Package 'OpVaR'

October 12, 2022

Type Package

Title Statistical Methods for Modelling Operational Risk

Version 1.2

Date 2021-09-08

Author Christina Zou [aut,cre],
Marius Pfeuffer [aut],
Matthias Fischer [aut],
Nina Buoni [ctb], Kristina Dehler [ctb], Nicole Derfuss [ctb], Benedikt Graswald [ctb], Linda Moestel [ctb], Jixuan Wang [ctb], Leonie Wicht [ctb]

Maintainer Christina Zou <christina.zou@maths.ox.ac.uk>

Description Functions for computing the value-at-risk in compound Poisson models.
The implementation comprises functions for modeling loss frequencies and loss severities with plain, mixed (Frigessi et al. (2012) <doi:10.1023/A:1024072610684>) or spliced distributions using Maximum Likelihood estimation and Bayesian approaches (Ergashev et al. (2013) <doi:10.21314/JOP.2013.131>).
In particular, the parametrization of tail distributions includes the fitting of Tukey-type distributions (Kuo and Headrick (2014) <doi:10.1155/2014/645823>). Furthermore, the package contains the modeling of bivariate dependencies between loss severities and frequencies, Monte Carlo simulation for total loss estimation as well as a closed-form approximation based on Degen (2010) <doi:10.21314/JOP.2010.084> to determine the value-at-risk.

License GPL-3

Imports VineCopula, tea, actuar, truncnorm, ReIns, MASS, pracma, evmix

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2021-09-08 16:00:11 UTC

R topics documented:

OpVaR-package 2

buildFreqdist . . . 3
buildMixingSevdist . . . 4
buildPlainSevdist . . . 5
buildSplicedSevdist . . . 6
dsevdist . . . 7
fitDependency . . . 8
fitFreqdist . . . 9
fitMixing . . . 10
fitPlain . . . 11
fitSpliced . . . 12
fitSplicedBayes . . . 13
fitSplicedBestFit . . . 15
fitThreshold . . . 16
fitWeights . . . 18
gh . . . 19
goftest . . . 20
gpd . . . 22
lossdat . . . 23
mcSim . . . 24
Mixing . . . 25
qqplot.sevdist . . . 26
sla . . . 27

Index **29**

`OpVaR-package` *Statistical Methods for Modelling Operational Risk*

Description

Functions for computing the value-at-risk in compound Poisson models. The implementation comprises functions for modeling loss frequencies and loss severities with plain, mixed (Frigessi et al. (2012) <doi:10.1023/A:1024072610684>) or spliced distributions using Maximum Likelihood estimation and Bayesian approaches (Ergashev et al. (2013) <doi:10.21314/JOP.2013.131>). In particular, the parametrization of tail distributions includes the fitting of Tukey-type distributions (Kuo and Headrick (2014) <doi:10.1155/2014/645823>). Furthermore, the package contains the modeling of bivariate dependencies between loss severities and frequencies, Monte Carlo simulation for total loss estimation as well as a closed-form approximation based on Degen (2010) <doi:10.21314/JOP.2010.084> to determine the value-at-risk.

Author(s)

Christina Zou [aut,cre], Marius Pfeuffer [aut], Matthias Fischer [aut], Nina Buoni [ctb], Kristina Dehler [ctb], Nicole Derfuss [ctb], Benedikt Graswald [ctb], Linda Moestel [ctb], Jixuan Wang [ctb], Leonie Wicht [ctb]

Maintainer: Christina Zou <christina.zou@maths.ox.ac.uk>

References

Degen, M. (2010): The calculation of minimum regulatory capital using single-loss approximations. The Journal of Operational Risk, 5(4), 3.

Dehler, K. (2017): Bayesianische Methoden im operationellen Risiko. Master's Thesis, Friedrich-Alexander-University Erlangen-Nuremberg.

Ergashev, B. et al. (2013): A Bayesian Approach to Extreme Value Estimation in Operational Risk Modeling. Journal of Operational Risk 8(4):55-81

Frigessi, A. et al. (2002): A Dynamic Mixture Model for Unsupervised Tail Estimation Without Threshold Selection. Extremes 5(3):219-235

Kuo, T. C. and Headrick, T. C. (2014): Simulating Univariate and Multivariate Tukey g-and-h Distributions Based on the Method of Percentiles. ISRN Probability and Statistics.

Pfaelzner, F. (2017): Einsatz von Tukey-type Verteilungen bei der Quantifizierung von operationellen Risiken. Master's Thesis, Friedrich-Alexander-University Erlangen-Nuremberg.

Reynkens, T. et al. (2017): Modelling Censored Losses Using Splicing: a global fit strategy with mixed Erlang and Extreme Value Distributions. Insurance: Mathematics and Economics 77:67-77

Tukey, J. W. (1960): The Practical Relationship between the Common Transformations of Counts of Amounts. Technical Report 36, Princeton University Statistical Techniques Research Group, Princeton.

`buildFreqdist` *Building a freqdist object*

Description

Building a freqdist object

Usage

```
buildFreqdist(freq.distr, freq.param)
```

Arguments

`freq.distr`	A string giving the name of the distribution. Distributions "pois" and "nbinom" are recognised.
`freq.param`	List of numerical values giving the parameters for the given distribution.

Details

A freqdist object with type 'plain' is generated, i.e. for a given cell the losses occur with frequency given by distribution 'distr' with parameters given by 'param'.

Author(s)

Christina Zou

See Also

The sevdist objects according to different types are built via `buildPlainSevdist`, `buildSplicedSevdist` and `buildMixingSevdist`. For given data, a freqdist object can be fitted using `fitFreqdist`.

