Package 'cascsim'

June 4, 2019

Title Casualty Actuarial Society Individual Claim Simulator

Version 0.3

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. Parodi (2013) <https://www.actuaries.org.uk/documents/triangle-free-reserving-non-traditional-framework-estimating-reserves-and-reserve-uncertainty>.

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 2019-06-04 11:40:06 UTC

R topics documented:

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)

#frequecy 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
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 idemnity 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

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

## S4 method for signature 'Simulation,data.frame'
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 reopenness analysis
`...`	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;
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, settlemet 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

```
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, ...)

## 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 reopenness analysis;
startDate	Date from which claim data is included in the analysis;
evaluationDate	Date of evaluation.

Examples

```
#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,iROPEN=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 reopenness 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(0,1,1e8,1e8),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))
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(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@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)
```

copulaFit *Copula fitting*

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
```

copulaFitPlot *Visualization Copula fitting*

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)
```

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)
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

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

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

Arguments

object	Copula Object
n	Number of samples
...	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)
copulaSample(nom.cop,100)
```

`Density` *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)
```

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

```
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

```
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(claim,limit)-deductible)

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

Random generation of Truncated empirical distribution max(0,min(claim,limit)-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(claim,limit)-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(claim,limit)-deductible)
`p`	Value of the probability
`n`	Number of samples

Examples

```
#discrete distribution
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
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
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(1.1,99)))),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`	Value of the variable after deductible and limit max(0,min(claim,limit)-deductible)
`rate`	Distribution 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

```
dtexp(5,0.1)
ptexp(5,0.1)
qtexp(0.5,0.1)
rtexp(100,0.1)
```

dtgamma *Density function of Truncated Gamma Distribution*

Description

Density function of Truncated Gamma Distribution

Cumulative probability function of Truncated Gamma Distribution

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

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

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

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

dtgeom *Density function of Truncated Geometric Distribution*

Description

Density function of Truncated Geometric Distribution

Cumulative probability function of Truncated Geometric Distribution

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

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

Usage

```
dtgeom(x, prob, min = 0, max = 1e+09)

ptgeom(q, prob, min = 0, max = 1e+09)

qtgeom(p, prob, min = 0, max = 1e+09)

rtgeom(n, prob, min = 0, max = 1e+09)
```

Arguments

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

```
dtgeom(3,0.3)
ptgeom(3,0.3)
qtgeom(0.7,0.3)
rtgeom(100,0.3)
```

`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

```
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

```
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

```
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

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

dtnorm *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 max(0,min(claim,limit)-deductible)

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

Usage

```
dtpareto(x, xm, alpha, min = xm, max = 1e+09)

ptpareto(q, xm, alpha, min = xm, max = 1e+09)

qtpareto(p, xm, alpha, min = xm, max = 1e+09)

rtpareto(n, xm, alpha, min = xm, 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)
```

dtpois *Density function of Truncated Poisson Distribution*

Description

Density function of Truncated Poisson Distribution

Cumulative probability function of Truncated Poisson Distribution

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

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

Usage

```
dtpois(x, lambda, min = 0, max = 1e+09)

ptpois(q, lambda, min = 0, max = 1e+09)

qtpois(p, lambda, min = 0, max = 1e+09)

rtpois(n, lambda, min = 0, max = 1e+09)
```

Arguments

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

```
dtpois(3,5)
ptpois(3,5)
qtpois(0.6,5)
rtpois(100,5)
```

`dtweibull` *Density function of Truncated Weibull Distribution*

Description

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.

`method` 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".

`iDL` 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.

`kstest` 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

Examples

```
library(cascsim)
data(claimdata)

#frequecy 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

Examples

```
library(cascsim)
data(claimdata)

#frequecy 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)
```

---

getTrend *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)

#frequecy 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*xm^alpha/x^(alpha+1))*

Description

Moment function of Pareto Distribution (PDF: alpha*xm^alpha/x^(alpha+1))

Density function of Pareto Distribution (PDF: alpha*xm^alpha/x^(alpha+1))

Cumulative probability function of Pareto Distribution (CDF: 1-(xm/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 *Negative Loglikelihood.*

Description

Negative Loglikelihood.

Usage

```
nloglik(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

```
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 *Plotting the data for distribution fitting*

Description

Plotting the data for distribution fitting

Usage

```
observationPlot(object, ...)

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

Arguments

object	FitDist Object
...	Additional function arguments

Examples

```
library(cascsim)
data(claimdata)

#frequecy 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)
```

PDFPlot *Plotting the PDF of data and fitted distribution*

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

```
library(cascsim)
data(claimdata)

#frequecy 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

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)

#frequecy 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)
```

```
## 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

```
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

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)

#frequecy 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

```
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

```
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

```
reopenprob<-c(0.02,0.01,0.005,0.005,0.003,0)
rreopen(rep(2,1000),reopenprob)
```

`sampleKurtosis` *Calculate the excess kurtosis of 10000 sampled values from the distribution.*

Description

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

Usage

```
sampleKurtosis(object, ...)

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

Arguments

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

Examples

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

sampleMean — *Calculate the mean of 100000 sampled values from the distribution.*

Description

Calculate the mean of 100000 sampled values from the distribution.

Usage

```
sampleMean(object, ...)

## S4 method for signature 'Distribution'
sampleMean(object)
```

Arguments

object	A Distribution Object
...	Additional function arguments

Examples

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

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

```
sampleSd(object, ...)

## S4 method for signature 'Distribution'
sampleSd(object)
```

Arguments

object	A Distribution Object
...	Additional function arguments

Examples

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

`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

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

`setCopulaParam<-` *Set copula parameters.*

Description

Set copula parameters.

