Package 'RobExtremes'

September 4, 2024

Version 1.3.1

Date 2024-09-04

Title Optimally Robust Estimation for Extreme Value Distributions

Description Optimally robust estimation for extreme value distributions using S4 classes and methods (based on packages 'distr', 'distrEx', 'distrMod', 'RobAStBase', and 'ROptEst'); the underlying theoretic results can be found in Ruckdeschel and Horbenko, (2013 and 2012), \doi{10.1080/02331888.2011.628022} and \doi{10.1007/s00184-011-0366-4}.

Depends R(>= 3.4), methods, distrMod(>= 2.8.0), ROptEst(>= 1.2.0), robustbase, evd

Suggests RUnit(>= 0.4.26), ismev(>= 1.39)

Enhances fitdistrplus(>= 1.0-9)

Imports RobAStRDA, distr, distrEx(>= 2.8.0), RandVar, RobAStBase(>= 1.2.0), startupmsg, actuar

ByteCompile yes

LazyLoad yes

License LGPL-3

Encoding UTF-8

URL https://r-forge.r-project.org/projects/robast/

LastChangedDate {$LastChangedDate: 2024-08-29 21:28:04 +0200 (Do, 29 Aug 2024) $}

LastChangedRevision {$LastChangedRevision: 1312 $}

VCS/SVNRevision 1318

NeedsCompilation yes

Author Nataliya Horbenko [aut, cph],
Bernhard Spangl [ctb] (contributed smoothed grid values of the Lagrange multipliers),
Sascha Desmettre [ctb] (contributed smoothed grid values of the Lagrange multipliers),
Eugen Massini [ctb] (contributed an interactive smoothing routine for

smoothing the Lagrange multipliers and smoothed grid values of the
Lagrange multipliers),
Daria Pupashenko [ctb] (contributed MDE-estimation for GEV distribution
in the framework of her PhD thesis 2011--14),
Gerald Kroisandt [ctb] (contributed testing routines),
Matthias Kohl [aut, cph] (<https://orcid.org/0000-0001-9514-8910>),
Peter Ruckdeschel [cre, aut, cph]
(<https://orcid.org/0000-0001-7815-4809>)

Maintainer Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

Repository CRAN

Date/Publication 2024-09-04 15:20:02 UTC

Contents

RobExtremes-package 3
.checkEstClassForParamFamily-methods 8
asvarMedkMAD 9
asvarPickands 10
asvarQBCC 12
checkmakeIC-methods 13
E 14
getCVaR 16
getStartIC-methods 17
GEV 19
GEV-class 20
GEVFamily 22
GEVFamilyMuUnknown 24
GEVParameter-class 26
GPareto 27
GPareto-class 28
GParetoFamily 30
GParetoParameter-class 32
Gumbel 33
Gumbel-class 34
GumbelLocationFamily 36
GumbelParameter-class 37
InternalEstimatorReturnClasses 38
interpolateSn 39
ismevgpdgevdiag-methods 41
kMAD 43
LDEstimate-class 45
LDEstimator 46
movToRef-methods 49
Pareto 50
Pareto-class 51
ParetoFamily 53
ParetoParameter-class 54

PickandsEstimator 56
QuantileBCCEstimator 58
RobExtremesConstants 59
validParameter-methods 60
var 61
WeibullFamily 65

Index **67**

`RobExtremes-package` *RobExtremes – Optimally Robust Estimation for Extreme Value Distributions*

Description

RobExtremes provides infrastructure for speeded-up optimally robust estimation (i.e., MBRE, OMSE, RMXE) for extreme value distributions, extending packages **distr**, **distrEx**, **distrMod**, **robustbase**, **RobAStBase**, and **ROptEst**.