Examples

```
# Poisson distributed frequency with lambda = 50

freqdist1 = buildFreqdist("pois", 50)
```

---

buildMixingSevdist *Building a dynamic mixture model as a sevdist object*

Description

Building a dynamic mixture model as a sevdist object

Usage

```
buildMixingSevdist(body.distr, body.param, tail.distr, tail.param, mixing.param)
```

Arguments

`body.distr`	A string giving the name of the body distribution.
`body.param`	Vector of the parameters for the given body distribution.
`tail.distr`	A string giving the name of the tail distribution.
`tail.param`	Vector of the parameters for the given tail distribution. Distributions "lgamma", "weibull", "gpd", "gh" are recognised.
`mixing.param`	Vector of the parameters for the given mixing distribution.

Details

A sevdist object with type 'mixing' is generated, i.e. the distribution of the severity is a dynamic mixture of the distributions 'body.distr' with parameters given by 'body.param' and 'tail.distr' with parameters given by 'tail.param' using the cdf of a mixing distribution 'mixing.distr' with parameters 'mixing.param' according to Frigessi et al. (2002).

The density $f(x)$ for a dynamic mixture model with body distribution having density g and tail distribution having density h and cumulative distribution function p of the mixing distribution is given below:

$$f(x) = C * [p(x) * g(x) + (1 - p(x)) * h(x)]$$

Author(s)

Christina Zou

References

Frigessi, A., Haug, O., & Rue, H. (2002). A dynamic mixture model for unsupervised tail estimation without threshold selection. Extremes, 5(3), 219-235.

Examples

```
# Create mixing sevdist object
sevdist = buildMixingSevdist("weibull", c(2,6), "gpd", c(0,4,-0.5), c(8,.8))
# Plot pdf
plot(sevdist)
```

`buildPlainSevdist` *Building a sevdist object with a plain distribution*

Description

Building a sevdist object with a plain distribution

Usage

```
buildPlainSevdist(distr, param)
```

Arguments

`distr`	A string giving the name of the distribution.
`param`	The parameters for the given distribution

Details

A sevdist object with type 'plain' is generated, i.e. the severity has distribution 'distr' with parameters given by 'param'.

Author(s)

Christina Zou

See Also

Other sevdist objects with type 'spliced' and 'mixing' are built via `buildSplicedSevdist` and `buildMixingSevdist`.

Examples

```
# Log-gamma distributed severity with shape = 2.2 and rate = 1.2

sevdist1 = buildPlainSevdist("lgamma", c(2.2, 1.2))
plot(sevdist1)
```

buildSplicedSevdist *Building a sevdist object with a spliced distribution*

Description

Building a sevdist object with a spliced distribution

Usage

```
buildSplicedSevdist(body.distr, body.param, tail.distr, tail.param, thresh, weight)
```

Arguments

`body.distr`	A string giving the name of the body distribution.
`body.param`	Vector of the parameters for the given body distribution.
`tail.distr`	A string giving the name of the tail distribution. Distributions "lgamma", "weibull", "gpd", "gh" are recognised.
`tail.param`	Vector of the parameters for the given tail distribution.
`thresh`	Numeric value giving the threshold of the distribution, the severity follows a truncated 'body.distr' distribution below the threshold and a truncated 'tail.distr' distribution above the threshold.
`weight`	Numeric value in [0,1] giving the probability that the severity distribution takes values below the threshold.

Details

A sevdist object with type 'spliced' is generated, i.e. the severity has distribution 'body.distr' in the body with parameters given by 'body.param' and distribution 'tail.distr' in the tail with parameters given by 'tail.param'.

The density $f(x)$ for a spliced distribution with threshold τ, weights w, body distribution with density g and cumulative distribution function G and tail distribution with density h and cumulative distribution function H is given as below:

If $x \leq \tau$: $f(x) = w * g(x)/G(\tau)$

If $x > \tau$: $f(x) = (1 - w) * h(x)/(1 - H(\tau))$

Author(s)

Christina Zou

See Also

Other sevdist objects with type 'plain' and 'mixing' are built via `buildPlainSevdist` and `buildMixingSevdist`.

Examples

```
# Spliced distributed severity with gamma distributed body with shape = 1.23, rate = 0.12
 # and GPD distributed tail with shape = 716 and scale = 0.1 and threshold = 2000.
 # The weight for the body is 0.8.

sevdist1 = buildSplicedSevdist("gamma", c(2.5, 0.012), "gpd", c(2000, 716, 0.1), 2000, 0.8)
 plot(sevdist1)
```

---

dsevdist *Evaluating Plain, Spliced or Mixing Severity Distributions*

Description

Functions for evaluating pdf, cdf, quantile function and random number generation given a loss severity model (sevdist object)

Usage

```
dsevdist(x, sevdist)
```

Arguments

`x`

`sevdist`

Author(s)

Marius Pfeuffer

Examples

```
## The function is currently defined as
function (x, sevdist)
{
    if (sevdist$type == "plain")
        return(sevdist$par[[1]](x))
    if (sevdist$type == "spliced")
        return(dspliced(x, sevdist))
    if (sevdist$type == "mixing")
        return(dmixing(x, sevdist))
  }
```

fitDependency *Function for fitting bivariate Copulas*

Description

Function for fitting bivariate dependency between loss frequencies and severities.

