible)
p Value of the probability
n Number of samples

**Examples**
dtnorm(0.5, 1, 2)
ptnorm(0.5, 1, 2)
qtnorm(0.5, 1, 2)
rtnorm(100, 1, 2)
**dtpareto**  

**Density function of Truncated Pareto Distribution**

**Description**

Density function of Truncated Pareto Distribution  
Cumulative probability function of Truncated Pareto Distribution  
Quantile function of Truncated Pareto Distribution  
Random generation of Truncated Pareto Distribution

**Usage**

\[
dtpareto(x, \text{xm}, \text{alpha}, \text{min} = \text{xm}, \text{max} = 1e+09)
\]

\[
ptpareto(q, \text{xm}, \text{alpha}, \text{min} = \text{xm}, \text{max} = 1e+09)
\]

\[
qtpareto(p, \text{xm}, \text{alpha}, \text{min} = \text{xm}, \text{max} = 1e+09)
\]

\[
rtpareto(n, \text{xm}, \text{alpha}, \text{min} = \text{xm}, \text{max} = 1e+09)
\]

**Arguments**

- **x**: Value of the variable after deductible and limit max(0, min(claim, limit)-deductible)  
- **xm**: Threshold value  
- **alpha**: Model parameter  
- **min**: Left truncation deductible  
- **max**: Right truncation limit  
- **q**: Value of the variable after deductible and limit max(0, min(claim, limit)-deductible)  
- **p**: Value of the probability  
- **n**: Number of samples

**Examples**

- `dtpareto(500,1000,2)`  
- `ptpareto(500,1000,2)`  
- `qtpareto(0.5,1000,2)`  
- `rtpareto(100,1000,2)`
### dt.poi

**Density function of Truncated Poisson Distribution**

**Cumulative probability function of Truncated Poisson Distribution**

**Quantile function of Truncated Poisson Distribution** $\max(0, \min(\text{claim, limit}) - \text{deductible})$

**Random generation of Truncated Poisson Distribution** $\max(0, \min(\text{claim, limit}) - \text{deductible})$

#### Usage

- `dt.poi(x, lambda, min = 0, max = 1e+09)`
- `pt.poi(q, lambda, min = 0, max = 1e+09)`
- `qt.poi(p, lambda, min = 0, max = 1e+09)`
- `rt.poi(n, lambda, min = 0, max = 1e+09)`

#### Arguments

- **x**: Value of the variable after deductible and limit $\max(0, \min(\text{claim, limit}) - \text{deductible})$
- **lambda**: Distribution parameter
- **min**: Left truncation deductible
- **max**: Right truncation limit
- **q**: Value of the variable after deductible and limit $\max(0, \min(\text{claim, limit}) - \text{deductible})$
- **p**: Value of the probability
- **n**: Number of samples

#### Examples

- `dt.poi(3, 5)`
- `pt.poi(3, 5)`
- `qt.poi(0.6, 5)`
- `rt.poi(100, 5)`
Density function of Truncated Weibull Distribution

Cumulative probability function of Truncated Weibull Distribution

Quantile function of Truncated Weibull Distribution max(0,min(claim,limit)-deductible)

Random generation of Truncated Weibull Distribution max(0,min(claim,limit)-deductible)

Usage

dtweibull(x, shape, scale, min = 0, max = 1e+09)

ptweibull(q, shape, scale, min = 0, max = 1e+09)

qtweibull(p, shape, scale, min = 0, max = 1e+09)

rtweibull(n, shape, scale, min = 0, max = 1e+09)

Arguments

x Value of the variable after deductible and limit max(0,min(claim,limit)-deductible)

shape Shape parameter

scale Scale parameter

min Left truncation deductible

max Right truncation limit

q Value of the variable after deductible and limit max(0,min(claim,limit)-deductible)

p Value of the probability

n Number of samples

Examples

dtweibull(2.5,2,3)

ptweibull(2.5,2,3)

qtweibull(0.5,2,3)

rtweibull(100,2,3)
expectZeros

Get the expected P0 based on settlement/close year.

Description

Get the expected P0 based on settlement/close year.

Usage

expectZeros(closeYear, zeroProb)

Arguments

closeYear Development years that claims are settled. It could be a number or a numeric vector.

zeroProb A vector that contains the P(0) based on development year.

Examples

zeroprob<-c(0.02,0.01,0.005,0.005,0.003,0)
expectZeros(c(2,3,6,9,100,1,2,3,4),zeroprob)

FitDist-class

An S4 class to represent distribution fitting.

Description

An S4 class to represent distribution fitting.

Slots

observation Raw data input containing loss sizes for severity analysis and number of losses for frequency analysis.

fitdata Processed data for distribution fitting. Frequency data may be provided as occurrence dates. The class will transform them into frequency data before distribution fitting.

trend Index object for detrending the data.

startDate Start date of claim data used for distribution fitting. The trend Index should also start from the same date (year-month).

endDate End date of claim data used for distribution fitting.

trail Trial Distribution object to start fitting.

fitted Fitted Distribution object.

reportLag Report lag distribution to adjust frequency data.

iLag Whether to adjust the frequency data with report lag distribution.
Distribution fitting method. Maximum likelihood estimation (mle), moment matching estimation (mme) and quantile matching estimation (qme) are available.

probs A vector containing the percentiles to be matched if qme is used for fitting.
ifreq A boolean indicating whether it is frequency data or severity data.
idate A boolean indicating whether frequency data is provided as occurrence dates (TRUE) or number of occurrences (FALSE).
datelist A vector containing occurrence dates. It could be a data field in a claim file.
freq A character string indicating the frequency: "Annual" or "Monthly".
ifDL A boolean indicating whether deductible and limit is considered in distribution fitting.
limit A vector containing the limit for each claim.
deductible A vector containing the deductible for each claim.
p0 A number that is the probability of having a zero-amount claim after deductible.
dof Degree of freedom.
psd A vector containing the standard deviation of parameter estimation. It is only available for mle.
aic Akaike information criterion.
bic Bayesian information criterion.
chisq Chi-Squared Test Statistic.
pchisq p-value of Chi-Squared Test.
ktest K-S Test Statistic. Only used for continuous distribution.
pkstest p-value of K-S Test. Only used for continuous distribution.
soutput Distribution fitting summary.

fitPlot

Compare the raw data and fitted distribution on density, CDF, Q-Q plot and P-P plot

Description

Compare the raw data and fitted distribution on density, CDF, Q-Q plot and P-P plot

Usage

fitPlot(object, ...) 

## S4 method for signature 'FitDist'
fitPlot(object, n = missing)

Arguments

object FitDist Object
... Additional function arguments
n Number of samples, should not be used in current setting
getCopula

Examples

library(cascsim)
data(claimdata)

# frequency fitting example
findex <- new("Index", startDate = as.Date("2012-01-01"), tabulate=TRUE, monthlyIndex = c(rep(1,11),
cumprod(c(1,rep(1.5*(1/12),11))),cumprod(c(1.5,rep((1.3/1.5)^(1/12),11))),
cumprod(c(1.3,rep((1.35/1.3)^(1/12),11))),cumprod(c(1.35,rep((1.4/1.35)^(1/12),11))),1.4))
rawdata <- as.data.frame(as.Date(claimdata[(claimdata[,"LoB"]=="Auto" &
claimdata[,"Type"]=="H"),]$occurrenceDate))
colnames(rawdata)<-"occurrenceDate"
xFit <- new("FitDist", observation=rawdata, trend=findex, startDate = as.Date("2012-01-01"),
method="mle", ifreq=TRUE, idate=TRUE, freq="Monthly")
xFit <- setFitdata(xFit)
setTrialDist(xFit) <- new("Poisson")
xFit@soutput
fitPlot(xFit)

getCopula  

Get the R copula object.

Description

Get the R copula object.

Usage

getcopula(object, ...)

## S4 method for signature 'CopulaObj'
getcopula(object)

Arguments

object  
R copula object

...  
Additional parameters that may or may not be used

Examples

library(cascsim)
dist1<-new("Pareto",p1=20,p2=3)
dist2<-new("Normal",p1=5,p2=3,min=0,max=20, truncated=TRUE)
nom.cop <- new("CopulaObj", param=c(0.5), marginal=list(dist1=dist1,dist2=dist2), dimension=2)
getcopula(nom.cop)
getIndex

Retrieve index value based on dates.

Description

getIndex get a time index to reflect inflation, underwriting cycle or seasonality.

Usage

getIndex(object, ...)

## S4 method for signature 'Index'
getIndex(object, dates)

Arguments

object Index Object
... Additional function arguments
dates dates to get index information

Examples

xindex <- new("Index", indexID = "IDX1", tabulate = FALSE, annualizedRate = 0.03)
xindex<-setIndex(xindex)
xindex@monthlyIndex
dates<-as.Date("2015-12-31")
getIndex(xindex,dates)

getObservation

Get input data from an object.

Description

Get input data from an object.

Usage

getObservation(object, ...)

## S4 method for signature 'FitDist'
getObservation(object)

Arguments

object Object
... Additional function arguments
getTrend

Examples

library(cascsim)
data(claimdata)

#frequency fitting example
findex <- new("Index", startDate = as.Date("2012-01-01"), tabulate=TRUE, monthlyIndex = c(rep(1,11),
cumprod(c(1,rep(1.5^(1/12),11))),cumprod(c(1.5,rep((1.3/1.5)^(1/12),11))),
cumprod(c(1.3,rep((1.35/1.3)^(1/12),11))),cumprod(c(1.35,rep((1.4/1.35)^(1/12),11))),1.4))
rawdata <- as.data.frame(as.Date(claimdata[(claimdata[,"LoB"]=="Auto" &
claimdata[,"Type"]=="H"),]occurrenceDate))
colnames(rawdata)<="occurrenceDate"
xFit <- new("FitDist", observation=rawdata, trend=findex,startDate = as.Date("2012-01-01"),
method="mle",ifreq=TRUE,idate=TRUE, freq="Monthly")
getObservation(xFit)

gterrned

Get the trend index.

Description

Get the trend index.

Usage

getTrend(object, ...)

# S4 method for signature 'FitDist'
getTrend(object)

Arguments

object Object

... Additional function arguments

Examples

library(cascsim)
data(claimdata)

#frequency fitting example
findex <- new("Index", startDate = as.Date("2012-01-01"), tabulate=TRUE, monthlyIndex = c(rep(1,11),
cumprod(c(1,rep(1.5^(1/12),11))),cumprod(c(1.5,rep((1.3/1.5)^(1/12),11))),
cumprod(c(1.3,rep((1.35/1.3)^(1/12),11))),cumprod(c(1.35,rep((1.4/1.35)^(1/12),11))),1.4))
rawdata <- as.data.frame(as.Date(claimdata[(claimdata[,"LoB"]=="Auto" &
claimdata[,"Type"]=="H"),]occurrenceDate))
colnames(rawdata)<="occurrenceDate"
xFit <- new("FitDist", observation=rawdata, trend=findex,startDate = as.Date("2012-01-01"),
method="mle",ifreq=TRUE,idate=TRUE, freq="Monthly")
getTrend(xFit)
Index-class

An S4 class to represent a time index for frequency or severity distribution.

Description

An S4 class to represent a time index for frequency or severity distribution.

Slots

- `indexID` A string to identify the index.
- `startDate` The date the index starts. It is expected to be consistent with the start date of the claim analysis.
- `tabulate` A boolean to indicate whether the index is determined by a constant rate (FALSE) or a series of index values (TRUE).
- `annualizedRate` A yearly index growth rate. It is only used when `tabulate == FALSE`.
- `yearlyIndex` A vector that contains index value on a yearly basis.
- `monthlyIndex` A vector that contains index value on a monthly basis.
- `seasonality` A vector that contains seasonal adjustment factor on a monthly basis.

KSTest

K-S Test

Description

K-S Test

Usage

KSTest(object, ...)  

## S4 method for signature 'FitDist'  
KSTest(object, n = missing)

Arguments

- `object` FitDist Object
- `...` Additional function arguments
- `n` Number of samples, should not be used in current setting
mpareto

Moment function of Pareto Distribution (PDF: \( \alpha \times m^\alpha / x^{(\alpha+1)} \))

Description

Moment function of Pareto Distribution (PDF: \( \alpha \times m^\alpha / x^{(\alpha+1)} \))
Density function of Pareto Distribution (PDF: \( \alpha \times m^\alpha / x^{(\alpha+1)} \))
Cumulative probability function of Pareto Distribution (CDF: \( 1 - (m/x)^\alpha \))
Quantile function of Pareto Distribution
Random generation of Pareto Distribution

Usage

mpareto(order, xm, alpha = 3)
dpareto(x, xm, alpha = 3)
ppareto(q, xm, alpha = 3)
qpareto(p, xm, alpha = 3)
rpareto(n, xm, alpha = 3)

Arguments

order Order of moment
xm Threshold value
alpha Default=3
x Value of the variable
q Value of the variable
p Value of the probability
n Number of samples

Examples

mpareto(1,1000,2)
dpareto(1500,1000,2)
ppareto(1500,1000,2)
qpareto(0.5,1000,2)
rpareto(100,1000,2)
nloglik  

*Description*

Negative Loglikelihood.