Details

Package:	RobExtremes
Version:	1.3.1
Date:	2024-09-04
Title:	Optimally Robust Estimation for Extreme Value Distributions
Description:	Optimally robust estimation for extreme value distributions using S4 classes and methods (based on packages distr, distrEx, distrMod, RobAStBase, and ROptEst).
Depends:	R(>= 3.4), methods, distrMod(>= 2.8.0), ROptEst(>= 1.2.0), robustbase, evd
Suggests:	RUnit(>= 0.4.26), ismev(>= 1.39)
Imports:	RobAStRDA, distr, distrEx(>= 2.8.0), RandVar, RobAStBase(>= 1.2.0), startupmsg,actuar
Authors:	Bernhard Spangl [contributed smoothed grid values of the Lagrange multipliers] Sascha Desmettre [contributed smoothed grid values of the Lagrange multipliers] Eugen Massini [contributed an interactive smoothing routine for smoothing the Lagrange multipliers and smoothed grid values of the Lagrange multipliers] Daria Pupashenko [contributed MDE-estimation for GEV distribution in the framework of her PhD thesis 2011–14] Gerald Kroisandt [contributed testing routines] Nataliya Horbenko ["aut","cph"] Matthias Kohl ["aut", "cph"] Peter Ruckdeschel ["cre", "aut", "cph"],
Contact:	peter.ruckdeschel@uni-oldenburg.de
ByteCompile:	yes
LazyLoad:	yes
License:	LGPL-3
URL:	https://r-forge.r-project.org/projects/robast/
Encoding:	UTF-8
VCS/SVNRevision:	1318

Distributions

Importing from packages **actuar**, **evd**, it provides S4 classes and methods for the

- Gumbel distribution
- Generalized Extreme Value distribution (GEVD)
- Generalized Pareto distribution (GPD)
- Pareto distribution

Functionals for Distributions

These distributions come together with particular methods for expectations. I.e., a functional E() as in package **distrEx**, which as first argument takes the distribution, and, optionally, can take as second argument a function which then is used as integrand. These particular methods are available for the GPD, Pareto, Gamma, Weibull, and GEV disdribution and use integration on the quantile scale, i.e.,

$$E[X] = \int_0^1 q^X(s)\, ds$$

where q^X is the quantile function of X. In addition, where they exist, we provide closed from expressions for variances, median, IQR, skewness, kurtosis.
In addition, extending estimators `Sn` and `Qn` from package **robustbase**, we provide functionals for Sn and Qn. A new asymmetric version of the `mad`, `kMAD` gives yet another robust scale estimator (and functional).

Models and Estimators

As to models, we provide the

- GPD model (with known threshold), together with (speeded-up) optimally robust estimators, with LDEstimators (in general, and with `medkMAD`, `medSn` and `medQn` as particular ones) and Pickands' estimator as starting estimators.
- GEVD model (with known or unknown threshold), together with (speeded-up) optimally robust estimators, with LDEstimators (see above) and Pickands' estimator as starting estimators.
- Pareto model
- Weibull model
- Gamma model

and for each of these, we provide speeded-up optimally robust estimation (i.e., MBRE, OMSE, RMXE).
We robust (high-breakdown) starting estimators for

- GPD (PickandsEstimator, medkMAD, medSn, medQn)
- GEV (PickandsEstimator)

- Pareto (Cramér-von-Mises-Minimum-Distance-Estimator)
- Weibull (the quantile based estimator of Boudt/Caliskan/Croux)
- Gamma (Cramér-von-Mises-Minimum-Distance-Estimator)

For all these families, of course, MLEs and Minimum-Distance-Estimators are also available through package "distrMod".

Diagnostics

We bridge to the diagnostics provided by package "ismev", i.e. our return objects can be plugged into the diagnostics of this package.
We have the usual diagnostic plots from package "RobAStBase", i.e.

- Outylingness plots `outlyingPlotIC`
- IC plots `plot`
- Information plots via `infoPlot`
- IC comparison plots via `comparePlot`
- Cniperpoint plots (from package "ROptEst") via `CniperPointPlot`

but also (adopted from package "distrMod")

- qqplots (with confidence bands) via `qqplot`
- returnlevel plots via `returnlevelplot`

Starting Point

As a starting point you may look at the included script '`"RobFitsAtRealData.R"`' in the scripts folder of the package, accessible by `file.path(system.file(package="RobExtremes"), "scripts/RobFitsAtRealDat`