Usage

```
fitDependency(cell, copula_family, method = "mle")
```

Arguments

`cell`	a lossdat cell
`copula_family`	0: independence, 1: Gaussian, 2: t, 3: Clayton, 4: Gumbel, 5: Frank, 6: Joe, see Vine Copula Package ?BiCop encoding
`method`	as in fitCopula of the copula package. The method "itau" should not be used as dependencies between a continuous and a discrete random variable are modeled.

Value

A BiCop object

Author(s)

Marius Pfeuffer

References

Schepsmeier et al.: VineCopula package, CRAN

Examples

```
### Fit Joe Copula
data(lossdat)
fitDependency(lossdat[[1]],6)
```

`fitFreqdist` *Fitting the frequency distribution*

Description

Fitting the frequency distribution of operational losses

Usage

```
fitFreqdist(cell, distr)
```

Arguments

`cell`	A data frame giving the losses (cell$Loss) and the user-defined period (cell$Period) in which the loss occured.
`distr`	The frequency distribution: either "pois" or "nbinom"

Details

First, the number of losses per period is counted. With the results a Poisson or Negative Binomial distribution is fitted for the number of losses per period using a Maximum Likelihood estimator.

Value

Returns a freqdist object with the given distribution fitted to the loss frequency.

Author(s)

Christina Zou

See Also

To build a freqdist object independently, use `buildFreqdist`.

Examples

```
data(lossdat)
fitFreqdist(lossdat[[1]],"pois")
fitFreqdist(lossdat[[2]],"nbinom")
```

fitMixing *Maximum Likelihood Estimation*

Description

Maximum Likelihood Estimation for the dynamic weighted mixture model of Frigessi et al., 2002

Usage

```
fitMixing(cell, body, tail, method="L-BFGS-B", c_location0=0.75, c_scale0=2)
```

Arguments

`cell`	lossdat cell
`body`	body distribution, either gamma, lnorm or weibull
`tail`	tail distribution, either gamma, lnorm, weibull or gpd
`method`	optimization method, default is "L-BFGS-B"
`c_location0`	empirical quantile of loss severity data used for initialization of Cauchy location parameter in optimization: quantile(cell$Loss,c_location0)
`c_scale0`	scaling factor for empirical standard deviation used for initialization of Cauchy scale parameter in optimization: sd(cell$Loss)/c_scale0

Details

Body and tail parameters are initialized by method of moments estimators. Cauchy location is initialized by empirical 70

Value

Returns a sevdist object of type 'mixing' with the given body and tail distributions fitted to the loss data.

Author(s)

Marius Pfeuffer

References

Frigessi et al. *A Dynamic Mixture Model for Unsupervised Tail Estimation without Threshold Selection*, Extremes 5(3):219-235, 2003

Examples

```
data(lossdat)
sev=fitMixing(lossdat[[1]],"weibull","gpd")
sev
plot(sev,5000)
```

`fitPlain` *Fit plain distribution models*

Description

Function for estimating the parameters of plain (non-spliced, non-mixing) distribution models

Usage

```
fitPlain(cell, family)
```

Arguments

`cell`	list containing the data in the component cell$Loss
`family`	distribution family

Value

Returns a sevdist object of type 'plain' with the given distribution fitted to the loss data.

Author(s)

Marius Pfeuffer

Examples

```
data(lossdat)
fitPlain(lossdat[[1]],"lnorm")
```

`fitSpliced`	*Estimation of the threshold, the body and the tail parameters for a spliced distribution*

Description

Given a dataset with a chosen distribution for the data in the body and another distribution in the tail, the threshold, the parameters of the body and the tail distributions and the weights are estimated.

Usage

```
fitSpliced(cell, body, tail, method, thresh = NULL)

fitSplicedPar(cell, thresh, body, tail)
```

Arguments

`cell`	List containing the data in the component cell$Loss
`body`	Distribution in the body. Can be chosen between "gamma", "lnorm", "weibull" or "erlang"
`tail`	Distribution in the tail. Can be chosen between "gpd", "gamma", "lnorm", "weibull" or "gh"
`method`	Method for the threshold estimation. In case of a GPD tail, it can be chosen between "BestFit", "Fixed", "dAMSE", "danielsson", "DK", "hall", "Himp", "HW", "mindist" or "RT"
`thresh`	Predetermined threshold quantile (if method="Fixed")

Details

In the spliced model, the distribution of the data is spliced into two distributions, one in the body and another in the tail. The density $f(x)$ for a spliced distribution with threshold τ, weights w, body distribution with density g and cumulative distribution function G and tail distribution with density h and cumulative distribution function H is given as below:

If $x \leq \tau$: $f(x) = w * g(x)/G(\tau)$

If $x > \tau$: $f(x) = (1 - w) * h(x)/(1 - H(\tau))$

The weight w is required to normalise the density function. The estimation of the spliced distribution consists of three steps. In the first step, the threshold for the selected body and tail distribution is estimated. For further details on the methods for the threshold estimation, see fitThreshold. In the second step, the parameters of the body and the tail distribution are obtained by maximum likelihood estimation. For spliced distributions, truncated body distributions are fitted to truncated data. For the estimation of the parameters of the tail distribution, only the data points above the threshold are used. In the last step, the weight w is obtained.

Value

Returns a sevdist object of type 'spliced' with the given body and tail distributions fitted to the loss data.

Author(s)

Christina Zou, Leonie Wicht

See Also

fitSplicedBestFit, fitThreshold

Examples

```
data(lossdat)
fitSpliced(lossdat[[3]],"gamma","gpd",method="Fixed",thresh=2000)
```

`fitSplicedBayes` *Parameter Estimation for Spliced Distributions*

Description

MCMC based parameter estimation for spliced distribution functions, where the body is modeled by a Weibull, gamma or log-normal distribution or Kernel density estimation for a log-normal distribution and the tail is fitted with a GPD.