*Usage*

\[\text{nloglik}(\text{paras, dist, fitdata, deductible, limit})\]

*Arguments*

- **paras**: A vector contain distribution parameters.
- **dist**: A Distribution Object.
- **fitdata**: A vector of loss data for fitting.
- **deductible**: A vector of deductible data for all loss data.
- **limit**: A vector of limit data for all loss data.

*Examples*

```r
paras<-c(1,1)
dist<-new("Normal")
fitdata<-rtnorm(1000,3,2,1,10)
deductible<-rep(1,1000)
limit<-rep(9,1000)
nloglik(paras,dist,fitdata,deductible,limit)
paras<-c(3,2)
nloglik(paras,dist,fitdata,deductible,limit)
```

---

observationPlot  

*Description*

Plotting the data for distribution fitting

*Usage*

\[\text{observationPlot}(\text{object, ...})\]

# S4 method for signature 'FitDist'
observationPlot(object)
PDFPlot

Arguments

object FitDist Object
...
Additional function arguments

Examples

library(cascsim)
data(claimdata)

#frequency fitting example
findex <- new("Index", startDate = as.Date("2012-01-01"), tabulate=TRUE, monthlyIndex = c(rep(1,11),
cumprod(c(1,rep(1.5*(1/12),11))),cumprod(c(1.5,rep((1/12),11))),
cumprod(c(1.3,rep((1.35/1.3)^(1/12),11))),cumprod(c(1.35,rep((1.4/1.35)^(1/12),11))),1.4))
rawdata <- as.data.frame(as.Date(claimdata[(claimdata[,"LoB"]=="Auto" & claimdata[,"Type"]=="H"),]$occurrenceDate))
colnames(rawdata)<-"occurrenceDate"
xFit <- new("FitDist", observation=rawdata, trend=findex, startDate = as.Date("2012-01-01"),
method="mle",ifreq=TRUE,idate=TRUE, freq="Monthly")
xFit <- setFitdata(xFit)
setTrialDist(xFit) <- new("Poisson")
xFit@soutput
observationPlot(xFit)

Description

Plotting the PDF of data and fitted distribution

Usage

PDFPlot(object, ...)

## S4 method for signature 'FitDist'
PDFPlot(object, n = missing)

Arguments

object FitDist Object
...
Additional function arguments

n Number of samples, should not be used in current setting
Examples

```r
library(cascsim)
data(claimdata)

# frequency fitting example
findex <- new("Index", startDate = as.Date("2012-01-01"), tabulate=TRUE, monthlyIndex = c(rep(1,11),
cumprod(c(1,rep(1.5^((1/12)),11))),cumprod(c(1.5,rep((1.3/1.5)^((1/12)),11))),
cumprod(c(1.3,rep((1.35/1.3)^((1/12)),11))),cumprod(c(1.35,rep((1.4/1.35)^((1/12)),11))),1.4))
rawdata <- as.data.frame(as.Date(claimdata[(claimdata[,"LoB"]=="Auto" & claimdata[,"Type"]=="H"),]$occurrenceDate))
colnames(rawdata)<-"occurrenceDate"
xFit <- new("FitDist", observation=rawdata, trend=findex, startDate = as.Date("2012-01-01"),
method="mle",ifreq=TRUE,idate=TRUE, freq="Monthly")
xFit <- setFitdata(xFit)
setTrialDist(xFit) <- new("Poisson")
xFit@soutput
PDFPlot(xFit)
```

---

**pempirical**

*Cumulative probability function of empirical distribution using linear interpolation*

**Description**

Cumulative probability function of empirical distribution using linear interpolation
Quantile function of Empirical Distribution
Random generation function of Empirical Distribution
Density function of Empirical Distribution based on simulation

**Usage**

- `pempirical(q, cdf)`
- `qempirical(p, cdf)`
- `rempirical(n, cdf)`
- `dempirical(x, cdf)`

**Arguments**

- `q` Value of the variable
- `cdf` empirical distribution (cdf for continuous distribution and pmf for discrete distribution)
- `p` Value of the probability
- `n` Number of samples
- `x` Value of the variable
Examples

# discrete distribution
pempirical(c(3,5,10),matrix(c(0.1,0.2,0.3,0.05,0.05,0.2,0.1,1:6,10),7,2))
# continuous distribution
pempirical(350,matrix(c(seq(0.01,1,0.01),cumprod(c(1,rep(1.1,99)))),100,2))
# discrete distribution
qempirical(c(0.3,0.65,1),matrix(c(0.1,0.2,0.3,0.05,0.05,0.2,0.1,1:6,10),7,2))
# continuous distribution
qempirical(c(0.3,0.65,0.8),matrix(c(seq(0.01,1,0.01),cumprod(c(1,rep(1.1,99))))),100,2)
# discrete distribution
rempirical(100,matrix(c(0.1,0.2,0.3,0.05,0.05,0.2,0.1,1:6,10),7,2))
# continuous distribution
rempirical(100,matrix(c(seq(0.01,1,0.01),cumprod(c(1,rep(1.1,99))))),100,2)
# discrete distribution
dempirical(3,matrix(c(0.1,0.2,0.3,0.05,0.05,0.2,0.1,1:6,10),7,2))
# continuous distribution
dempirical(30,matrix(c(seq(0.01,1,0.01),qnorm(seq(0.01,1,0.01),30,20))),100,2))

plotText

Plot text content

Description

Plot text content

Usage

plotText(content)

Arguments

content A string to plot

Examples

plotText("You are awesome!")

PPPlot

P-P Plot of data and fitted distribution

Description

P-P Plot of data and fitted distribution
Probability

Usage

PPPlot(object, ...)

## S4 method for signature 'FitDist'
PPPlot(object, n = missing)

Arguments

object FitDist Object
...

Additional function arguments

n Number of samples, should not be used in current setting

Examples

library(cascsim)
data(claimdata)

# frequency fitting example
findex <- new("Index", startDate = as.Date("2012-01-01"), tabulate=TRUE, monthlyIndex = c(rep(1,11),
cumprod(c(1,rep(1.5^(1/12),11))),cumprod(c(1.5,rep((1.3/1.5)^(1/12),11))),
cumprod(c(1.3,rep((1.35/1.3)^(1/12),11))),cumprod(c(1.35,rep((1.4/1.35)^(1/12),11))),1.4))
rawdata <- as.data.frame(as.Date(claimdata[(claimdata[,"LoB"]=="Auto" & claimdata[,"Type"]=="H"),]$occurrenceDate))
colnames(rawdata)<-"occurrenceDate"
xFit <- new("FitDist", observation=rawdata, trend=findex, startDate = as.Date("2012-01-01"),
method="mle",ifreq=TRUE,idate=TRUE, freq="Monthly")
xFit <- setFitdata(xFit)
setTrialDist(xFit) <- new("Poisson")
xFit@soutput
observationPlot(xFit)
PPPlot(xFit)

Probability

Probability function.

Description

Probability function.

Usage

Probability(object, q, ...)

## S4 method for signature 'Normal'
Probability(object, q)

## S4 method for signature 'Beta'
Probability(object, q)
QQPlot

## S4 method for signature 'Exponential'
Probability(object, q)

## S4 method for signature 'Gamma'
Probability(object, q)

## S4 method for signature 'Geometric'
Probability(object, q)

## S4 method for signature 'Lognormal'
Probability(object, q)

## S4 method for signature 'NegativeBinomial'
Probability(object, q)

## S4 method for signature 'Pareto'
Probability(object, q)

## S4 method for signature 'Poisson'
Probability(object, q)

## S4 method for signature 'Uniform'
Probability(object, q)

## S4 method for signature 'Weibull'
Probability(object, q)

## S4 method for signature 'Empirical'
Probability(object, q)

### Arguments

- **object**: Distribution Object
- **q**: Variable value
- **...**: Additional function arguments

### Examples

```r
xPareto <- new("Pareto", p1=20, p2=3)
Probability(xPareto, 50)
```

---

**QQPlot**

*Q-Q Plot of data and fitted distribution*

**Description**

Q-Q Plot of data and fitted distribution
Quantile

Usage

QQPlot(object, ...)

## S4 method for signature 'FitDist'
QQPlot(object, n = missing)

Arguments

object FitDist Object
...
Additional function arguments
n Number of samples, should not be used in current setting

Examples

library(cascsim)
data(claimdata)
#frequency fitting example
findex <- new("Index", startDate = as.Date("2012-01-01"),
     tabulate=TRUE, monthlyIndex = c(rep(1,11),
     cumprod(c(1,rep(1.5*(1/12),11))),cumprod(c(1.5,rep((1.3/1.5)^(1/12),11)),
     cumprod(c(1.3,rep((1.35/1.3)^(1/12),11))),cumprod(c(1.35,rep((1.4/1.35)^(1/12),11))),1.4))
rawdata <- as.data.frame(as.Date(claimdata[(claimdata[,"LoB"]=='Auto' &
     claimdata[,"Type"]=='H"),]$occurrenceDate))
colnames(rawdata)<-"occurrenceDate"
xFit <- new("FitDist", observation=rawdata, trend=findex,startDate = as.Date("2012-01-01"),
     method="mle",ifreq=TRUE,idate=TRUE, freq="Monthly")
xFit <- setFitdata(xFit)
setTrialDist(xFit) <- new("Poisson")
xFit@soutput
QQPlot(xFit)

Quantile

Quantile function.

Description

Quantile function.

Usage

Quantile(object, p, ...)

## S4 method for signature 'Normal'
Quantile(object, p)

## S4 method for signature 'Beta'
Quantile(object, p)
## S4 method for signature 'Exponential'
Quantile(object, p)

## S4 method for signature 'Gamma'
Quantile(object, p)

## S4 method for signature 'Geometric'
Quantile(object, p)

## S4 method for signature 'Lognormal'
Quantile(object, p)

## S4 method for signature 'NegativeBinomial'
Quantile(object, p)

## S4 method for signature 'Pareto'
Quantile(object, p)

## S4 method for signature 'Poisson'
Quantile(object, p)

## S4 method for signature 'Uniform'
Quantile(object, p)

## S4 method for signature 'Weibull'
Quantile(object, p)

## S4 method for signature 'Empirical'
Quantile(object, p)

### Arguments

- object: Distribution Object
- p: Probability
- ...: Additional function arguments

### Examples

```r
xPareto <- new("Pareto", p1=20, p2=3)
Quantile(xPareto, 0.6)
```

---

**rreopen**  
*Simulate whether closed claims will be reopened or not.*

### Description

Simulate whether closed claims will be reopened or not.
Usage

\texttt{rreopen(closeYear, reopenProb)}

Arguments

closeYear Years after claim closure. It could be a number or a numeric vector.
reopenProb A vector that contains the reopen probability based on closeYear.

Examples

\begin{verbatim}
reopenprob<-c(0.02,0.01,0.005,0.005,0.003,0)
  rreopen(rep(2,1000),reopenprob)
\end{verbatim}

-------------------

\textbf{sampleKurtosis} \hspace{1cm} Calculate the excess kurtosis of 10000 sampled values from the distribution.

-------------------

Description

Calculate the excess kurtosis of 10000 sampled values from the distribution.

Usage

\texttt{sampleKurtosis(object, ...)}

## S4 method for signature 'Distribution'
sampleKurtosis(object)

Arguments

object A Distribution Object
... Additional function arguments

Examples

\begin{verbatim}
xLognormal <- new("Lognormal",p1=2,p2=3)
sampleKurtosis(xLognormal)
\end{verbatim}
sampleMean

Calculate the mean of 100000 sampled values from the distribution.

Description

Calculate the mean of 100000 sampled values from the distribution.

Usage

```r
calculateMean(x)
```

## S4 method for signature 'Distribution'
calculateMean(x)

Arguments

- `object`: A Distribution Object
- `...`: Additional function arguments

Examples

```r
x <- new("Lognormal", p1=2, p2=3)
calculateMean(x)
```

sampleSd

Calculate the standard deviation of 10000 sampled values from the distribution.

Description

Calculate the standard deviation of 10000 sampled values from the distribution.

Usage

```r
calculateSd(x)
```

## S4 method for signature 'Distribution'
calculateSd(x)

Arguments

- `object`: A Distribution Object
- `...`: Additional function arguments

Examples

```r
x <- new("Lognormal", p1=2, p2=3)
calculateSd(x)
```
sampleSkew

Calculate the skewness of 10000 sampled values from the distribution.