Classes

```
[*]: there is a generating function with the same name in RobExtremes
[**]:  generating function from distrMod, but with (speeded-up)
       opt.rob-estimators in RobExtremes
##########################
Distribution Classes
##########################
"Distribution" (from distr)
|>"UnivariateDistribution" (from distr)
|>|>"AbscontDistribution" (from distr)
|>|>|>"Gumbel"    [*]
|>|>|>"Pareto"    [*]
|>|>|>"GPareto"   [*]
|>|>|>"GEVD"      [*]
##########################
Parameter Classes
##########################
"OptionalParameter" (from distr)
```

```
|>"Parameter" (from distr)
|>|>"GumbelParameter"
|>|>"ParetoParameter"
|>|>"GEVDParameter"
|>|>"GParetoParameter"
#########################
ProbFamily classes
#########################
slots: [<name>(<class>)]
"ProbFamily"                                (from distrMod)
|>"ParamFamily"                             (from distrMod)
|>|>"L2ParamFamily"                         (from distrMod)
|>|>|>"L2GroupParamFamily"                  (from distrMod)
|>|>|>|>"ParetoFamily"                [*]
|>|>|>|>"L2ScaleShapeUnion"                (from distrMod)
|>|>|>|>|>"GammaFamily"               [**]
|>|>|>|>|>"GParetoFamily"              [*]
|>|>|>|>|>"GEVFamily"                  [*]
|>|>|>|>|>"WeibullFamily"              [**]
|>|>|>|>"L2LocationScaleUnion"  /VIRTUAL/  (from distrMod)
|>|>|>|>|>"L2LocationFamily"             (from distrMod)
|>|>|>|>|>|>"GumbelLocationFamily"     [*]
|>|>|>|>"L2LocScaleShapeUnion"  /VIRTUAL/  (from distrMod)
|>|>|>|>|>"GEVFamilyMuUnknown"          [*]
```

Functions

```
LDEstimator     Estimators for scale-shape models based on
                location and dispersion
medSn                    loc=median disp=Sn
medQn                    loc=median disp=Qn
medkMAD                  loc=median disp=kMAD
asvarMedkMAD               [asy. variance to MedkMADE]
PickandsEstimator        PickandsEstimator
asvarPickands              [asy. variance to PickandsE]
QuantileBCCEstimator     Quantile based estimator for the Weibull distribution
asvarQBCC                  [asy. variance to QuantileBCCE]
```

Generating Functions

```
Distribution Classes
Gumbel                  Generating function for Gumbel-class
GEVD                    Generating function for GEVD-class
GPareto                 Generating function for GPareto-class
Pareto                  Generating function for Pareto-class
L2Param Families
ParetoFamily            Generating function for ParetoFamily-class
GParetoFamily           Generating function for GParetoFamily-class
GEVFamily               Generating function for GEVFamily-class
```

WeibullFamily	Generating function for WeibullFamily-class

Methods

Functionals:	
E	Generic function for the computation of (conditional) expectations
var	Generic functions for the computation of functionals
IQR	Generic functions for the computation of functionals
median	Generic functions for the computation of functionals
skewness	Generic functions for the computation of functionals
kurtosis	Generic functions for the computation of functionals
Sn	Generic function for the computation of (conditional) expectations
Qn	Generic functions for the computation of functionals

Constants

EULERMASCHERONICONSTANT
APERYCONSTANT

Acknowledgement

This package is joint work by Peter Ruckdeschel, Matthias Kohl, and Nataliya Horbenko (whose PhD thesis went into this package to a large extent), with contributions by Dasha Pupashenko, Misha Pupashenko, Gerald Kroisandt, Eugen Massini, Sascha Desmettre, and Bernhard Spangl, in the framework of project "Robust Risk Estimation" (2011-2016) funded by Volkswagen foundation (and gratefully ackknowledged). Thanks also goes to the maintainers of CRAN, in particully to Uwe Ligges who greatly helped us with finding an appropriate way to store the database of interpolating functions which allow the speed up – this is now package RobAStRDA on CRAN.

Start-up-Banner

You may suppress the start-up banner/message completely by setting `options("StartupBanner"="off")` somewhere before loading this package by `library` or `require` in your R-code / R-session. If option `"StartupBanner"` is not defined (default) or setting `options("StartupBanner"=NULL)` or `options("StartupBanner"="complete")` the complete start-up banner is displayed. For any other value of option `"StartupBanner"` (i.e., not in `c(NULL,"off","complete")`) only the version information is displayed. The same can be achieved by wrapping the `library` or `require` call into either `suppressStartupMessages()` or `onlytypeStartupMessages(.,atypes="version")`. As for general `packageStartupMessage`'s, you may also suppress all the start-up banner by wrapping the `library` or `require` call into `suppressPackageStartupMessages()` from **startupmsg**-version 0.5 on.