Usage

```
fitSplicedBayesWeibullGPD(cell, prior, burnin=10, niter=100,
proposal_scale=evmix::fweibullgpd(cell,method="Nelder-Mead")$se,
start=evmix::fweibullgpd(cell,method="Nelder-Mead")$optim$par)

fitSplicedBayesGammaGPD(cell, prior, burnin=10, niter=100,
proposal_scale=evmix::fgammagpd(cell,method="Nelder-Mead")$se,
start=evmix::fgammagpd(cell,method="Nelder-Mead")$optim$par)

fitSplicedBayesLognormGPD(cell, prior, burnin=10, niter=100,
proposal_scale=evmix::flognormgpd(cell,method="Nelder-Mead")$se,
start=evmix::flognormgpd(cell,method="Nelder-Mead")$optim$par)

fitSplicedBayesKDEGPD(cell, prior, burnin=10, niter=100,
proposal_scale=evmix::flognormgpd(cell,method="Nelder-Mead")$se,
start=evmix::flognormgpd(cell,method="Nelder-Mead")$optim$par)
```

Arguments

`cell`	database for the parameter estimation
`prior`	parameters of the particular prior distribution
`burnin`	length of the burn-in-phase, standard value = 10
`niter`	MCMC sample size, standard value = 100
`proposal_scale`	respective scale values used for determining the prior distribution for each parameter
`start`	starting values (shapes) for the MH-algorithm steps

Details

The input contains the dataset, whose compound distribution should be fitted with the function. The second part of the input is a list with the parameters of the prior distribution. It is necessary to indicate the `prior` as a list containing the full parameter vector in the order `bodypar_1=c()`,`bodypar_2=c()`, `threshold=c()`, `beta=c()`, `xi=c()`, where `bodypar_1` and `bodypar_2` are the parameters of the body (Weibull: wshape and wscale as shape and scale parameter, log-norm: mu and sigma, gamma: gshape and gscale as shape and scale parameter).`threshold` is the threshold seperating the distributions.`beta` and `xi` are the scale and shape parameters of the generalized Pareto distribution. Using the MH-algorithm the true parameters of the body and tail distribution should be approximated.

Value

Sevdist object containing the parameters of the target distribution.

Author(s)

Kristina Dehler, Nicole Derfuss

References

Behrens, C. N.; Lopes, H. F.; Gamerman, D. (2004). Bayesian analysis of extreme events with threshold estimation. Statistical Modelling 4, 227-244.

Dehler, K. (2017). Bayesianische Methoden im operationellen Risiko. Master’s thesis.

Ergashev, B.; Mittnik, S.; Sekeris, E. (2013). A Bayesian Approach to Extreme Value Estimation in Operational Risk Modeling. The Journal of Operational Risk 8(4), 55-81.

MacDonald, A.; Scarrott, C.J.; Lee, D.; Darlow, B.; Reale, M.; Russell, G. (2011). A flexible extreme value mixture model. Computational Statistics and Data Analysis 55(6), 2137-2157.

See Also

Read `buildSplicedSevdist` for further information about building sevdist objects with type "spliced". The initial parameter values are fitted using the methods from the package ’evmix’, i.e. for the WeibullGPD the function `fweibullgpd`. For the algorithm also the package ’truncnorm’ is required.

Examples

```
### Example for estimating the parameters of the spliced Weibull-GPD

## Not run:

data("lossdat")
data=lossdat[[1]]$Loss

## starting values - method of moments

# Weibull distribution (shape, scale)
model <- function(x) c(F1 = x[2]*gamma(1+1/x[1])-mean(data),
F2 = x[2]^2*(gamma(1+2/x[1])-(gamma(1+1/x[1]))^2)-sd(data)^2)
weibullpar <- rootSolve::multiroot(f = model,start = c((sd(data)/mean(data))^(-1.086),
mean(data)/(gamma(1+1/((sd(data)/mean(data))^(-1.086))))))$root

# generalized Pareto distribution (tresh, beta, xi)
thresh=quantile(data,.9) # threshold
data2=data[which(data>thresh)]
model=function(x) c(F1=thresh+x[1]/(1-x[2])-mean(data2),
F2=x[1]^2/(1-x[2])^2/(1-2*x[2])-sd(data2)^2)
gpdpar=c(thresh,rootSolve::multiroot(f=model,start=c(mean(data2),1/mean(data2)))$root)

## parameters of the prior distribution

prior <- list(xi = c(gpdpar[3],sd(data)),tau = c(quantile(data,.9),sd(data)/10),
beta = c(gpdpar[2],sd(data)),wscale = c(weibullpar[2],sd(data)),
wshape = c(weibullpar[1],sd(data)))

## estimation of (shape, scale, thresh, beta, xi)

fitSplicedBayesWeibullGPD(data, prior = prior, proposal_scale = evmix::fweibullgpd(data,
method = "Nelder-Mead", pvector = c(weibullpar[1], weibullpar[2], gpdpar[1],
gpdpar[2], gpdpar[3]))$se, start = evmix::fweibullgpd(data,
method = "Nelder-Mead", pvector = c(weibullpar[1], weibullpar[2], gpdpar[1],
gpdpar[2], gpdpar[3]))$optim$par)

## End(Not run)
```

fitSplicedBestFit *Fitting a spliced distribution over a given data set*

Description

A spliced distribution is fitted based on given loss data and required distributions for the body and the tail using a best-fit method. The function returns a sevdist object with type "spliced".