Description

Calculate the skewness of 10000 sampled values from the distribution.

Usage

sampleSkew(object, ...)

## S4 method for signature 'Distribution'
sampleSkew(object)

Arguments

object A Distribution Object
...
Additional function arguments

Examples

xLognormal <- new("Lognormal",p1=2,p2=3)
sampleSkew(xLognormal)

setAnnualizedRate<- Set the annualized level rate to construct the index. Only used when tabulate == FALSE.

Description

Set the annualized level rate to construct the index. Only used when tabulate == FALSE.

Usage

setAnnualizedRate(this, ...) <- value

## S4 replacement method for signature 'Index,numeric'
setAnnualizedRate(this) <- value

Arguments

this Index Object
...
Additional function arguments
value Numeric Value (default:0.02)
**Examples**

```r
xindex <- new("Index")
setID(xindex)<-"IDX1"
setTabulate(xindex)<-FALSE
setAnnualizedRate(xindex)<-0.03
xindex<-setIndex(xindex)
xindex@monthlyIndex
```

```r
copula <- new("CopulaObj", param=c(0.5),marginal=list(dist1=dist1,dist2=dist2),dimension=2)
setCopulaParam(copula) <- 0.6
```

---

**setCopulaParam**

`setCopulaParam<-  Set copula parameters.`

**Description**

Set copula parameters.

**Usage**

```r
setCopulaParam(this, ...) <- value
```

```r
## S4 replacement method for signature 'CopulaObj,numeric'
setCopulaParam(this) <- value
```

**Arguments**

- `this` Copula Object
- `...` Additional function arguments
- `value` The copula parameters

**Examples**

```r
library(cascsim)
dist1<-new("Pareto",p1=20,p2=3)
dist2<-new("Normal",p1=5,p2=3,min=0,max=20,truncated=TRUE)
cop <- new("CopulaObj", param=c(0.5),marginal=list(dist1=dist1,dist2=dist2),dimension=2)
setCopulaParam(cop) <- 0.6
```
setCopulaType<-  

Description

Set copula type.

Usage

setCopulaType(this, ...) <- value

## S4 replacement method for signature 'CopulaObj,character'
setCopulaType(this) <- value

Arguments

this  Copula Object
...
value  The copula type

Examples

library(cascsim)
dist1<-'Pareto',p1=20,p2=3)
dist2<-'Normal',p1=5,p2=3,min=0,max=20,truncated=TRUE)
cop <- new("CopulaObj", param=c(0.5),marginal=list(dist1=dist1,dist2=dist2),dimension=2)
setCopulaType(cop) <- "joe"

setDevFac

Description

Set up an IBNER loss development schedule.

Usage

setDevFac(object, ...)

## S4 method for signature 'DevFac'
setDevFac(object)
Arguments

object DevFac Object

... Additional function arguments

Examples

xIBNERFactor <- new("DevFac", FacID = "IF1", FacModel = FALSE, meanList = c(1.26,1.1,1.05,1.02,1),
volList = rep(0.02,5))
xIBNERFactor<-setDevFac(xIBNERFactor)
xIBNERFactor

xIBNERFactor <- new("DevFac")
setID(xIBNERFactor)<-"IF1"
setFacModel(xIBNERFactor)<-TRUE
setFun(xIBNERFactor)<-"identity"
setXname(xIBNERFactor)<- c("x1","x2","x3")
setParas(xIBNERFactor)<-c(0.6,-0.2,0.01,-0.3,0.02,0.03,0.01,0.02)
xIBNERFactor<-setDevFac(xIBNERFactor)
xIBNERFactor

setDf<- Set the degree of freedom for t Copula.

Description

Set the degree of freedom for t Copula.

Usage

setDf(this, ...) <- value

## S4 replacement method for signature 'CopulaObj,numeric'
setDf(this) <- value

Arguments

this Copula Object

... Additional function arguments

value The degree of freedom. The default value is 3.

Examples

library(cascsim)
dist1<-new("Pareto",p1=20,p2=3)
dist2<-new("Normal",p1=5,p2=3,min=0,max=20, truncated=TRUE)
cop <- new("CopulaObj", type="t", param=c(0.5),marginal=list(dist1=dist1,dist2=dist2),dimension=2)
setDf(cop) <- 5
setDimension<-

Set the dimension of the copula.

Description

Set the dimension of the copula.

Usage

setDimension(this, ...) <- value

## S4 replacement method for signature 'CopulaObj,numeric'
setDimension(this) <- value

Arguments

this Copula Object
...

value The dimension of the copula. It can also be set by providing marginal distributions

Examples

library(cascsim)
dist1 <- new("Pareto",p1=20,p2=3)
dist2 <- new("Normal",p1=5,p2=3,min=0,max=20,truncate=TRUE)
cop <- new("CopulaObj", param=c(0.5), marginal=list(dist1=dist1,dist2=dist2),dimension=2)
dist3 <- new("Pareto",p1=10,p2=3)
setDimension(cop) <- 3
setMarginal(cop) <- list(dist1=dist1,dist2=dist2,dist3=dist3)

setDispstr<- Set parameter matrix format of Elliptical copula.

Description

Set parameter matrix format of Elliptical copula.

Usage

setDispstr(this, ...) <- value

## S4 replacement method for signature 'CopulaObj,character'
setDispstr(this) <- value
setEmpirical<-  

Arguments  

this Copula Object  
... Additional function arguments  
value  

The matrix format. The default is "un" for unstructured. Other choices include "ex" for exchangeable, "ar1" for AR(1), and "toep" for Toeplitz (toeplitz).

Examples  

library(cascsim)  

dist1<-'Pareto',p1=20,p2=3)'  
dist2<-'Normal',p1=5,p2=3,min=0,max=20,truncated=TRUE)'  
cop <- 'CopulaObj', param=c(0.5),marginal=list(dist1=dist1,dist2=dist2),dimension=2)'  
setDispstr(cop) <- 'ex'  

Description  

Set the list of values and corresponding probabilities (Pr(X<value) for continuous variable and Pr(X==value) for discrete variable). It is only used for empirical distribution.

Usage  

setEmpirical(this, ...) <- value  

## S4 replacement method for signature 'Distribution,matrix'  
setEmpirical(this) <- value  

Arguments  

this Distribution Object  
... Additional function arguments.  
value Two-column matrix with values and probabilities dist <- new("Normal") setEmpirical(dist) <- matrix(c(0.01,0.25,0.5,0.75,0.99, 0.01,0.25,0.5,0.75,0.99, 0.01,0.25,0.5,0.75,0.99, 0.01,0.25,0.5,0.75,0.99, 0.01,0.25,0.5,0.75,0.99), nrow = 5, ncol = 2) dist
setFacModel <- \textit{Determine whether the development factor is determined by a predictive model or a fixed schedule by development year}\texttt{\textbackslash}\texttt{Description}\texttt{\textbackslash}\texttt{Usage}\texttt{\textbackslash}\texttt{Arguments}\texttt{\textbackslash}\texttt{Examples}\texttt{\textbackslash\texttt{setFitdata \textit{Preparing the input data (observation) for distribution fitting, including detrending, translating occurrence dates to frequency data, etc.}}}
setfitmethod <-
setfitmethod <-

Usage
setFitdata(object, ...)

## S4 method for signature 'FitDist'
setFitdata(object)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>FitDist Object</td>
</tr>
<tr>
<td>...</td>
<td>Additional function arguments</td>
</tr>
</tbody>
</table>

Examples

library(cascsim)
data(claimdata)

# frequency fitting example
findex <- new("Index", startDate = as.Date("2012-01-01"), tabulate=TRUE, monthlyIndex = c(rep(1,11), cumprod(c(1,rep(1.5^(1/12),11))),cumprod(c(1.5,rep((1.3/1.5)^(1/12),11)))), cumprod(c(1.3,rep((1.35/1.3)^(1/12),11)))),cumprod(c(1.35,rep((1.4/1.35)^(1/12),11)))),1.4))
rawdata <- as.data.frame(as.Date(claimdata[(claimdata[,"LoB"]=="Auto" & claimdata,"Type"]=="H"),]$occurrenceDate))
colnames(rawdata)<-"occurrenceDate"
xFit <- new("FitDist", observation=rawdata, trend=findex,startDate = as.Date("2012-01-01"), method="mle",ifreq=TRUE,idate=TRUE, freq="Monthly")
xFit <- setFitdata(xFit)
xFit@fitdata

setfitmethod <-

Set distribution fitting method.

Description
Set distribution fitting method.

Usage
setfitmethod(this, ...) <- value

## S4 replacement method for signature 'FitDist,character'
setfitmethod(this) <- value

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>this</td>
<td>FitDist Object</td>
</tr>
<tr>
<td>...</td>
<td>Additional function arguments</td>
</tr>
<tr>
<td>value</td>
<td>A character string: &quot;mle&quot;, &quot;mme&quot;, or &quot;qme&quot;</td>
</tr>
</tbody>
</table>
Examples

```r
library(cascsim)
data(claimdata)

# frequency fitting example
findex <- new("Index", startDate = as.Date("2012-01-01"), tabulate=TRUE, monthlyIndex = c(rep(1,11),
cumprod(c(1,rep(1.5^(1/12),11))),
cumprod(c(1.5,rep((1.3/1.5)^(1/12),11))),
cumprod(c(1.3,rep((1.35/1.3)^(1/12),11))),1.4))
rawdata <- as.data.frame(as.Date(claimdata[(claimdata[,"LoB"]=="Auto" & claimdata[,"Type"]=="H"),]$occurrenceDate))
colnames(rawdata)<-"occurrenceDate"
xFit <- new("FitDist", observation=rawdata, trend=findex, startDate = as.Date("2012-01-01"),
method="mle",ifreq=TRUE,idate=TRUE, freq="Monthly")
setfitmethod(xFit) <- "mme"
xFit@method
```

---

**Description**

Directly set the fitted distribution without fitting it to the data.

**Usage**

```r
setFittedDist(this) <- value
```

## S4 replacement method for signature 'FitDist,Distribution'

```r
setFittedDist(this) <- value
```

**Arguments**

- **this**: FitDist Object
- **value**: Fitted distribution

**Examples**

```r
library(cascsim)
data(claimdata)

# frequency fitting example
findex <- new("Index", startDate = as.Date("2012-01-01"), tabulate=TRUE, monthlyIndex = c(rep(1,11),
cumprod(c(1,rep(1.5^(1/12),11))),
cumprod(c(1.5,rep((1.3/1.5)^(1/12),11))),
cumprod(c(1.3,rep((1.35/1.3)^(1/12),11))),1.4))
rawdata <- as.data.frame(as.Date(claimdata[(claimdata[,"LoB"]=="Auto" & claimdata[,"Type"]=="H"),]$occurrenceDate))
colnames(rawdata)<-"occurrenceDate"
xFit <- new("FitDist", observation=rawdata, trend=findex, startDate = as.Date("2012-01-01"),
method="mle",ifreq=TRUE,idate=TRUE, freq="Monthly")
```

setfreq<- 

xFit <- setFitdata(xFit) 
setTrialDist(xFit) <- new("Poisson") 
xFit@fitted

setfreq<-  

Set the data frequency.

Description

Set the data frequency.

Usage

setfreq(this, ...) <- value

## S4 replacement method for signature 'FitDist,character'
setfreq(this) <- value

Arguments

this  FitDist Object

...  Additional function arguments

value  A character string: "Annual" or "Monthly"

Examples

library(cascsim)
data(claimdata)

#frequency fitting example
findex <- new("Index", startDate = as.Date("2012-01-01"), tabulate=TRUE, monthlyIndex = c(rep(1,11), cumprod(c(1,rep(1.5*(1/12),11))),cumprod(c(1.5,rep((1.3/1.5)^((1/12),11)))),
cumprod(c(1.3,rep((1.35/1.3)^((1/12),11)))),cumprod(c(1.35,rep((1.4/1.35)^((1/12),11)))),1.4))
rawdata <- as.data.frame(as.Date(claimdata[(claimdata[,"LoB"]=='Auto' & claimdata[,"Type"]=='H"),]occurrenceDate))
colnames(rawdata)<-"occurrenceDate"
xFit <- new("FitDist", observation=rawdata, trend=findex,startDate = as.Date("2012-01-01"),
method="mle",ifreq=TRUE,idate=TRUE, freq="Annual")
setfreq(xFit) <- "Monthly"
xFit@freq
**setFun**

*Set the model format/link function (identity/inverse/log/exponential). Only used when FacModel == TRUE.*

**Description**

Set the model format/link function (identity/inverse/log/exponential). Only used when FacModel == TRUE.