Package versions

Note: The first two numbers of package versions do not necessarily reflect package-individual development, but rather are chosen for the RobAStXXX family as a whole in order to ease updating "depends" information.

Author(s)

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>,
Matthias Kohl <Matthias.Kohl@stamats.de>, and
Nataliya Horbenko <nhorbenko@gmail.com>,
Maintainer: Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

References

Horbenko, N., Ruckdeschel, P., and Bae, T. (2011): Robust Estimation of Operational Risk. Journal of Operational Risk *6*(2), 3-30.
M. Kohl (2005). Numerical Contributions to the Asymptotic Theory of Robustness. Dissertation. University of Bayreuth. https://epub.uni-bayreuth.de/id/eprint/839/2/DissMKohl.pdf.
M. Kohl, P. Ruckdeschel, and H. Rieder (2010). Infinitesimally Robust Estimation in General Smoothly Parametrized Models. Statistical Methods and Applications *19*(3): 333-354. doi:10.1007/s1026001001330.
Ruckdeschel, P. and Horbenko, N. (2013): Optimally-Robust Estimators in Generalized Pareto Models. *Statistics*. **47**(4), 762–791. doi:10.1080/02331888.2011.628022.
Ruckdeschel, P. and Horbenko, N. (2012): Yet another breakdown point notion: EFSBP –illustrated at scale-shape models. *Metrika*, **75**(8), 1025–1047. doi:10.1007/s0018401103664.
Ruckdeschel, P., Kohl, M., Stabla, T., and Camphausen, F. (2006): S4 Classes for Distributions, *R News*, *6*(2), 2-6. https://CRAN.R-project.org/doc/Rnews/Rnews_2006-2.pdf.
A vignette for packages **distr**, **distrSim**, **distrTEst**, and **RobExtremes** is included into the mere documentation package **distrDoc** and may be called by `require("distrDoc");vignette("distr")`.
A homepage to this package is available under http://robast.r-forge.r-project.org/.

See Also

distr-package, distrEx-package, distrMod-package, RobAStBase-package, ROptEst-package

---

.checkEstClassForParamFamily-methods

Methods for Function .checkEstClassForParamFamily in Package 'RobExtremes'

Description

.checkEstClassForParamFamily-methods

Arguments

PFam	a parametric family.
estimator	an estimator.

Details

The respective methods can be used to cast an estimator to a model-specific subclass with particular methods.

Value

The GParetoFamily,Estimate-method returns the estimator cast to S4 class GPDEstimate,
the GParetoFamily,LDEstimate-method cast to S4 class GPDLDEstimate,
the GParetoFamily,MCEstimate-method cast to S4 class GPDMCEstimate,
the GParetoFamily,kStepEstimate-method cast to S4 class GPDkStepEstimate,
the GParetoFamily,ORobEstimate-method cast to S4 class GPDORobEstimate,
the GParetoFamily,MDEstimate-method cast to S4 class GPDMDEstimate,
the GParetoFamily,MLEstimate-method cast to S4 class GPDML.ALEstimate,
the GParetoFamily,CvMMDEstimate-method cast to S4 class GPDCvMMD.ALEstimate,

The GEVFamily,Estimate-method returns the estimator cast to S4 class GEVEstimate,
the GEVFamily,LDEstimate-method cast to S4 class GEVLDEstimate,
the GEVFamily,MCEstimate-method cast to S4 class GEVMCEstimate,
the GEVFamily,kStepEstimate-method cast to S4 class GEVkStepEstimate,
the GEVFamily,ORobEstimate-method cast to S4 class GEVORobEstimate,
the GEVFamily,MDEstimate-method cast to S4 class GEVMDEstimate,
the GEVFamily,MLEstimate-method cast to S4 class GEVML.ALEstimate,
the GEVFamily,CvMMDEstimate-method cast to S4 class GEVCvMMD.ALEstimate,