Usage

```
fitSplicedBestFit(cell, body, tail, thresh0 = 0.7, thresh.max = 0.98)
```

Arguments

`cell`	List containing the loss data in cell$Loss.
`body`	Character string giving the name of the distribution in the body. Can be chosen between "gamma", "lnorm", "weibull" or "erlang".
`tail`	Character string giving the name of the distribution in the tail. Can be chosen between "gpd", "gamma", "lnorm", "weibull" or "gh".
`thresh0`	Initial value for the quantile of the threshold between body and tail
`thresh.max`	Terminal value for the quantile of the threshold between body and tail

Details

Starting with a spliced distribution with threshold given by the quantile thresh0, the respective parameters of the given body and tail distribution are fitted, as well as the weights of both parts. A list of possible thresholds is given by all loss values between all quantiles between thresh0 and thresh.max. For each of these thresholds a spliced distribution will be fitted and then the best fitting threshold with the corresponding parameters will be chosen with a Kolmogorow-Smirnov test and the result is returned in a sevdist object with type "spliced".

Author(s)

Christina Zou

See Also

fitSplicedPar, fitThreshold, fitSpliced

Examples

```
data(lossdat)

param<-fitSplicedBestFit(lossdat[[1]], "lnorm", "gpd")
param[[1]] ## Parameters of the body distribution
param[[2]] ## Parameters of the tail distribution
```

`fitThreshold` *Threshold estimation for spliced distribution*

Description

Given a dataset with a chosen distribution for the data in the body and another distribution in the tail, the threshold is estimated.

Usage

```
fitThreshold(cell, body, tail, method)
```

Arguments

`cell`	list containing the data in the component Loss
`body`	list containing the data in the component Loss
`tail`	list containing the data in the component Loss
`method`	Method for the threshold estimation. In case of a GPD tail, it can be chosen between "dAMSE", "danielsson", "DK", "GH", "gomes", "hall", "Himp", "HW", "mindist", "RT"

Details

## GPD tail In case of a Generalized Pareto Function in the tail, there are several methods to estimate the threshold which are based on extreme value statistics. Most of them are based on the Asymptotic Mean Squared Error (AMSE) approach. where the asymptotic squared error

$$AsyE((Hill(N,k) - gamma)^2)$$

between the true inverse tail index gamma and the Hill-estimator is minimized: "dAMSE"", "danielsson"", "DK", "GH"", gomes", "hall", "Himp" and "HW". For further details, see tea. "RT" is a heuristic approach where the smallest order statistic k that minimizes the expression $1/ksum_i = 1^k i^b eta|gamma_i - median(gamma_1, ..., gamma_k)|$ is chosen for the threshold. "mindist" minimizes the distance between the largest upper order statistics (the empirical tail) and the theoretical tail of a Pareto distribution. For further details, see tea.

Value

Returns a numeric value giving threshold fitted according the given method.

Author(s)

Christina Zou, Leonie Wicht

References

# GPD Tail:

Caeiro, F. and Gomes, M.I. (2014) On the bootstrap methodology for the estimation of the tail sample fraction. Proceedings of COMPSTAT, 545-552.

Caeiro, J. and Gomes, M.I. (2016) Threshold selection in extreme value analysis. Extreme Value Modeling and Risk Analysis:Methids and Applications, 69-86

Danielsson, J. and Ergun, L.M. and de Haan, L. and de Vries, C.G. (2016) Tail Index Estimation: Quantile Driven Threshold Selection

Danielsson, J. and Haan, L. and Peng, L. and Vries, C.G. (2001) Using a bootstrap method to choose the sample fraction in tail index estimation. Journal of Multivariate analysis, 2, 226-248.

Drees, H. and Kaufmann, E. (1998) Selecting the optimal sample fraction in univariate extreme value estimation. Stochastic Processes and their Applications, 75(2), 149-172.

Gomes, M.I. and Figueiredo, F. and Neves, M.M. (2012) Adaptive estimation of heavy right tails: resampling-based methods in action. Extremes, 15, 463-489

Guillou, A. and Hall, P. (2001) A Diagnostic for Selecting the Threshold in Extreme Value Analysis Journal of the Royal Statistical Society, 63(2), 293-305

Hall, P. (1990) Using the Bootstrap to Estimate Mean Squared Error and Select Smoothing Parameter in Nonparametric Problems. Journal of Multivariate Analysis, 32, 177-203

Hall, P. and Welsh, A.H. (1985) Adaptive estimates of parameters of regular variation. The Annals of Statistics, 13(1), 331-341.

Reiss, R.-D. and Thomas, M. (2007) Statistical Analysis of Extreme Values: With Applications to Insurance, Finance, Hydrology and Other Fields. Birkhauser, Boston.

Examples

```
data(lossdat)
fitThreshold(lossdat[[1]], method="dAMSE")
```

fitWeights *Fitting the weights of the body and the tail for a spliced distribution*

Description

Fitting the weights of the spliced distribution.

Usage

```
fitWeights(cell, thresh)
```

Arguments

`cell`	List containing the data in the component Loss
`thresh`	Threshold between the body and the tail.

Details

Given the threshold t, this function fits the weight on the body distribution as part of fitting the parameters of the spliced distribution. The weights depends only on the number of data points below and above the threshold, not on the distribution functions in the body and the tail.