**Usage**

```r
setFun(this, ...) <- value
```

```r
## S4 replacement method for signature 'DevFac,character'
setFun(this) <- value
```

**Arguments**

- **this** DevFac Object
- **...** Additional function arguments
- **value** String Value (default: "identity")

**Examples**

```r
xIBNERFactor <- new("DevFac")
setID(xIBNERFactor)<-"IF1"
setFacModel(xIBNERFactor)<-TRUE
setFun(xIBNERFactor)<-"identity"
setXname(xIBNERFactor)<- c("x1","x2","x3")
setParas(xIBNERFactor)<-c(0.6,-0.2,0.01,-0.3,0.02,0.03,0.01,0.02)
xIBNERFactor<-setDevFac(xIBNERFactor)
xIBNERFactor
```

**setID**

*setID Set the ID for an object*

**Description**

setID Set the ID for an object
setidate<-  

Usage

setID(this, ...) <- value

## S4 replacement method for signature 'Index,character'
setID(this) <- value

## S4 replacement method for signature 'DevFac,character'
setID(this) <- value

Arguments

this Self
...
value ID

Examples

xindex <- new("Index")
setID(xindex)<-"IDX1"
xindex@indexID

setidate<- Set whether occurrence dates will be used for frequency data.

Description

Set whether occurrence dates will be used for frequency data.

Usage

setidate(this, ...) <- value

## S4 replacement method for signature 'FitDist,logical'
setidate(this) <- value

Arguments

this FitDist Object
...
value A boolean
Examples

```r
library(cascsim)
data(claimdata)

# frequency fitting example
findex <- new("Index", startDate = as.Date("2012-01-01"), tabulate=TRUE, monthlyIndex = c(rep(1,11),
cumprod(c(1,rep(1.5^(1/12),11))),(1.5,rep((1.35/1.3)^(1/12),11))))
rawdata <- as.data.frame(as.Date(claimdata[claimdata[,"LoB"]=='Auto' &
claimdata[,"Type"]=='H'],1)$occurrenceDate))
colnames(rawdata)<-'occurrenceDate"'
xFit <- new("FitDist", observation=rawdata, trend=findex, startDate = as.Date("2012-01-01"),
method="mle",ifreq=TRUE,odate=FALSE, freq="Monthly")
setidate(xFit) <- TRUE
xFit@idate
```

```
setifreq<- Set the data type: frequency or severity/time lag.

Description

Set the data type: frequency or severity/time lag.

Usage

setifreq(this, ...) <- value

## S4 replacement method for signature 'FitDist,logical'
setifreq(this) <- value

Arguments

this FitDist Object

... Additional function arguments

value A boolean

Examples

```r
library(cascsim)
data(claimdata)

# frequency fitting example
findex <- new("Index", startDate = as.Date("2012-01-01"), tabulate=TRUE, monthlyIndex = c(rep(1,11),
cumprod(c(1,rep(1.5^(1/12),11))),(1.5,rep((1.35/1.3)^(1/12),11))))
rawdata <- as.data.frame(as.Date(claimdata[claimdata[,"LoB"]=='Auto' &
claimdata[,"Type"]=='H'],1)$occurrenceDate))
colnames(rawdata)<-'occurrenceDate"
```
xFit <- new("FitDist", observation=rawdata, trend=findex, startDate = as.Date("2012-01-01"), method="mle", ifreq=TRUE, idate=TRUE, freq="Monthly")
setifreq(xFit) <- FALSE
xFit@ifreq

setIndex

Set up a time index for frequency or severity.

Description

setIndex sets a time index to reflect inflation, underwriting cycle or seasonality.

Usage

setIndex(object, ...)

## S4 method for signature 'Index'
setIndex(object)

Arguments

object         Index Object
...            Additional function arguments

Examples

xindex <- new("Index", indexID = "IDX1", tabulate = FALSE, annualizedRate = 0.03)
xindex<-setIndex(xindex)
xindex@monthlyIndex

xindex <- new("Index")
setID(xindex)<-"IDX1"
setTabulate(xindex)<-TRUE
setAnnualizedRate(xindex)<-0.03
setYearlyIndex(xindex)<- c(1,1.05,1.2,0.95,1.3)
set.seed(123)
setSeasonality(xindex)<-rnorm(12,mean=1,sd=0.03)
xindex<-setIndex(xindex)
xindex@monthlyIndex
setMarginal<-  
Set the marginal distributions of the copula.

Description
Set the marginal distributions of the copula.

Usage
setMarginal(this, ...) <- value

## S4 replacement method for signature 'CopulaObj,list'
setMarginal(this) <- value

Arguments
this  Copula Object
...  Additional function arguments
value  The list of marginal distributions.

Examples
library(cascsim)
dist1<-new("Pareto",p1=20,p2=3)
dist2<-new("Normal",p1=5,p2=3,min=0,max=20,truncated=TRUE)
cop <- new("CopulaObj", param=c(0.5),marginal=list(dist1=dist1,dist2=dist2),dimension=2)
dist3<-new("Pareto",p1=10,p2=3)
dist4<-new("Normal",p1=2,p2=3,min=0,max=20,truncated=TRUE)
setMarginal(cop) <- list(dist1=dist3,dist2=dist4)

setMeanList<-  
Set the year-to-year loss development factor.

Description
setMeanList<- sets expected year-to-year loss development factor. Years after It is only used when ibnerfModel == FALSE.

Usage
setMeanList(this, ...) <- value

## S4 replacement method for signature 'DevFac,vector'
setMeanList(this) <- value
Arguments

   this         DevFac Object
...          Additional function arguments
   value       Numeric Vector

Examples

xIBNERFactor <- new("DevFac")
setId(xIBNERFactor)<-"IF1"
setFacModel(xIBNERFactor)<-FALSE
setMeanList(xIBNERFactor)<-c(1.26,1.1,1.05,1.02,1)
setVolList(xIBNERFactor)<-rep(0.02,5)
xIBNERFactor

setMin(object, ...)  

## S4 method for signature 'Distribution'
setMin(object, minval)

Arguments

   object       A Distribution Object
...          Additional function arguments.
   minval      The minimum value.

Examples

xLognormal <- new("Lognormal",p1=2,p2=3)
xLognormal <- setMin(xLognormal,50)

setMin
Set the minimum of the distribution. For example, the distribution of settlement lag for open claims

Description

Set the minimum of the distribution. For example, the distribution of settlement lag for open claims

Usage

setMin(object, ...)
setMonthlyIndex<-  *Set monthly index values.*

**Description**

`setMonthlyIndex<-` sets monthly index values.

**Usage**

```r
setMonthlyIndex(this, ...) <- value
```

## S4 replacement method for signature 'Index,vector'

```r
setMonthlyIndex(this) <- value
```

**Arguments**

- `this`  
  Index Object
- `...`  
  Additional function arguments
- `value`  
  Numeric Vector

**Examples**

```r
xindex <- new("Index")
setID(xindex)<-"IDX1"
setTabulate(xindex)<-TRUE
setMonthlyIndex(xindex)<- rep(1,360)
xindex<-setIndex(xindex)
xindex@monthlyIndex
```

---

setObservation<-  *Input the raw data.*

**Description**

Input the raw data.

**Usage**

```r
setObservation(this) <- value
```

## S4 replacement method for signature 'CopulaObj,matrix'

```r
setObservation(this) <- value
```

## S4 replacement method for signature 'FitDist,matrix'

```r
setObservation(this) <- value
```
Arguments

this  
FitDist Object or Copula Object

value  
A data frame or a matrix. For FitDist object, it could be a two-column data frame with the occurrence date and loss size/number of occurrence. Or a one-column data frame with loss size (ifreq == FALSE) or number of occurrence (ifreq == TRUE && idate == FALSE) or occurrence dates (ifreq == TRUE && idate == TRUE). For Copula object, it could be a matrix with each column contains the experience data of a variable.

Examples

library(cascsim)
dist1 <- new("Pareto", p1 = 20, p2 = 3)
dist2 <- new("Normal", p1 = 5, p2 = 3, min = 0, max = 20, truncated = TRUE)
nom.cop <- new("CopulaObj", param = c(0.5), marginal = list(dist1 = dist1, dist2 = dist2), dimension = 2)
setObservation(nom.cop) <- copulaSample(nom.cop, 100)
nom.cop@observation

dist <- new("Normal")
setParams(dist) <- c(2, 3)
dist

setParams<-  
Set distribution parameters.

Description

Set distribution parameters.

Usage

setParams(this, ...) <- value

## S4 replacement method for signature 'Distribution,numeric'
setParams(this) <- value

Arguments

this  
Distribution Object

...  
Additional function arguments.

value  
Numeric vector containing parameters examples dist <- new("Normal") setParams(dist) <- c(2, 3) dist
**Description**

`setParas<-` sets model parameters. Their order must match the order of c("Intercept","DevelopmentYear","IncurredLoss","OSRatio",xname,"Volatility"). "Volatility" stands for the volatility of the error term in the model and used to simulate IBNER development factors. The parameter vector is only used when ibnerfModel == TRUE.

**Usage**

```r
setParas(this, ...) <- value
```

## S4 replacement method for signature 'DevFac,vector'

```r
setParas(this) <- value
```

**Arguments**

- `this` DevFac Object
- `...` Additional function arguments
- `value` Numeric Vector

**Examples**

```r
xIBNERFactor <- new("DevFac")
setID(xIBNERFactor)<-"IF1"
setFacModel(xIBNERFactor)<-TRUE
setFun(xIBNERFactor)<-"identity"
setXname(xIBNERFactor)<- c("x1","x2","x3")
setParas(xIBNERFactor)<-c(0.6,-0.2,0.01,-0.3,0.02,0.03,0.01,0.02)
xIBNERFactor<-setDevFac(xIBNERFactor)
xIBNERFactor
```

---

**setprobs<-**

*Set the percentiles to be matched. Only used when qme is chosen for fitting method.*

**Description**

Set the percentiles to be matched. Only used when qme is chosen for fitting method.

**Usage**

```r
setprobs(this, ...) <- value
```

## S4 replacement method for signature 'FitDist,vector'

```r
setprobs(this) <- value
```
setRange<- 67

Arguments

this  FitDist Object
...  Additional function arguments
value  A numeric vector with values between 0 and 1.

Examples

library(cascsim)
data(claimdata)

# frequency fitting example
findex <- new("Index", startDate = as.Date("2012-01-01"), tabulate=TRUE, monthlyIndex = c(rep(1,11),
cumprod(c(1,1.5*(1/12),11))),cumprod(c(1.5,rep((1.3/1.5)^(1/12),11))),
cumprod(c(1.3,rep((1.35/1.3)^(1/12),11))),cumprod(c(1.35,rep((1.4/1.35)^(1/12),11))),1.4))
rawdata <- as.data.frame(as.Date(claimdata[(claimdata[,"LoB"]=='Auto' &
claimdata[,"Type"]=='H' ),$occurrenceDate))
colnames(rawdata)<-"occurrenceDate"
xFit <- new("FitDist", observation=rawdata, trend=findex, startDate = as.Date("2012-01-01"),
method="mle",ifreq=TRUE, idate=TRUE, freq="Monthly")
setprobs(xFit) <- c(0.1,0.5,0.9)
xFit@probs

setRange<-  Set the min and max of the variable.

Description

Set the min and max of the variable.

Usage

setRange(this, ...) <- value

## S4 replacement method for signature 'Distribution,numeric'
setRange(this) <- value

Arguments

this  Distribution Object
...  Additional function arguments.
value  a two-element vector contains min and max.
setRectangle

*Set up the rectangle based on simulated data.*

**Description**

setRectangle sets up the rectangle based on a data file.

**Usage**

```r
setRectangle(object, data, ...)  
## S4 method for signature 'Triangle, data.frame'
setRectangle(object, data,  
    evaluationDate = as.Date("2016-12-31"),  
    futureDate = as.Date("2017-12-31"), lob = "Total", ctype = "Total")
```

**Arguments**

- `object`: Triangle Object
- `data`: Simulated Data
- `...`: Additional function arguments.
- `evaluationDate`: Evaluation Date;
- `futureDate`: End of projection date;
- `lob`: Line of Business;
- `ctype`: Claim Type.

setSeasonality<-

*Set seasonality on a monthly basis.*

**Description**

setSeasonality<- sets monthly multiplier to reflect seasonal impact.

**Usage**

```r
setSeasonality(this, ...) <- value  
## S4 replacement method for signature 'Index, vector'
setSeasonality(this) <- value
```

**Arguments**

- `this`: Index Object
- `...`: Additional function arguments
- `value`: Numeric Vector (default: `rep(1, 12)`)
Examples