the GEVFamilyMuUnknown,Estimate-method cast to S4 class GEVEstimate,
the GEVFamilyMuUnknown,LDEstimate-method cast to S4 class GEVLDEstimate,
the GEVFamilyMuUnknown,MCEstimate-method cast to S4 class GEVMCEstimate,
the GEVFamilyMuUnknown,kStepEstimate-method cast to S4 class GEVkStepstimate.
the GEVFamilyMuUnknown,ORobEstimate-method cast to S4 class GEVORobEstimate,
the GEVFamilyMuUnknown,MDEstimate-method cast to S4 class GEVMDEstimate,
the GEVFamilyMuUnknown,MLEstimate-method cast to S4 class GEVML.ALEstimate,
the GEVFamilyMuUnknown,CvMMDEstimate-method cast to S4 class GEVCvMMD.ALEstimate.

Author(s)

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

---

asvarMedkMAD *Function to compute asymptotic variance of MedkMAD estimator*

Description

Function asvarMedkMAD computes the asymptotic (co)variance of a MedkMAD estimator at a Scale-Shape model.

Usage

```
asvarMedkMAD( model, k=1)
```

Arguments

model	an object of class "ScaleShapeUnion".
k	numeric (>0); additional parameter for kMAD.

Details

For the Generalized Pareto Family all terms are analytic; in case of the general scale-shape model, numerical integration is used.

Value

A 2x2 matrix; the covariance.

Author(s)

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

References

Ruckdeschel, P. and Horbenko, N. (2011): Optimally-Robust Estimators in Generalized Pareto Models. ArXiv 1005.1476. To appear at *Statistics*. DOI: 10.1080/02331888.2011.628022.

See Also

LDEstimator

Examples

```
GP <- GParetoFamily(scale=1,shape=0.7)
asvarMedkMAD(GP,k=1)

## for didactical purposes turn GP into a non-GPD
setClass("noGP",contains="L2ScaleShapeUnion")
GP2 <- GP
class(GP2) <- "noGP"
asvarMedkMAD(GP2,k=1) ### uses numerical integration
```

---

asvarPickands *Function to compute asymptotic variance of Pickands estimator*

Description

Function asvarPickands computes the asymptotic (co)variance of a Pickands estimator at a GPD or GEVD model – the latter with location mu known or unknown.

Usage

```
asvarPickands( model, alpha=2)
```

Arguments

model	an object of class "ScaleShapeUnion".
alpha	numeric > 1; determines the variant of the Pickands-Estimator based on matching the empirical $a_1 = 1-1/\alpha$ and $a_1 = 1-1/\alpha^2$ quantiles against the population counter parts. The "classical" Pickands Estimator is obtained for alpha=2 (GPD) resp. for alpha=1/log(2) (GEVD).

Details

All terms are analytic.

Value

A 2x2 matrix (resp., for mu unknown in the GEV model a 3x3 matrix); the covariance.

Author(s)

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

References

Ruckdeschel, P. and Horbenko, N. (2013): Optimally-Robust Estimators in Generalized Pareto Models. *Statistics* 47(4), 762–791. DOI: 10.1080/02331888.2011.628022.

See Also

PickandsEstimator

Examples

```
GP <- GParetoFamily(scale=1,shape=0.7)
asvarPickands(GP)
asvarPickands(GP,alpha=2.3)
GE <- GEVFamily(loc=0,scale=1,shape=0.7)
asvarPickands(GE)
GE0 <- GEVFamilyMuUnknown(loc=0,scale=1,shape=0.7)
asvarPickands(GE0)
```

asvarQBCC *Function to compute asymptotic variance of QuantileBCC estimator*

Description

Function asvarQBCC computes the asymptotic (co)variance of a QuantileBCC estimator at a Weibull model.

Usage

```
asvarQBCC( model, p1 = 1/3, p2 = 2/3)
```

Arguments

model	an object of class "ScaleShapeUnion".
p1, p2	levels of the quantiles; maximal breakdown point is achieved for $p1 = p2 - p1 = 1 - p2 = 1/3$ which is the default.

Details

All terms are analytic.

Value

A 2x2 matrix; the covariance.