Author(s)

Christina Zou, Leonie Wicht

See Also

fitThreshold, fitSpliced

Examples

```
data(lossdat)
thresh<-quantile(lossdat[[1]]$Loss, 0.9)
w<-fitWeights(lossdat, thresh)
```

`gh` *Tukey's gh Distribution*

Description

Density, distribution function, quantile function, and random generation for Tukey's gh distribution.

Usage

```
dgh(x, A, B, g, h, log = FALSE)
pgh(q, A, B, g, h, log.p = FALSE)
qgh(q, A, B, g, h, log.p =FALSE)
rgh(n, A, B, g, h)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations.
`A`	location parameter.
`B`	scale parameter, has to be positive.
`g`	skewness parameter.
`h`	kurtosis parameter.
`log, log.p`	logical; if TRUE, probabilities p are given as log(p)

Details

Tukey's gh distribution with location parameter A, scale parameter B, skewness parameter g, and kurtosis parameter h is obtained by transforming a standard normal variable X by

$$T(X) = A + Bexp(h/2X^2)(exp(gX) - 1)/g$$

if g is not equal to zero, and else by

$$T(X) = A + Bexp(h/2X^2)X.$$

Value

dgh gives the density, pgh gives the distribution function, qgh gives the quantile function, and rgh generates random deviates.

The length of the result is determined by n for rgh, and is the length of the numerical arguments for the other function.

Author(s)

Linda Moestel

References

Tukey, J. W. (1960): The Practical Relationship between the Common Transformations of Counts of Amounts. Technical Report 36, Princeton University Statistical Techniques Research Group, Princeton.

Klein, I. and Fischer, M. (2002): Symmetrical gh-transformed Distributions. in S. Mittnek and I. Klein: *Contributions to Modern Econometrics, Kluwer Academic Publishers.*

Pfaelzner, F. (2017): Einsatz von Tukey-type Verteilungen bei der Quantifizierung von operationellen Risiken. MMasterthesis Friedrich-Alexander-University Erlangen-Nueremberg.

Examples

```
##Parameters  for a gh distribution
  A=500
  B=3
  g=0.2
  h=0.5

  hist(rgh(n=1000,A,B,g,h))
  curve(dgh(x,A,B,g,h),480,520)
  curve(pgh(x,A,B,g,h),480,520)
  curve(qgh(x,A,B,g,h),0,1)
```

`goftest` *Goodness of fit tests for severity distributions*

Description

Testing the goodness-of-fit of the distributions given in a sevdist or a freqdist object to the loss data.

Usage

```
goftest(cell, object)
```

Arguments

`cell`	A data frame giving the losses (cell$Loss) and the user-defined period (cell$Period) in which the loss occured.
`object`	A loss severity model (sevdist object) or a loss frequency model (freqdist object)

Details

If object is of type sevdist, then the Kolmogorov-Smirnov test is performed on the loss data of a single cell and the respected fitted distribution given by the sevdist object.

If object is of type freqdist, then a χ^2 goodness-of-fit test will be performed.

Value

If object is of type sevdist, then the return is the test results from the Kolmogorow-Smirnov test.

If object is of type freqdist, then the functions returns the result of the χ^2 test.

Refer with $p.value to the p-values and $statistic to the test statistic of each of the tests.

Author(s)

Marius Pfeuffer, Christina Zou

References

Birnbaum, Z. W., and Fred H. Tingey. "One-sided confidence contours for probability distribution functions". The Annals of Mathematical Statistics 22.4 (1951): 592-596.

Conover, William Jay, and William Jay Conover. "Practical nonparametric statistics" (1980).

Durbin, James. "Distribution theory for tests based on the sample distribution function". Society for Industrial and Applied Mathematics, 1973.

See Also

We use the `ks.test` and `chisq.test` from `stats`.

Examples

```
data("lossdat")
opriskmodel = list()
opriskmodel[[1]] = list()
opriskmodel[[1]]$freqdist = fitFreqdist(lossdat[[1]],"pois")
opriskmodel[[1]]$sevdist = fitPlain(lossdat[[1]],"gamma")

# perform test on the sevdist object
goftest(lossdat[[1]], opriskmodel[[1]]$sevdist)

# show result p-value and test statistics the Kolmogorow-Smirnov test
test = goftest(lossdat[[1]], opriskmodel[[1]]$sevdist)
test$p.value
test$statistic
```

```
# perform test on the freqdist object
goftest(lossdat[[1]], opriskmodel[[1]]$freqdist)

# the p-value is given by
goftest(lossdat[[1]], opriskmodel[[1]]$freqdist)$p.value
```

gpd *Generalized Pareto Distribution*

Description

Density, distribution function, quantile function, and random generation for generalized Pareto distribution.

Usage

```
dgpd(x, loc, scale, shape, log=FALSE)
pgpd(q, loc, scale, shape)
qgpd(p, loc, scale, shape)
rgpd(n, loc, scale, shape)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations.
`loc`	location parameter.
`scale`	scale parameter.
`shape`	shape parameter.
`log`	logical; logarithm of output

Details

Generalized Pareto Distribution with density

$$f(x) = 1/scale * (1 + shape * (x - loc)/scale)^(- 1/shape - 1)$$

Value

dgpd gives the density, pgpd gives the distribution function, qgpd gives the quantile function, and rgpd generates random deviates.

Examples

```
dgpd(110,100,2,4) # 0.01112233
```

lossdat *Example loss data set*

Description

A minimal loss data set comprising 4 cells (referring to a 2x2 business line/event type operational risk setting).