```r
xindex <- new("Index")
setID(xindex)<-"IDX1"
setTabulate(xindex)<-TRUE
setAnnualizedRate(xindex)<-0.03
setYearlyIndex(xindex)<- c(1,1.05,1.2,0.95,1.3)
set.seed(123)
setSeasonality(xindex)<-rnorm(12,mean=1,sd=0.03)
xindex<-setIndex(xindex)
index<-setIndex(xindex)
xindex@monthlyIndex
```

---

```r
setStartDate<-
```

**Set the start date for the claim simulation exercise**

---

**Description**

Set the start date for the claim simulation exercise

**Usage**

```r
setStartDate(this, ...) <- value
```

### S4 replacement method for signature 'Index,Date'

```r
setStartDate(this) <- value
```

**Arguments**

- `this`: Self
- `...`: Additional function arguments
- `value`: Start Date

---

```r
setTabulate<-
```

**Determine whether the index values are constructed from a constant rate or provided directly**

---

**Description**

Determine whether the index values are constructed from a constant rate or provided directly

**Usage**

```r
setTabulate(this, ...) <- value
```

### S4 replacement method for signature 'Index,logical'

```r
setTabulate(this) <- value
```
setTrend<-  

Arguments

this  Index Object
...
value Logical Value (default:FALSE)

Examples

xindex <- new("Index")
setID(xindex)<-"IDX1"
setTabulate(xindex)<-FALSE
setAnnualizedRate(xindex)<-0.03
xindex<-setIndex(xindex)
xindex@monthlyIndex

---

setTrend<-  

Set the trend with an Index Object.

Description

Set the trend with an Index Object.

Usage

setTrend(this, ...) <- value

## S4 replacement method for signature 'FitDist,Index'
setTrend(this) <- value

Arguments

this  FitDist Object
...
value  An Index Object

Examples

library(cascsim)
data(claimdata)

#frequency fitting example
findex <- new("Index", startDate = as.Date("2012-01-01"), tabulate=TRUE, monthlyIndex = c(rep(1,11), cumprod(c(1,rep(1.5^(1/12),11))),cumprod(c(1.5,rep((1.3/1.5)^(1/12),11))),
cumprod(c(1.3,rep((1.35/1.3)^(1/12),11))),cumprod(c(1.35,rep((1.4/1.35)^(1/12),11))),1.4))
rawdata <- as.data.frame(as.Date(claimdata[,"LoB"]=="Auto" & claimdata[,"Type"]="H"),)$occurrenceDate)
colnames(rawdata)<-"occurrenceDate"
xFit <- new("FitDist", observation=rawdata, startDate = as.Date("2012-01-01"),
setTrialDist<- Distribution fitting and testing.

Description

Distribution fitting and testing.

Usage

`setTrialDist(this) <- value`

## S4 replacement method for signature 'FitDist,Distribution'
`setTrialDist(this) <- value`

Arguments

`this` FitDist Object
`value` Distribution to fit to

Examples

library(cascsim)
data(claimdata)

# frequency fitting example
findex <- new("Index", startDate = as.Date("2012-01-01"), tabulate=TRUE, monthlyIndex = c(rep(1,11), cumprod(c(1,rep(1.5^((1/12)),11))),cumprod(c(1.5,rep((1.3/1.5)^((1/12)),11))), cumprod(c(1.3,rep((1.35/1.3)^((1/12)),11))),cumprod(c(1.35,rep((1.4/1.35)^((1/12)),11))),1.4))
rawdata <- as.data.frame(as.Date(claimdata[(claimdata[,"LoB"]=='Auto' & claimdata[,"Type"]=='H'&claimdata[,"occurrenceDate"]"])$occurrenceDate))
colnames(rawdata)<-"occurrenceDate"
xFit <- new("FitDist", observation=rawdata, trend=findex,startDate = as.Date("2012-01-01"), method="mle",ifreq=TRUE,idate=TRUE, freq="Monthly")
xFit <- setFitdata(xFit)
setTrialDist(xFit) <- new("Poisson")
xFit@soutput
observationPlot(xFit)
fitPlot(xFit)
**setTrialDistErr**

*Distribution fitting and testing. Same as setTrialDist except for error tolerance.*

**Description**

Distribution fitting and testing. Same as setTrialDist except for error tolerance.

**Usage**

```r
setTrialDistErr(this) <- value
```

```r
## S4 replacement method for signature 'FitDist,Distribution'
setTrialDistErr(this) <- value
```

**Arguments**

- **this**: FitDist Object
- **value**: Distribution to fit to

**Examples**

```r
library(cascsim)
data(claimdata)

# frequency fitting example
findex <- new("Index", startDate = as.Date("2012-01-01"), tabulate=TRUE, monthlyIndex = c(rep(1,11),
cumprod(c(1,rep(1.5*(1/12),11))),cumprod(c(1.5,rep((1.3/1.5)^(1/12),11))),
cumprod(c(1.3,rep((1.35/1.3)^(1/12),11))),cumprod(c(1.35,rep((1.4/1.35)^(1/12),11))),1.4))
rawdata <- as.data.frame(as.Date(claimdata[(claimdata[,"LoB"] == "Auto" & claimdata[,"Type"] == "H"),]$occurrenceDate))
colnames(rawdata)<-"occurrenceDate"
xFit <- new("FitDist", observation=rawdata, trend=findex, startDate = as.Date("2012-01-01"),
method="mle",ifreq=TRUE,idate=TRUE, freq="Monthly")
xFit <- setFitdata(xFit)
setTrialDistErr(xFit) <- new("Poisson")
xFit@soutput
observationPlot(xFit)
fitPlot(xFit)
```

**setTruncated**

*Set the indicator of truncated distribution.*

**Description**

Set the indicator of truncated distribution.
Usage

setTruncated(this, ...) <- value

## S4 replacement method for signature 'Distribution,logical'
setTruncated(this) <- value

Arguments

this Distribution Object
...
value Boolean to indicate whether the distribution is truncated by min and max or not.

---

setUpperKeep Set up the upper triangle for non-simulated data.

Description

setUpperKeep sets up the upper triangle for non-simulated data.

Usage

setUpperKeep(object, data, ...)

## S4 method for signature 'Triangle, data.frame'
setUpperKeep(object, data,
  evaluationDate = as.Date("2016-12-31"), lob = "Total",
  ctype = "Total")

Arguments

object Triangle Object
data Claim Data
...
Additional function arguments.
evaluationDate Evaluation Date;
lob Line of Business;
ctype Claim Type.

Examples

library(cascsim)
data(claimdata)
xTri <- new("Triangle", triID = "TRI1", type = "reportedCount", startDate=as.Date("2012-01-01"),
frequency="yearly", sim=1, percentile=50, iRBNER=TRUE, iROPEN=TRUE)
xTri<-setUpperTriangle(xTri,claimdata)
xTri<-setUpperKeep(xTri,claimdata)
setUpperTriangle

setUpperTriangle

Description

setUpperTriangle sets up the upper triangle based on a data file.

Usage

setUpperTriangle(object, data, ...)

# S4 method for signature 'Triangle,data.frame'
setUpperTriangle(object, data,
    evaluationDate = as.Date("2016-12-31"), lob = "Total",
    ctype = "Total")

Arguments

object Triangle Object
data Claim Data
... Additional function arguments.
evaluationDate Evaluation Date;
lob Line of Business;
ctype Claim Type.

Examples

library(cascsim)
data(claimdata)
xTri <- new("Triangle", triID = "TRI1", type = "closedCount", startDate=as.Date("2012-01-01"),
    frequency="quarterly", sim=1, percentile=50, iRBNER=FALSE, iROPEN=TRUE)
xTri<-setUpperTriangle(xTri,claimdata)
xTri<-setUpperKeep(xTri,claimdata)
xTri@upperkeep

xTri <- new("Triangle", triID = "TRI1", type = "incurredLoss", startDate=as.Date("2012-01-01"),
    frequency="yearly", sim=1, percentile=50, iRBNER=TRUE, iROPEN=FALSE)
xTri<-setUpperTriangle(xTri,claimdata)
xTri<-setUpperKeep(xTri,claimdata,lob="Auto",ctype="H")
xTri@upperkeep
setVolList<-  

xTri@upper

xTri <- new("Triangle", triID = "TRI1", type = "closedCount", startDate=as.Date("2012-01-01"),
frequency="quarterly", sim=1, percentile=50)
xTri<-setUpperTriangle(xTri,claimdata)
xTri@upper

xTri <- new("Triangle", triID = "TRI1", type = "incurredLoss", startDate=as.Date("2012-01-01"),
frequency="yearly", sim=1, percentile=50)
xTri<-setUpperTriangle(xTri,claimdata,lob="Auto",ctype="H")
xTri@upper

xTri <- new("Triangle", triID = "TRI1", type = "paidLoss", startDate=as.Date("2012-01-01"),
frequency="yearly", sim=1, percentile=50)
xTri<-setUpperTriangle(xTri,claimdata,lob="Auto",ctype="H")
xTri@upper

---

setVolList<-  

Set the year-to-year loss development factor volatility.

Description

setMeanList<- sets year-to-year loss development factor volatility. It is used to simulate loss
development factor assuming a normal distribution. It can be set to zero for deterministic estimation.
It is only used when ibnerfModel == FALSE.

Usage

setVolList(this, ...) <- value

## S4 replacement method for signature 'DevFac,vector'
setVolList(this) <- value

Arguments

this       DevFac Object
...        Additional function arguments
value      Numeric Vector

Examples

xIBNERFactor <- new("DevFac")
setID(xIBNERFactor)<="IF1"
setFacModel(xIBNERFactor)<=FALSE
setMeanList(xIBNERFactor)<=c(1.26,1.1,1.05,1.02,1)
setVolList(xIBNERFactor)<=rep(0.02,5)
xIBNERFactor
setXname<-  

Set additional explanatory variable names.

Description

setXname<- sets explanatory variable names in addition to "Intercept","DevelopmentYear","IncurredLoss", and "OSRatio". Additional variable names must match exactly with claim data. The xname vector is only used when ibnerfModel == TRUE.

Usage

setXname(this, ...) <- value

## S4 replacement method for signature 'DevFac,vector'
setXname(this) <- value

Arguments

this          DevFac Object
...           Additional function arguments
value         Character Vector

Examples

xIBNERFactor <- new("DevFac")
setID(xIBNERFactor)<-'IF1'
setFacModel(xIBNERFactor)<-'TRUE'
setFun(xIBNERFactor)<-'identity'
setXname(xIBNERFactor)<- c("x1","x2","x3")
setParas(xIBNERFactor)<-c(0.6,-0.2,0.01,-0.3,0.02,0.03,0.01,0.02)
xIBNERFactor<-setDevFac(xIBNERFactor)
xIBNERFactor

setYearlyIndex<-  

Set yearly index values.

Description

setYearlyIndex<- sets yearly index values. Monthly index will be constructed assuming constant growth rate during a year.

Usage

setYearlyIndex(this, ...) <- value

## S4 replacement method for signature 'Index,vector'
setYearlyIndex(this) <- value
**shiftIndex**

**Arguments**

```r
this                Index Object
...                Additional function arguments
value               Numeric Vector
```

**Examples**

```r
xindex <- new("Index")
setID(xindex)<-"IDX1"
setTabulate(xindex)<-TRUE
setYearlyIndex(xindex)<- c(1,1.05,1.2,0.95,1.3)
xindex@yearlyIndex
```

---

**shiftIndex**  
*Shift monthly index with a new start date and replace the unknown index value with zero.*

**Description**

Shift monthly index with a new start date and replace the unknown index value with zero.

**Usage**

```r
shiftIndex(object, ...)
```

```r
# S4 method for signature 'Index'
shiftIndex(object, newStartDate, endDate)
```

**Arguments**

```r
object                Index Object
...                   Additional function arguments
newStartDate          new start date
endDate               end date
```

**Examples**

```r
xindex <- new("Index", indexID = "IDX1", tabulate = FALSE, annualizedRate = 0.03)
xindex<-setIndex(xindex)
xindex@monthlyIndex
shiftIndex(xindex,as.Date("2016-10-15"),as.Date("2018-10-15"))
shiftIndex(xindex,as.Date("2010-10-15"),as.Date("2013-10-15"))
```
**simP0**

Simulate whether claims will have zero payment.

**Description**

Simulate whether claims will have zero payment.

**Usage**

```r
simP0(devYear, zeroProb)
```

**Arguments**

- `devYear`: Development Year. It could be a number or a numeric vector.
- `zeroProb`: A vector that contains the probability of zero payment based on development year.

**Examples**

```r
zeroprobs <- c(0.02, 0.01, 0.005, 0.005, 0.003, 0)
simP0(rep(2, 1000), zeroprobs)
```

---

**simReport**

Generate claim simulation result report in html

**Description**

Generate claim simulation result report in html.

**Usage**