Author(s)

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

See Also

QuantileBCCEstimator

Examples

```
GP <- WeibullFamily(scale=1,shape=0.7)
asvarQBCC(GP)
asvarQBCC(GP, p1=1/4, p2= 5/8)
```

checkmakeIC-methods *Methods for Functions checkIC and makeIC in Package 'RobExtremes'*

Description

checkIC checks accuracy of the centering and Fisher consistency condition of an IC, makeIC, by centering and restandardizing warrants these conditions.

Methods

checkIC `signature(IC="IC", L2Fam = "ParetoFamily")`: To enhance accuracy, the method for `"ParetoFamily"` uses integration via the quantile transform, i.e., $E[h(X)]$ for a random variable $X \sim F$ with quantil function q is computed as $\int_0^1 h(q(s))\, ds$

checkIC `signature(IC="IC", L2Fam = "GParetoFamily")`: As for `"ParetoFamily"`, to enhance accuracy, the method for `"GParetoFamily"` uses integration via the quantile transform.

checkIC `signature(IC="IC", L2Fam = "GEVFamily")`: As for `"ParetoFamily"`, to enhance accuracy, the method for `"GEVFamily"` uses integration via the quantile transform.

checkIC `signature(IC="IC", L2Fam = "GEVFamilyMuUnknown")`: As for `"ParetoFamily"`, to enhance accuracy, the method for `"GEVFamilyMuUnknown"` uses integration via the quantile transform.

makeIC `signature(IC="IC", L2Fam = "ParetoFamily")`: As with `"checkIC"`, to enhance accuracy, the method for `"makeIC"` for `"ParetoFamily"` uses integration via the quantile transform.

makeIC `signature(IC="IC", L2Fam = "GParetoFamily")`: As for `"ParetoFamily"`, to enhance accuracy, the method for `"GParetoFamily"` uses integration via the quantile transform.

makeIC `signature(IC="IC", L2Fam = "GEVFamily")`: As for `"ParetoFamily"`, to enhance accuracy, the method for `"GEVFamily"` uses integration via the quantile transform.

makeIC `signature(IC="IC", L2Fam = "GEVFamilyMuUnknown")`: As for `"ParetoFamily"`, to enhance accuracy, the method for `"GEVFamilyMuUnknown"` uses integration via the quantile transform.

Author(s)

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

See Also

checkIC,makeIC

E *Generic Function for the Computation of (Conditional) Expectations*

Description

Generic function for the computation of (conditional) expectations.

Usage

```
E(object, fun, cond, ...)

## S4 method for signature 'GEV,missing,missing'
E(object, low = NULL, upp = NULL, ..., diagnostic = FALSE)
## S4 method for signature
## 'DistributionsIntegratingByQuantiles,function,missing'
E(object,
         fun, low = NULL, upp = NULL,
         rel.tol= getdistrExOption("ErelativeTolerance"),
         lowerTruncQuantile = getdistrExOption("ElowerTruncQuantile"),
         upperTruncQuantile = getdistrExOption("EupperTruncQuantile"),
        IQR.fac = max(1e4,getdistrExOption("IQR.fac")), ..., diagnostic = FALSE)
## S4 method for signature 'Gumbel,missing,missing'
E(object, low = NULL, upp = NULL, ..., diagnostic = FALSE)
## S4 method for signature 'GPareto,missing,missing'
E(object, low = NULL, upp = NULL, ..., diagnostic = FALSE)
## S4 method for signature 'GPareto,function,missing'
E(object, fun, low = NULL, upp = NULL,
             rel.tol= getdistrExOption("ErelativeTolerance"),
             lowerTruncQuantile = getdistrExOption("ElowerTruncQuantile"),
             upperTruncQuantile = getdistrExOption("EupperTruncQuantile"),
         IQR.fac = max(1e4,getdistrExOption("IQR.fac")), ..., diagnostic = FALSE)
## S4 method for signature 'Pareto,missing,missing'
E(object, low = NULL, upp = NULL, ..., diagnostic = FALSE)
```

Arguments

object	object of class "Distribution"
fun	if missing the (conditional) expectation is computed else the (conditional) expection of fun is computed.
cond	if not missing the conditional expectation given cond is computed.
rel.tol	relative tolerance for distrExIntegrate.
low	lower bound of integration range.
upp	upper bound of integration range.
lowerTruncQuantile	lower quantile for quantile based integration range.

upperTruncQuantile
: upper quantile for quantile based integration range.