Usage

```
data("lossdat")
```

Format

The format is: List of 4 $:'data.frame': 1965 obs. of 3 variables: ..$ Loss : num [1:1965] 1877 1807 918 1480 1218 ... ..$ Period: num [1:1965] 1 1 1 1 1 1 1 1 1 1 ... ..$ Date : Date[1:1965], format: "2016-12-30" "2016-12-25" "2016-12-22" ... $:'data.frame': 2025 obs. of 3 variables: ..$ Loss : num [1:2025] 181 610 961 1312 259 ... ..$ Period: num [1:2025] 1 1 1 1 1 1 1 1 1 1 ... ..$ Date : Date[1:2025], format: "2016-12-30" "2016-12-30" "2016-12-28" ... $:'data.frame': 1995 obs. of 3 variables: ..$ Loss : num [1:1995] 1334 1067 1068 714 1590 ... ..$ Period: num [1:1995] 1 1 1 1 1 1 1 1 1 1 ... ..$ Date : Date[1:1995], format: "2016-12-31" "2016-12-28" "2016-12-26" ... $:'data.frame': 1941 obs. of 3 variables: ..$ Loss : num [1:1941] 650 847 242 513 264 ... ..$ Period: num [1:1941] 1 1 1 1 1 1 1 1 1 1 ... ..$ Date : Date[1:1941], format: "2016-12-31" "2016-12-28" "2016-12-22" ...

Details

4 cells with Losses, Time Stamp and Discrete-Time Period (referring to e.g., a quarterly or annual basis).

Source

Hypothetical, simulated example with bivariate dependencies between loss frequencies and severities

Examples

```
data(lossdat)
summary(lossdat)
```

`mcSim` *Monte Carlo Simulation from opriskmodel objects for total loss estimation*

Description

Function for conducting Monte Carlo Simulation of complete opriskmodel objects (list of cells with (1) frequency model, (2) severity model and (3) dependencymodel)

Usage

```
mcSim(opriskmodel, n_sim, verbose=TRUE)
VaR(mc_out, alpha)
```

Arguments

`opriskmodel`	an opriskmodel object
`n_sim`	number of simulations
`mc_out`	Monte Carlo simulation output
`alpha`	significance level for quantile/value-at-risk
`verbose`	verbose mode

Value

A mcsim object, which can be further processed by the VaR function to estimate empirical quantiles as value-at-risk measure

Author(s)

Marius Pfeuffer

See Also

`sla`

Examples

```
### Load Example Data Set
data(lossdat)

### Estimation of Complete Risk Model
opriskmodel1=list()
for(i in 1:length(lossdat)){
  opriskmodel1[[i]]=list()
  opriskmodel1[[i]]$freqdist=fitFreqdist(lossdat[[i]],"pois")
  opriskmodel1[[i]]$sevdist=fitPlain(lossdat[[i]],"lnorm")
}
```

```
### Cell 1: Gumbel Copula, Cell 2: Independence, Cell 3: Frank Copula, Cell 4: Independence
opriskmodel1[[1]]$dependency=fitDependency(lossdat[[1]],6)
opriskmodel1[[3]]$dependency=fitDependency(lossdat[[3]],4)

### Monte Carlo Simulation
mc_out=mcSim(opriskmodel1,100)

### Evaluation of 95
VaR(mc_out,.95)
sla(opriskmodel1,.95)

### Monte Carlo Simulation
mc_out=mcSim(opriskmodel1,100)

### Evaluation of 95% Value-at-Risk
VaR(mc_out,.95)
```

Mixing *Extended Dynamic Weighted Mixture Model*

Description

Implementation of the dynamic weighted mixture model of Frigessi et al., 2001

Usage

```
dmixing(x,sevdist,log=FALSE)
pmixing(q,sevdist,lower.tail=TRUE)
qmixing(p,sevdist,lower.tail=TRUE)
rmixing(n,sevdist)
```

Arguments

`x,q`	vector of quantiles
`p`	vector of probabilities.
`n`	positive integer, number of observations
`sevdist`	sevdist object
`log`	logical; if TRUE, probabilities p are given as log(p).
`lower.tail`	logical; if TRUE (default), probabilities are P[X <= x] otherwise, P[X > x].

Details

The dynamic mixture pdf is given by $f(x) = 1/r * ((1 - W(x)) * g(x) + W(x) * h(x))$, where W(x) denotes a Cauchy cdf, g and h body and tail pdfs, r is a normalizing constant.

Author(s)

Marius Pfeuffer

References

A. Frigessi et al.: *A Dynamic Mixture Model for Unsupervised Tail Estimation without Threshold Selection*, Extremes 5(3):219-235, 2002

Y. Hu: *User's Guide for evmix Package in R.*, Working Paper, 2013

Examples

```
### Create mixing sevdist object
sevdist=buildMixingSevdist("weibull", c(2,6), "gpd", c(0,4,-0.5), c(8,.8))
### Evaluate pdf
dmixing(1,sevdist)
```

`qqplot.sevdist` *Density plot and Q-Q plot for plain, mixing, and spliced distributions*

Description

Given a dataset and a fitted sevdist object of type plain, mixing, or spliced, a density plot or qqplot is created to help determine the best distribution for modelling loss severity.

Usage

```
qqplot.sevdist(cell, sevdist, probabilities = NULL)
```

Arguments

`sevdist`	Fitted sevdist object
`cell`	List containing the data in the component cell$Loss
`probabilities`	Vector containing probabilities for desired quantiles. If no vector is given, default probabilities of 0.001, 0.002, ..., 0.99 will be used.

Value

Returns a density plot of the fitted sevdist object or a qqplot of the theoretical loss quantiles (sevdist object) vs. the experimental loss quantiles (loss data).