```r
simReport(object, simSummary, ...)
```

```r
## S4 method for signature 'Simulation,data.frame'
simReport(object, simSummary, 
  simTriangle = NA, startDate = as.Date("2012-01-01"),
  evaluationDate = as.Date("2016-12-31"),
  futureDate = as.Date("2017-12-31"), iYear = FALSE)
```
Arguments

- **object**: ClaimType object
- **simSummary**: simulation result summary generated by simSummary
- **...**: Additional parameters that may or may not be used.
- **simTriangle**: triangle summary generated by simTriangle;
- **startDate**: Date after which claims are analyzed;
- **evaluationDate**: Date of evaluation for existing claims and IBNR;
- **futureDate**: Date of evaluation for UPR (future claims);
- **iYear**: Boolean that indicates whether summary by accident year should be produced in the report.

Examples

```r
# run time is about 30s(>10s) and is commented out here to avoid long waiting time
#library(cascsim)
data(claimdata)
#lines <- c("Auto")
#types <- c("N")

#AutoN <- new("ClaimType", line = "Auto", claimType = "N")
#AutoN@exposureIndex <- setIndex(new("Index", indexID="I1", tabulate= FALSE,
#startDate=as.Date("2012-01-01"), annualizedRate = 0)) # level exposure across time
#AutoN@frequency <- new("Poisson", p1 =50)
#AutoN@severityIndex <- setIndex(new("Index", indexID="I2", tabulate= FALSE,
#startDate=as.Date("2012-01-01"), annualizedRate = 0.02)) # assuming a 2% annual inflation
#AutoN@severity <- new("Lognormal", empirical=matrix(c(0,1,100,100),2,2))
#AutoN@deductible <- new("Empirical", empirical=matrix(c(0,1,1e8,1e8),2,2))
#AutoN@limit <- new("Empirical", empirical=matrix(c(0,1,1e8,1e8),2,2))
#AutoN@p0<-new("DevFac",meanList=c(0,0),volList=c(0,0))
#AutoN@reportLag <- new("Exponential", p1 =0.1)
#AutoN@settlementLag <- new("Exponential", p1 =0.05)
#AutoN@Copula <- TRUE # use copula
#AutoN@ssrCopula <- new("CopulaObj", type ="normal", dimension = 3,
#param = c(0,1,0.2,0.1))#A Gaussian Copula
#AutoN@ssrCopula@margin <- c(AutoN@severity,AutoN@settlementLag,AutoN@reportLag)
#AutoN@laeDevFac <- new("DevFac",FacID="F1",FacModel= TRUE,fun="linear",
#paras =c(5,1.5,0.005,1,2,3))
#AutoN@FIBNER <- new("DevFac",FacID="D1",FacModel= FALSE,
#meanList =c(1.2,1.15,1.1,0.05,1),volList =c(0,0,0,0,0))
#AutoN@reopen <- new("DevFac",FacID="D2",FacModel= FALSE,
#meanList =c(0.02,0.015,0.01,0.005,1),volList =c(0.003, 0.002, 0.001, 0.001, 0))
#AutoN@reDevFac <- new("DevFac",FacID="D3",FacModel= FALSE,
#meanList =c(1.05,1.1,1,1,1),volList =c(0.00589,0.0037,0.00632,0.00815,0))
#AutoN@reopenLag <- new("Exponential", p1 =0.01)
#AutoN@resettleLag <- new("Exponential", p1 =0.25)

#simobj <- new("Simulation", lines=lines, types=types,
#claimobjs= list(AutoN),workingFolder=tempdir())
#simobj@simNo <- 1
#simobj@RBNER <-FALSE
#simobj@ROPEN <-FALSE
```
#simobj@iIBNR <- TRUE
#simobj@iUPR <- FALSE
#simdata <- claimSimulation(simobj, claimdata, startDate = as.Date("2012-01-01"),
#evaluationDate = as.Date("2016-12-31"), futureDate = as.Date("2017-12-31"))
#simSummary <- simSummary(simobj, simdata, startDate = as.Date("2012-01-01"))
#simTriangle <- simTriangle(simobj, claimdata, simdata, startDate = as.Date("2016-01-01"))
#simReport(simobj, simSummary, simTriangle, startDate = as.Date("2012-01-01"))

---

simSummary

Claim simulation result summary

Description

Claim simulation result summary

Usage

simSummary(object, simdata, ...)

## S4 method for signature 'Simulation, data.frame'
 simSummary(object, simdata,
 startDate = as.Date("2012-01-01"),
 evaluationDate = as.Date("2016-12-31"),
 futureDate = as.Date("2017-12-31"))

Arguments

object Simulation object
simdata simulation data generated by claimSimulation
... Additional parameters that may or may not be used.
startDate Date after which claims are analyzed;
evaluationDate Date of evaluation for existing claims and IBNR;
futureDate Date of evaluation for UPR (future claims).

Examples

# run time is about 30s (>10s) and is commented out here to avoid long waiting time
#library(cascsim)
data(claimdata)
lines <- c("Auto")
types <- c("N")
AutoN <- new("ClaimType", line = "Auto", claimType = "N")
AutoN@exposureIndex <- setIndex(new("Index", indexID="I1", tabulate= FALSE,
startDate=as.Date("2012-01-01"), annualizedRate = 0)) # level exposure across time
AutoN@Frequency <- new("Poisson", p1 =50)
AutoN@severityIndex <- setIndex(new("Index", indexID="I2", tabulate= FALSE,
startDate=as.Date("2012-01-01"), annualizedRate = 0.02)) #assuming a 2% annual inflation
\#AutoN@severity <- new("Lognormal", p1 =2, p2 =3)  
\#AutoN@deductible <- new("Empirical", empirical=matrix(c(0,1,100,100),2,2))  
\#AutoN@limit <- new("Empirical", empirical=matrix(c(0,1e8,1e8),2,2))  
\#AutoN@p0<-new("DevFac",meanList=c(0,0),volList=c(0,0))  
\#AutoN@reportLag <- new("Exponential", p1 =0.1)  
\#AutoN@settlementLag <- new("Exponential", p1 =0.05)  
\#AutoN@iCopula <- TRUE #use copula  
\#AutoN@ssrCopula <- new("CopulaObj", type ="normal", dimension = 3,  
\#param = c(0.1,0.2,0.1))#A Gaussian Copula  
\#AutoN@ssrCopula@marginal <- c(AutoN@severity,AutoN@settlementLag,AutoN@reportLag)  
\#AutoN@laeDevFac <- new("DevFac",FacID="F1",FacModel= TRUE,fun="linear",  
\#paras =c(5,1.5,0.005,1.2,3))  
\#AutoN@FIBNER <- new("DevFac",FacID="D1",FacModel= FALSE,  
\#meanList =c(1.2,1.15,1.1,1.05,1),volList =c(0,0,0,0,0))  
\#Auto@reopen <- new("DevFac",FacID="D2",FacModel= FALSE,  
\#meanList =c(0.02,0.015,0.01,0.005,0),volList =c(0.003, 0.002, 0.001, 0.001, 0))  
\#Auto@roDevFac <- new("DevFac",FacID="D3",FacModel= FALSE,  
\#meanList =c(1.05,1.1,1,1,1),volList =c(0.00589,0.0037,0.00632,0.00815,0))  
\#Auto@reopenLag <- new("Exponential", p1 =0.01)  
\#Auto@resettleLag <- new("Exponential", p1 =0.25)  
\#simobj <- new("Simulation",lines=lines, types=types,  
\#claimobjs= list(AutoN),workingFolder=tempdir())  
\#simobj@simNo <- 1  
\#simobj@IIBNER <-FALSE  
\#simobj@IROPEN <-FALSE  
\#simobj@IIBNR <-TRUE  
\#simobj@IUFR <-FALSE  
\#simdata <- claimSimulation(simobj,claimdata, startDate = as.Date("2012-01-01"),  
\#evaluationDate = as.Date("2016-12-31"), futureDate = as.Date("2017-12-31"))  
\#simSummary <- simSummary(simobj,simdata, startDate = as.Date("2012-01-01"))

### simTriangle

**Claim simulation result triangles**

**Description**

Claim simulation result triangles

**Usage**

```
simTriangle(object, claimdata, simdata, ...)  
```

## S4 method for signature 'Simulation,data.frame,data.frame'

```
simTriangle(object, claimdata,  
            simdata, frequency = "yearly", startDate = as.Date("2012-01-01"),  
            evaluationDate = as.Date("2016-12-31"), futureDate = as.Date("2017-12-31"))
```
Arguments

object Simulation object
claimdata claim data used as basis for simulation
simdata simulation data generated by claimSimulation

... Additional parameters that may or may not be used.
frequency triangle frequency, either "yearly" or "quarterly";
startDate Date after which claims are analyzed;
evaluationDate Date of evaluation for existing claims and IBNR;
futureDate Date of evaluation for UPR (future claims).

Examples

#run time is about 30s(>10s) and is commented out here to avoid long waiting time
#library(cascsim)
data(claimdata)
#lines <- c("Auto")
#types <- c("N")
#AutoN <- new("ClaimType", line = "Auto", claimType = "N")
#AutoN@exposureIndex <- setIndex(new("Index",indexID="I1",tabulate= FALSE,
#startDate=as.Date("2012-01-01"), annualizedRate = 0)) # level exposure across time
#AutoN@frequency <- new("Poisson", p1 =50)
#AutoN@severityIndex <- setIndex(new("Index",indexID="I2",tabulate= FALSE,
#startDate=as.Date("2012-01-01"), annualizedRate = 0.02)) #assuming a 2% annual inflation
#AutoN@severity <- new("Lognormal", p1 =2, p2 =3)
#AutoN@deductible <- new("Empirical", empirical=matrix(c(0,1,100,100),2,2))
#AutoN@limit <- new("Empirical", empirical=matrix(c(0,1,1e8,1e8),2,2))
#AutoN@p0<-new("DevFac",meanList=c(0,0),volList=c(0,0))
#AutoN@reportLag <- new("Exponential", p1 =0.1)
#AutoN@settlementLag <- new("Exponential", p1 =0.05)
#AutoN@Copula <- TRUE #use copula
#AutoN@ssrCopula <- new("CopulaObj", type ="normal", dimension = 3,
#param = c(0,1.0,2.0,1.1))#A Gaussian Copula
#AutoN@ssrCopula@marginal <- c(AutoN@severity,AutoN@settlementLag,AutoN@reportLag)
#AutoN@laeDevFac <- new("DevFac",FacID="F1",FacModel= TRUE,fun= "linear",
#paras =c(5,1.5,0.005,1.2,3))
#AutoN@FIBNER <- new("DevFac",FacID="D1",FacModel= FALSE,
#meanList =c(1.2,1.15,1.1,1.05,1),volList =c(0,0,0,0,0))
#AutoN@reopen <- new("DevFac",FacID="D2",FacModel= FALSE,
#meanList =c(0.02,0.015,0.01,0.005,0),volList =c(0.003, 0.002, 0.001, 0.001, 0))
#AutoN@reDevFac <- new("DevFac",FacID="D3",FacModel= FALSE,
#meanList =c(1.05,1.1,1,1,1),volList =c(0.00585,0.0037,0.00632,0.00815,0))
#AutoN@reopenLag <- new("Exponential", p1 =0.01)
#AutoN@resettleLag <- new("Exponential", p1 =0.25)
#simobj <- new("Simulation", lines=lines, types=types,
#claimobjs= list(AutoN),workingFolder=tempdir())
#simobj@simNo <- 1
#simobj@iRBNER <-FALSE
#simobj@iROPEN <-FALSE
#simobj@iIBNR <-TRUE
Simulation-class

An S4 class to represent a simulation task.

Description

An S4 class to represent a simulation task.

Slots

startNo  The starting simulation index.

simNo  Number of simulation.

lines  A string vector to identify the business line(s) to be simulated.

types  A string vector to identify the claim types to be simulated.

iRBNER  A Boolean indicating whether IBNER claims need to be simulated.

iROPEN  A Boolean indicating whether claim reopening needs to be simulated.

iIBNR  A Boolean indicating whether IBNR claims need to be simulated.

iUPR  A Boolean indicating whether future claims need to be simulated.

claimobjs  A list of claim objects.

workingFolder  A string to specify the working folder where the simulation results will be saved.

copula  A Boolean indicating whether to use copula for frequency simulation.

copula  Frequency copula.

freqCopula  Frequency copula.

iSummary  A Boolean indicating whether to summarize the simulation results.

iReport  A Boolean indicating whether to generate an HTML report.

iFit  A Boolean indicating whether to fit some simulation parameters based on claim data.

ncores  Number of cores used for simulation.

tag  A unique tag for the simulation object including date and a random ID.

fitfile  A string to set the distribution fitting file name. If omitted, a name based on tag will be used.

copfile  A string to set the copula fitting file name. If omitted, a name based on tag will be used.

facfile  A string to set the factor fitting file name. Factor table is development year dependent. It could be the probability of zero payment, reopen probability, or loss development factors. If omitted, a name based on tag will be used.

fitRpt  A string to set the distribution fitting HTML report file name. If omitted, a name based on tag will be used.
TEKurt

simfile A string to set the simulation result file name. If omitted, a name based on tag will be used.
sumfile A string to set the summary file name. If omitted, a name based on tag will be used.
plog A string to set the parallel run log file name. If omitted, a name based on tag will be used.
htmlRpt A string to set the html report name. If omitted, a name based on tag will be used.
libpath A string to the R library folder where required packages are installed.

---

**Description**

Calculate Theoretical Excessive Kurtosis of distribution. min and max are not applied

**Usage**