Usage

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

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

Arguments

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

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)
setCopulaParam(cop) <- 0.6
```

setCopulaType<- *Set copula type.*

Description

Set copula type.

Usage

```
setCopulaType(this, ...) <- value

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

Arguments

this	Copula Object
...	Additional function arguments
value	The copula type

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)
setCopulaType(cop) <- "joe"
```

setDevFac *Set up an IBNER loss development schedule.*

Description

setDevFac sets a loss development schedule, from either a predictive model or a year-to-year factor vector.

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
...	Additional function arguments
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,truncated=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
```

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<-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)
setDispstr(cop) <- "ex"
```

---

setEmpirical<- *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.*

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, 11,12,13,14,15), nrow = 5, ncol = 2) dist

`setFacModel<-` *Determine whether the development factor is determined by a predictive model or a fixed schedule by development year*

Description

Determine whether the development factor is determined by a predictive model or a fixed schedule by development year

Usage

```
setFacModel(this, ...) <- value

## S4 replacement method for signature 'DevFac,logical'
setFacModel(this) <- value
```

Arguments

`this`	DevFac Object
`...`	Additional function arguments
`value`	Logical Value (default:FALSE)

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
```

`setFitdata` *Preparing the input data (observation) for distribution fitting, including detrending, translating occurrence dates to frequency data, etc.*

Description

Preparing the input data (observation) for distribution fitting, including detrending, translating occurrence dates to frequency data, etc.

Usage

```
setFitdata(object, ...)

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

Arguments

object	FitDist Object
...	Additional function arguments

Examples

```
library(cascsim)
data(claimdata)

#frequecy 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

this	FitDist Object
...	Additional function arguments
value	A character string: "mle", "mme", or "qme"

Examples

```
library(cascsim)
data(claimdata)

#frequecy 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")
setfitmethod(xFit) <- "mme"
xFit@method
```

`setFittedDist<-` *Directly set the fitted distribution without fitting it to the data.*

Description

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

Usage

```
setFittedDist(this) <- value

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

Arguments

`this`	FitDist Object
`value`	Fitted distribution

Examples

```
library(cascsim)
data(claimdata)

#frequecy 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@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)

#frequecy 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

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

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

Arguments

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

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
```

setID<- *setID Set the ID for an object*

Description

setID Set the ID for an object

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
...	Additional function arguments
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
...	Additional function arguments
value	A boolean

Examples

```
library(cascsim)
data(claimdata)

#frequecy 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=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

```
library(cascsim)
data(claimdata)

#frequecy 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")
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 *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, ...)

## 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)
```

setMonthlyIndex<- *Set monthly index values.*

Description

setMonthlyIndex<- sets monthly index values.

Usage

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

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

Arguments

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

Examples

```
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

```
setObservation(this) <- value

## S4 replacement method for signature 'CopulaObj,matrix'
setObservation(this) <- value

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

---

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

setParas<- *Set the values of model parameters.*

Description

setParas<- sets model parameters. Their order must match the order of c("Intercept","DevelopmentYear","IncurredLoss","C
"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

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

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

Arguments

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

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

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

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

Arguments

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

Examples

```
library(cascsim)
data(claimdata)

#frequecy 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")
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

```
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

```
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

```
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
```

---

setStartDate<- *Set the start date for the claim simulation exercise*

Description

Set the start date for the claim simulation exercise

Usage

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

## S4 replacement method for signature 'Index,Date'
setStartDate(this) <- value
```

Arguments

this	Self
...	Additional function arguments
value	Start Date

---

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

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

## S4 replacement method for signature 'Index,logical'
setTabulate(this) <- value
```

Arguments

this	Index Object
...	Additional function arguments
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
...	Additional function arguments
value	An Index Object

Examples

```
library(cascsim)
data(claimdata)

#frequecy 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, startDate = as.Date("2012-01-01"),
```

```
method="mle",ifreq=TRUE,idate=TRUE, freq="Monthly")
setTrend(xFit) <- findex
xFit@trend
```

---

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)

#frequecy 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)
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

```
setTrialDistErr(this) <- value

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

Arguments

this	FitDist Object
value	Distribution to fit to

Examples

```
library(cascsim)
data(claimdata)

#frequecy 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
...	Additional function arguments.
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)
```

```
xTri@upperkeep

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
```

`setUpperTriangle` *Set up the upper triangle based on claim data.*

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 = "reportedCount", startDate=as.Date("2012-01-01"),
frequency="yearly", sim=1, percentile=50)
xTri<-setUpperTriangle(xTri,claimdata)
```

```
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
```

Arguments

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

Examples

```
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

```
shiftIndex(object, ...)

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

Arguments

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

Examples

```
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

```
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

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

simReport *Generate claim simulation result report in html*

Description

Generate claim simulation result report in html

Usage

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

## 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

```
#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@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))
#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(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@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"))
#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(0,1,1e8,1e8),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@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))
#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(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@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"))
#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@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))
#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(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@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"))
#simSummary <- simSummary(simobj,simdata, startDate = as.Date("2012-01-01"))
#simTriangle <- simTriangle(simobj,claimdata,simdata, startDate = as.Date("2012-01-01"))
```

`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.

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

`freqCopula` Frequency copula.

`iSummary` A Boolean indicating whether to summarzie 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 dependant. 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.

`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 liabrary folder where required packages are installed.

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

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)
```

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

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

Arguments

object	Distribution Object
...	Additional function arguments

Examples

```
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

```
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 *Convert US date mm/dd/yyyy to yyyy-mm-dd format*

Description

Convert US date mm/dd/yyyy to yyyy-mm-dd format

Usage

```
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)

## 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

```
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

```
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.

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