Author(s)

Nina Buoni, Marius Pfeuffer, Christina Zou

Examples

```
data(lossdat)
sev1 <- fitPlain(lossdat[[1]], "weibull")
plot(sev1)

sev2 <- fitSpliced(lossdat[[2]], "gamma", "gpd", method = "dAMSE")
qqplot.sevdist(lossdat[[2]], sev2)
```

---

sla *Single-Loss Approximation for Operational Value at Risk*

Description

Given the object opriskmodel using the single-loss approximation a list of alpha-quantiles is created for every cell in opriskmodel.

Usage

```
sla(opriskmodel, alpha, xi_low = 0.8, xi_high = 1.2, plot = FALSE)
```

Arguments

`opriskmodel`	Object containing the parameters for the severities and the frequencies
`alpha`	Level of the quantile for the total loss-process of the single-loss-approximation
`xi_low`	Lower interpolation-point for the spline-function, standard value = 0.8
`xi_high`	Upper interpolation-point for the spline-function, standard value = 1.2
`plot`	Plot of the interpolated correction term if xi is between xi_low and xi_high, standard value = FALSE

Details

In the first step the tailindex of the severity distribution is determined. If it does not lie in the critical zone, i.e., not between xi_low and xi_high, the closed-form single-loss approximation from Degen is used. If the tailindex lies in the critical zone, then the R-function splinefun with method "hyman" is used to create a spline with given data points xi lying in [xi_low - 0.2, xi_low] and [xi_high, xi_high + 0.2] using the value at xi = 1 as an anchor. A plot of the interpolation is provided such that user-defined adjustment of xi_low and xi_high is possible.

Author(s)

Benedikt Graswald, Jixuan Wang, Christina Zou

References

Bocker, Klaus, and Claudia Kluppelberg. "Operational VaR: a closed-form approximation." Risk-London-Rsik Magazine Limited- 18.12 (2005): 90.

Degen, Matthias. "The calculation of minimum regulatory capital using single-loss approximations." The Journal of Operational Risk 5.4 (2010): 3.

Examples

```
#Example: SLA for the spliced log-gamma gpd model (tail-index = 0.014, no interpolation required)
 opriskmodel = list()
 opriskmodel[[1]] = list()
 opriskmodel[[1]]$sevdist = buildSplicedSevdist("lgamma", c(1.23, 0.012),
                                               "gpd", c(200, 716, 0.014), 2000, 0.8)
 opriskmodel[[1]]$freqdist = buildFreqdist("pois", 50)

#Example: SLA for the spliced log-gamma gpd model (tail-index = 0.9, interpolation performed)
 opriskmodel[[2]] = list()
 opriskmodel[[2]]$sevdist = buildSplicedSevdist("lgamma", c(1.23, 0.012),
                                               "gpd", c(200, 716, 0.9), 2000, 0.8)
 opriskmodel[[2]]$freqdist = buildFreqdist("pois", 50)

 sla(opriskmodel, alpha = 0.95)

 #generate plot if interpolation was performed
 sla(opriskmodel, alpha = 0.95, plot = TRUE)
```

Index

∗ **datasets**
  lossdat, 23

buildFreqdist, 3, *9*
buildMixingSevdist, *4*, 4, *5*, *6*
buildPlainSevdist, *4*, 5, *6*
buildSplicedSevdist, *4*, *5*, 6, *14*

chisq.test, *21*

dgh *(*gh*)*, 19
dgpd *(*gpd*)*, 22
dmixing *(*Mixing*)*, 25
dsevdist, 7
dspliced *(*fitSpliced*)*, 12

evaluate *(*buildMixingSevdist*)*, 4

fitDependency, 8
fitFreqdist, *4*, 9
fitMixing, 10
fitPlain, 11
fitSpliced, 12
fitSplicedBayes, 13
fitSplicedBayesGammaGPD *(*fitSplicedBayes*)*, 13
fitSplicedBayesKDEGPD *(*fitSplicedBayes*)*, 13
fitSplicedBayesLognormGPD *(*fitSplicedBayes*)*, 13
fitSplicedBayesWeibullGPD *(*fitSplicedBayes*)*, 13
fitSplicedBestFit, 15
fitSplicedPar *(*fitSpliced*)*, 12
fitThreshold, 16
fitWeights, 18
fweibullgpd, *14*

gh, 19
goftest, 20
gpd, 22

ks.test, *21*

lossdat, 23

mcSim, 24
Mixing, 25

OpVaR *(*OpVaR-package*)*, 2
OpVaR-package, 2

pgh *(*gh*)*, 19
pgpd *(*gpd*)*, 22
plot.sevdist *(*qqplot.sevdist*)*, 26
pmixing *(*Mixing*)*, 25
print.freqdist *(*buildFreqdist*)*, 3
print.sevdist *(*dsevdist*)*, 7
psevdist *(*dsevdist*)*, 7
pspliced *(*fitSpliced*)*, 12

qgh *(*gh*)*, 19
qgpd *(*gpd*)*, 22
qmixing *(*Mixing*)*, 25
qqplot.sevdist, 26
qsevdist *(*dsevdist*)*, 7
qspliced *(*fitSpliced*)*, 12

rgh *(*gh*)*, 19
rgpd *(*gpd*)*, 22
rmixing *(*Mixing*)*, 25
rsevdist *(*dsevdist*)*, 7
rspliced *(*fitSpliced*)*, 12

sla, *24*, 27
stats, *21*

VaR *(*mcSim*)*, 24