```
TEKurt(object, ...)  
## S4 method for signature 'Normal'
TEKurt(object) 
## S4 method for signature 'Beta'
TEKurt(object) 
## S4 method for signature 'Exponential'
TEKurt(object) 
## S4 method for signature 'Gamma'
TEKurt(object) 
## S4 method for signature 'Geometric'
TEKurt(object) 
## S4 method for signature 'Lognormal'
TEKurt(object) 
## S4 method for signature 'NegativeBinomial'
TEKurt(object) 
## S4 method for signature 'Pareto'
TEKurt(object) 
## S4 method for signature 'Poisson'
TEKurt(object)
```

TMean

## S4 method for signature 'Uniform'
TEKurt(object)

## S4 method for signature 'Weibull'
TEKurt(object)

### Arguments

- object: Distribution Object
- ...: Additional function arguments

### Examples

```r
xPareto <- new("Pareto", p1=20, p2=5)
TEKurt(xPareto)
```

---

**TMean**

*Calculate Theoretical Mean of distribution. min and max are not applied*

---

### Description

Calculate Theoretical Mean of distribution. min and max are not applied

### Usage

```r
TMean(object, ...)
```

## S4 method for signature 'Normal'
TMean(object)

## S4 method for signature 'Beta'
TMean(object)

## S4 method for signature 'Exponential'
TMean(object)

## S4 method for signature 'Gamma'
TMean(object)

## S4 method for signature 'Geometric'
TMean(object)

## S4 method for signature 'Lognormal'
TMean(object)

## S4 method for signature 'NegativeBinomial'
TMean(object)

## S4 method for signature 'Pareto'
TMean(object)

## S4 method for signature 'Poisson'
TMean(object)

## S4 method for signature 'Uniform'
TMean(object)

## S4 method for signature 'Weibull'
TMean(object)

Arguments

object Distribution Object

... Additional function arguments

Examples

xPareto <- new("Pareto", p1=20, p2=3)
TMean(xPareto)

toDate

\begin{verbatim}
Convert US date mm/dd/yyyy to yyyy-mm-dd format
\end{verbatim}

toDate(d)

Arguments

d vector of dates in possible US format

Examples

toDate("3/5/2017")
Triangle-class

An S4 class to represent a triangle or rectangle object.

Description

An S4 class to represent a triangle or rectangle object.

Slots

triID A character string to identify the triangle object.
type A character string that indicates the triangle type, such as reportedCount, closedCount, paidLoss, and incurredLoss.
startDate The start date for the accident year or Quarter.
frequency A character that indicates the frequency of the triangle, "yearly" or "quarterly".
sim A number that indicates the simulation number used to complete the rectangle. Zero means using the average value.
percentile A number that indicates the percentile used to complete the rectangle. It is only used when sim is NA.
iRBNER A Boolean that indicates whether open claims are simulated. If not, current information will be used for constructing rectangles. Otherwise, simulated data will be used.
iROPEN A Boolean that indicates whether claim reopen are simulated. If not, current information will be used for constructing rectangles. Otherwise, simulated data will be used.
percentile A number that indicates the percentile used to complete the rectangle. It is only used when sim is NA.
upper A matrix that contains the upper triangle based on claim data.
upperkeep A matrix that contains the upper triangle that are not simulated. It will be used to construct the rectangle for the non-simulated part.
rectangle A matrix that contains the entire rectangle based on simulation data.

truncate

Truncate a numeric vector

Description

Truncate a numeric vector

Usage

truncate(x, lower, upper)
Arguments

- **x**: A numeric vector
- **lower**: Lower bound
- **upper**: Upper bound

Examples

```
trunc(rnorm(100, 3, 6), 0, 7)
```

---

**TSD**  
*Calculate Theoretical Standard Deviation of distribution. min and max are not applied*

---

Description

Calculate Theoretical Standard Deviation of distribution. min and max are not applied

Usage

```
TSD(object, ...)
```

- **S4 method for signature 'Normal'**
  ```
  TSD(object)
  ```

- **S4 method for signature 'Beta'**
  ```
  TSD(object)
  ```

- **S4 method for signature 'Exponential'**
  ```
  TSD(object)
  ```

- **S4 method for signature 'Gamma'**
  ```
  TSD(object)
  ```

- **S4 method for signature 'Geometric'**
  ```
  TSD(object)
  ```

- **S4 method for signature 'Lognormal'**
  ```
  TSD(object)
  ```

- **S4 method for signature 'NegativeBinomial'**
  ```
  TSD(object)
  ```

- **S4 method for signature 'Pareto'**
  ```
  TSD(object)
  ```

- **S4 method for signature 'Poisson'**
  ```
  TSD(object)
  ```
TSkewness

TSD(object)

## S4 method for signature 'Uniform'
TSD(object)

## S4 method for signature 'Weibull'
TSD(object)

Arguments

object Distribution Object
...
Additional function arguments

Examples

xPareto <- new("Pareto", p1=20, p2=3)
TSD(xPareto)

---

TSkewness Calculate Theoretical Skewness of distribution. min and max are not applied

Description

Calculate Theoretical Skewness of distribution. min and max are not applied

Usage

TSkewness(object, ...)

## S4 method for signature 'Normal'
TSkewness(object)

## S4 method for signature 'Beta'
TSkewness(object)

## S4 method for signature 'Exponential'
TSkewness(object)

## S4 method for signature 'Gamma'
TSkewness(object)

## S4 method for signature 'Geometric'
TSkewness(object)

## S4 method for signature 'Lognormal'
TSkewness(object)
## S4 method for signature 'NegativeBinomial'
TSkewness(object)

## S4 method for signature 'Pareto'
TSkewness(object)

## S4 method for signature 'Poisson'
TSkewness(object)

## S4 method for signature 'Uniform'
TSkewness(object)

## S4 method for signature 'Weibull'
TSkewness(object)

### Arguments

- **object**: Distribution Object
- **...**: Additional function arguments

### Examples

```r
xPareto <- new("Pareto", p1=20, p2=4)
TSkewness(xPareto)
```

---

**ultiDevFac**

*Calculate ultimate development factor based on current development year, a mean development factor schedule and its volatility. It is used to simulate the ultimate loss for open claims.*

### Description

Calculate ultimate development factor based on current development year, a mean development factor schedule and its volatility. It is used to simulate the ultimate loss for open claims.

### Usage

```r
ultiDevFac(Years, meanDevFac, sdDevFac = rep(0, length(meanDevFac)),
            distType = "normal")
```

### Arguments

- **Years**: Include two columns: Current development year and Settlement Year
- **meanDevFac**: A vector that contains the expected development factor schedule for Normal distribution. It is mu for Lognormal distribution and shape for Gamma distribution.
ultiDevFac

sdDevFac  A vector that contains the standard deviation of expected development factor schedule for Normal distribution. It is sigma for Lognormal distribution and scale for Gamma distribution.

distType  distribution type for development factor. It can be "normal", "lognormal" or "gamma".

Examples

meanfac<-c(1.1,1.08,1.05,1.03,1.01,1)
volfac<-rep(0.02,6)
years<-matrix(c(1:6),3,2)
ultiDevFac(years,meanfac,volfac)
# Index

*Topic **datasets**  
  - claimdata, 5

| CDFPlot | 4 | CDFPlot, ANY-method (CDFPlot) | 4 | CDFPlot, FitDist-method (CDFPlot) | 4 |
| ChiSqrTest | 5 | ChiSqrTest, ANY-method (ChiSqrTest) | 5 |
| claimFitting | 5 | claimFitting, ANY-method (claimFitting) | 6 |
| claimSample | 8 | claimSample, ANY-method (claimSample) | 8 |
| claimSample, ClaimType-method (claimSample) | 8 |
| claimSimulation | 8 | claimSimulation, ANY-method (claimSimulation) | 8 |
| claimSimulation, Simulation-method (claimSimulation) | 8 |
| ClaimType-class | 10 |
| copulaDataPlot | 11 | copulaDataPlot, ANY-method (copulaDataPlot) | 11 |
| copulaDataPlot, CopulaObj-method (copulaDataPlot) | 11 |
| copulaFit | 12 | copulaFit, ANY-method (copulaFit) | 12 |
| copulaFit, CopulaObj-method (copulaFit) | 12 |
| copulaFitPlot | 13 | copulaFitPlot, ANY-method (copulaFitPlot) | 13 |
| copulaFitPlot, CopulaObj-method (copulaFitPlot) | 13 |
| copulaObj-class | 14 | copulaPlot | 14 |
| copulaPlot, ANY-method (copulaPlot) | 14 |
| copulaPlot, CopulaObj-method (copulaPlot) | 14 |
| copulaSample | 15 | copulaSample, ANY-method (copulaSample) | 15 |
| copulaSample, CopulaObj, numeric-method (copulaSample) | 15 |
| dempirical (pempirical) | 40 |
| Density | 16 | Density, ANY-method (Density) | 16 |
| Density, Beta-method (Density) | 16 |
| Density, Empirical-method (Density) | 16 |
| Density, Exponential-method (Density) | 16 |
| Density, Gamma-method (Density) | 16 |
| Density, Geometric-method (Density) | 16 |
| Density, Lognormal-method (Density) | 16 |
| Density, NegativeBinomial-method (Density) | 16 |
| Density, Normal-method (Density) | 16 |
| Density, Pareto-method (Density) | 16 |
| Density, Poisson-method (Density) | 16 |
| Density, Uniform-method (Density) | 16 |
| Density, Weibull-method (Density) | 16 |
| DevFac-class | 17 |
| Distribution-class | 18 | doPlot | 18 |
| doPlot, ANY-method (doPlot) | 18 |
| doPlot, Distribution-method (doPlot) | 18 |
| doSample | 19 | doSample, ANY-method (doSample) | 19 |
| doSample, Beta, numeric-method (doSample) | 19 |
| doSample, Empirical, numeric-method (doSample) | 19 |
| doSample, Exponential, numeric-method (doSample) | 19 |

92
INDEX

getObservation, FitDist-method
(getObservation), 34
getTrend, 35
getTrend, ANY-method (getTrend), 35
getTrend, FitDist-method (getTrend), 35

Index-class, 36

KSTest, 36
KSTest, ANY-method (KSTest), 36
KSTest, FitDist-method (KSTest), 36

mpareto, 37

nloglik, 38

observationPlot, 38
observationPlot, ANY-method (observationPlot), 38
observationPlot, FitDist-method (observationPlot), 38

PDFPlot, 39
PDFPlot, ANY-method (PDFPlot), 39
PDFPlot, FitDist-method (PDFPlot), 39

pempirical, 40

plotText, 41

PPPlot, 41
PPPlot, ANY-method (PPPlot), 41
PPPlot, FitDist-method (PPPlot), 41

Probability, 42
Probability, ANY-method (Probability), 42
Probability, Beta-method (Probability), 42
Probability, Empirical-method (Probability), 42
Probability, Exponential-method (Probability), 42
Probability, Gamma-method (Probability), 42
Probability, Geometric-method (Probability), 42
Probability, Lognormal-method (Probability), 42
Probability, NegativeBinomial-method (Probability), 42
Probability, Normal-method (Probability), 42

expectZeros, 31

FitDist-class, 31
fitPlot, 32
fitPlot, ANY-method (fitPlot), 32
fitPlot, FitDist-method (fitPlot), 32

getCopula, 33
getCopula, ANY-method (getCopula), 33
getCopula, CopulaObj-method (getCopula), 33
getIndex, 34
getIndex, ANY-method (getIndex), 34
getIndex, Index-method (getIndex), 34
getObservation, 34
getObservation, ANY-method (getObservation), 34

dosample, Gamma, numeric-method (dosample), 19
dosample, Geometric, numeric-method (dosample), 19
dosample, Lognormal, numeric-method (dosample), 19
dosample, NegativeBinomial, numeric-method (dosample), 19
dosample, Normal, numeric-method (dosample), 19

dosample, Pareto, numeric-method (dosample), 19

dosample, Poisson, numeric-method (dosample), 19

dosample, Uniform, numeric-method (dosample), 19

dosample, Weibull, numeric-method (dosample), 19

dpareto (mpareto), 37

dbeta, 20
dtempirical, 21
dexp, 22
dgamma, 23
dgeom, 24
dlnorm, 25
dtnbinom, 26
dtnorm, 27
dtpareto, 28
dtpois, 29
dweibull, 30

expectZeros, 31

FitDist-class, 31
fitPlot, 32
fitPlot, ANY-method (fitPlot), 32
fitPlot, FitDist-method (fitPlot), 32

getCopula, 33
getCopula, ANY-method (getCopula), 33
getCopula, CopulaObj-method (getCopula), 33
getIndex, 34
getIndex, ANY-method (getIndex), 34
getIndex, Index-method (getIndex), 34
getObservation, 34
getObservation, ANY-method (getObservation), 34
Probability, Pareto-method (Probability), 42
Probability, Poisson-method (Probability), 42
Probability, Uniform-method (Probability), 42
Probability, Weibull-method (Probability), 42
ptbeta (dbeta), 20
ptempirical (dtempirical), 21
ptexp (dtexp), 22
ptgamma (dtgamma), 23
ptgeom (dgeom), 24
ptlnorm (dtlnorm), 25
ptnbinom (dtnbinom), 26
ptnorm (dtnorm), 27
ptpareto (dtpareto), 28
ptpois (dtpois), 29
ptweibull (dtweibull), 30
qempirical (pempirical), 40
qpareto (mpareto), 37
QQPlot, ANY-method (QQPlot), 43
QQPlot, FitDist-method (QQPlot), 43
qtbeta (dbeta), 20
qtempirical (dtempirical), 21
qtexp (dtexp), 22
qtgamma (dtgamma), 23
qtgeom (dgeom), 24
qtlnorm (dtlnorm), 25
qtnbinom (dtnbinom), 26
qtlnorm (dtlnorm), 27
qtpareto (dtpareto), 28
qtpois (dtpois), 29
qtweibull (dtweibull), 30
Quantile, ANY-method (Quantile), 44
Quantile, Beta-method (Quantile), 44
Quantile, Empirical-method (Quantile), 44
Quantile, Exponential-method (Quantile), 44
Quantile, Gamma-method (Quantile), 44
Quantile, Geometric-method (Quantile), 44
Quantile, Lognormal-method (Quantile), 44
Quantile, NegativeBinomial-method (Quantile), 44
Quantile, Normal-method (Quantile), 44
Quantile, Poisson-method (Quantile), 44
Quantile, Uniform-method (Quantile), 44
Quantile, Weibull-method (Quantile), 44
rempirical (pempirical), 40
rpareto (mpareto), 37
rreopen, 45
rtbeta (dbeta), 20
rtempirical (dtempirical), 21
rtexp (dtexp), 22
rtgamma (dtgamma), 23
rtgeom (dgeom), 24
rtlnorm (dtlnorm), 25
rtnbinom (dtnbinom), 26
rtlnorm (dtlnorm), 27
rtpareto (dtpareto), 28
rtpois (dtpois), 29
rtpareto (dtpareto), 28
rtweibull (dtweibull), 30
sampleKurtosis, 46
sampleKurtosis, ANY-method (sampleKurtosis), 46
sampleKurtosis, Distribution-method (sampleKurtosis), 46
sampleMean, 47
sampleMean, ANY-method (sampleMean), 47
sampleMean, Distribution-method (sampleMean), 47
sampleSd, 47
sampleSd, ANY-method (sampleSd), 47
sampleSd, Distribution-method (sampleSd), 47
sampleSkew, 48
sampleSkew, ANY-method (sampleSkew), 48
sampleSkew, Distribution-method (sampleSkew), 48
setAnnualizedRate, ANY-method (setAnnualizedRate<->), 48
setAnnualizedRate<->, 48
setAnnualizedRate<-, Index, numeric-method (setAnnualizedRate<->), 48
setCopulaParam, ANY-method (setCopulaParam<->), 49
setCopulaParam<-, 49
setCopulaParam<-, CopulaObj, numeric-method (setCopulaParam<->), 49
setCopulaType, ANY-method (setCopulaType<->), 50
setCopulaType<-, 50
INDEX

setCopulaType<-, CopulaObj, character-method (setCopulaType<-), 50
setDevFac, 50
setDevFac, ANY-method (setDevFac), 50
setDevFac, DevFac-method (setDevFac), 50
setDf, ANY-method (setDf<-), 51
setDf<-, 51
setDf<-, CopulaObj, numeric-method (setDf<->), 51
setDimension, ANY-method (setDimension<->), 52
setDimension<-, 52
setDimension<-, CopulaObj, numeric-method (setDimension<->), 52
setDispstr, ANY-method (setDispstr<->), 52
setDispstr<-, 52
setDispstr<-, CopulaObj, character-method (setDispstr<->), 52
setEmpirical, ANY-method (setEmpirical<->), 53
setEmpirical<-, 53
setEmpirical<-, Distribution, matrix-method (setEmpirical<->), 53
setFacModel, ANY-method (setFacModel<->), 54
setFacModel<-, 54
setFacModel<-, DevFac, logical-method (setFacModel<->), 54
setFittedDist, ANY-method (setFittedDist<->), 56
setFittedDist<-, 56
setFittedDist<-, DevFac, logical-method (setFittedDist<->), 56
setFreq, ANY-method (setFreq<-), 57
setFreq<-, 57
setFreq<-, DevFac, character-method (setFreq<->), 57
setFun, ANY-method (setFun<-), 58
setFun<-, 58
setFittedDist<-, DevFac, character-method (setFittedDist<->), 58
setID, ANY-method (setID<-), 58
setID<-, 58
setID<-, DevFac, character-method (setID<->), 58
setID<-, Index, character-method (setID<->), 58
setidate, ANY-method (setidate<-), 59
setidate<-, 59
setidate<-, FitDist, logical-method (setidate<->), 59
setifreq, ANY-method (setifreq<-), 60
setifreq<-, 60
setifreq<-, FitDist, logical-method (setifreq<->), 60
setIndex, 61
setIndex, ANY-method (setIndex<->), 61
setIndex, Index-method (setIndex<->), 61
setMarginal, ANY-method (setMarginal<->), 62
setMarginal<-, 62
setMarginal<-, CopulaObj, list-method (setMarginal<->), 62
setMeanList, ANY-method (setMeanList<->), 62
setMeanList<-, 62
setMonthlyIndex<-, DevFac, vector-method (setMonthlyIndex<->), 62
setMin, 63
setMin, ANY-method (setMin<->), 63
setMin, Distribution-method (setMin<->), 63
setMonthlyIndex, ANY-method (setMonthlyIndex<->), 64
setMonthlyIndex<-, 64
setMonthlyIndex<-, Index, vector-method (setMonthlyIndex<->), 64
setObservation, ANY-method (setObservation<->), 64
setObservation<-, 64
setObservation<-, CopulaObj, matrix-method (setObservation<->), 64
setObservation<-, FitDist, matrix-method (setObservation<->), 64
setParams, ANY-method (setParams<->), 65
setParams<-, 65
setParams<-, Distribution, numeric-method (setParams<->), 65
INDEX

setParas, ANY-method (setParas<-), 66
setParas<-, 66
setParas<-, DeVFac, vector-method
(setParas<-), 66
setprobs, ANY-method (setprobs<-), 66
setprobs<-, 66
setprobs<-, FitDist, vector-method
(setprobs<-), 66
setRange, ANY-method (setRange<-), 67
setRange<-, 67
setRange<-, Distribution, numeric-method
(setRange<-), 67
setRectangle, ANY-method (setRectangle)
setRectangle<-, 68
setRectangle, ANY-method (setRectangle), 68
setRectangle, Triangle, data.frame-method
(setRectangle), 68
setSeasonality, ANY-method
(setSeasonality<-), 68
setSeasonality<-, 68
setSeasonality<-, Index, vector-method
(setSeasonality<-), 68
setStartDate, ANY-method
(setStartDate<-), 69
setStartDate<-, 69
setStartDate<-, Index, Date-method
(setStartDate<-), 69
setTabulate, ANY-method (setTabulate<-), 69
setTabulate<-, 69
setTabulate<-, Index, logical-method
(setTabulate<-), 69
setTrend, ANY-method (setTrend<-), 70
setTrend<-, 70
setTrend<-, FitDist, Index-method
(setTrend<-), 70
setTrialDist, ANY-method
(setTrialDist<-), 71
setTrialDist<-, 71
setTrialDist<-, FitDist, Distribution-method
(setTrialDist<-), 71
setTrialDistErr, ANY-method
(setTrialDistErr<-), 72
setTrialDistErr<-, 72
setTrialDistErr<-, FitDist, Distribution-method
(setTrialDistErr<-), 72
setTruncated, ANY-method
(setTruncated<-), 72
setTruncated<-, Distribution, logical-method
(setTruncated<-), 72
setUpperKeep, 73
setUpperKeep, ANY-method (setUpperKeep), 73
setUpperKeep, Triangle, data.frame-method
(setUpperKeep), 73
setUpperTriangle, 74
setUpperTriangle, ANY-method
(setUpperTriangle), 74
setUpperTriangle, Triangle, data.frame-method
(setUpperTriangle), 74
setVolList, ANY-method (setVolList<-), 75
setVolList<-, 75
setVolList<-, DeVFac, vector-method
(setVolList<-), 75
setXname, ANY-method (setXname<-), 76
setXname<-, 76
setXname<-, DeVFac, vector-method
(setXname<-), 76
setYearlyIndex, ANY-method
(setYearlyIndex<-), 76
setYearlyIndex<-, 76
setYearlyIndex<-, Index, vector-method
(setYearlyIndex<-), 76
shiftIndex, 77
shiftIndex, ANY-method (shiftIndex), 77
shiftIndex, Index-method (shiftIndex), 77
simP0, 78
simReport, 78
simReport, ANY-method (simReport), 78
simReport, Simulation, data.frame-method
(simReport), 78
simSummary, 80
simSummary, ANY-method (simSummary), 80
simSummary, Simulation, data.frame-method
(simSummary), 80
simTriangle, 81
simTriangle, ANY-method (simTriangle), 81
simTriangle, Simulation, data.frame, data.frame-method
(simTriangle), 81
Simulation-class, 83
TEKurt, 84
TEKurt, ANY-method (TEKurt), 84
TEKurt, Beta-method (TEKurt), 84
TEKurt, Exponential-method (TEKurt), 84
TEKurt, Gamma-method (TEKurt), 84
INDEX

TEKurt, Geometric-method (TEKurt), 84
TEKurt, Lognormal-method (TEKurt), 84
TEKurt, NegativeBinomial-method (TEKurt), 84
TEKurt, Normal-method (TEKurt), 84
TEKurt, Pareto-method (TEKurt), 84
TEKurt, Uniform-method (TEKurt), 84
TEKurt, Weibull-method (TEKurt), 84
TMean, 85
TMean, ANY-method (TMean), 85
TMean, Beta-method (TMean), 85
TMean, Exponential-method (TMean), 85
TMean, Gamma-method (TMean), 85
TMean, Geometric-method (TMean), 85
TMean, Lognormal-method (TMean), 85
TMean, NegativeBinomial-method (TMean), 85
TMean, Normal-method (TMean), 85
TMean, Pareto-method (TMean), 85
TMean, Poisson-method (TMean), 85
TMean, Uniform-method (TMean), 85
TMean, Weibull-method (TMean), 85
toDate, 86
Triangle-class, 87
truncate, 87
TSD, 88
TSD, ANY-method (TSD), 88
TSD, Beta-method (TSD), 88
TSD, Exponential-method (TSD), 88
TSD, Gamma-method (TSD), 88
TSD, Geometric-method (TSD), 88
TSD, Lognormal-method (TSD), 88
TSD, NegativeBinomial-method (TSD), 88
TSD, Normal-method (TSD), 88
TSD, Pareto-method (TSD), 88
TSD, Poisson-method (TSD), 88
TSD, Uniform-method (TSD), 88
TSD, Weibull-method (TSD), 88
TSkewness, 89
TSkewness, ANY-method (TSkewness), 89
TSkewness, Beta-method (TSkewness), 89
TSkewness, Exponential-method (TSkewness), 89
TSkewness, Gamma-method (TSkewness), 89
TSkewness, Geometric-method (TSkewness), 89
TSkewness, Lognormal-method (TSkewness), 89
ultiDevFac, 90