IQR.fac
: factor for scale based integration range (i.e.; median of the distribution $\pm$IQR.fac$\times$IQR).

...
: additional arguments to fun

diagnostic
: logical; if TRUE, the return value obtains an attribute "diagnostic" with diagnostic information on the integration, i.e., a list with entries method ("integrate" or "GLIntegrate"), call, result (the complete return value of the method), args (the args with which the method was called), and time (the time to compute the integral).

Details

The precision of the computations can be controlled via certain global options; cf. distrExOptions. Also note that arguments low and upp should be given as named arguments in order to prevent them to be matched by arguments fun or cond. Also the result, when arguments low or upp is given, is the *unconditional value* of the expectation; no conditioning with respect to low <= object <= upp is done. To be able to use integration after transformation via the respective probability transformation to [0,1], we introduce a class union "DistributionsIntegratingByQuantiles", which currently comprises classes "GPareto", "Pareto", "Weibull", "GEV". In addition, the specific method for "GPareto", "function", "missing" uses integration on [0,1] via the substitution method (y := log(x)).

Diagnostics on the involved integrations are available if argument diagnostic is TRUE. Then there is attribute diagnostic attached to the return value, which may be inspected and accessed through showDiagnostic and getDiagnostic.

Value

The expectation is computed.

Methods

object = "Gumbel", fun = "missing", cond = "missing": exact evaluation using explicit expressions.

object = "GPareto", fun = "missing", cond = "missing": exact evaluation using explicit expressions.

object = "DistributionsIntegratingByQuantiles", fun = "function", cond = "missing": use probability transform, i.e., a substitution y = p(object)(x) for numerical integration.

object = "GPareto", fun = "function", cond = "missing": use substitution method (y := log(x)) for numerical integration.

object = "Pareto", fun = "missing", cond = "missing": exact evaluation using explicit expressions.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de> and Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

See Also

distrExIntegrate, m1df, m2df, Distribution-class

Examples

```
GP <- GPareto(shape=0.3)

E(GP)
E(GP, fun = function(x){2*x^2}) ## uses the log trafo

P <- Pareto()
E(P)
E(P,fun = function(x){1/(x^2+1)})
```

---

getCVaR *Risk Measures for Scale-Shape Families*

Description

Functions to compute Value-at-Risk (VaR), Conditional Value-at-Risk (CVaR) and Expected Loss (EL) at data from scale-shape families.

Usage

```
getVaR(data, model, level, rob=TRUE)
getCVaR(data, model, level, rob=TRUE)
getEL(data, model, N0, rob=TRUE)
## S3 method for class 'riskMeasure'
print(x, level=NULL, ...)
```

Arguments

data	data at which to compute the risk measure.
model	an object of class "L2ScaleShapeFamily". The parametric family at which to evaluate the risk measure.
level	real: probability needed for VaR and CVaR.
N0	real: expected frequency for expected loss.
rob	logical; if TRUE (default) the RMXE-parametric estimator is used; otherwise the MLE.
x	an object of (S3-)class "riskmeasure".
...	further arguments for print.

Value

The risk measures getVaR, getCVaR, getEL return an (S3) object of class "riskMeasure", i.e., a numeric vector of length 2 with components "Risk" and "varofRisk" containing the respective risk measure and a corresponding (asymptotic) standard error for the risk measure. To the return class "riskMeasure", there is a particular print-method; if the corresponding argument level is NULL (default) the corresponding standard error is printed together with the risk measure; otherwise a corresponding CLT-based confidence interval for the risk meausre is produced.

Author(s)

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

References

P. Ruckdeschel, N. Horbenko (2013): Optimally-Robust Estimators in Generalized Pareto Models. Statistics 47(4), 762–791. doi:10.1080/02331888.2011.628022.
N. Horbenko, P. Ruckdeschel, T. Bae (2011): Robust Estimation of Operational Risk. Journal of Operational Risk 6(2), 3–30.

See Also

GParetoFamily, GEVFamily, WeibullFamily, GammaFamily

Examples

```
 # to reduce checking time
set.seed(123)
GPD <- GParetoFamily(loc=20480, scale=7e4, shape=0.3)
data <- r(GPD)(500)
getCVaR(data,GPD,0.99)
getVaR(data,GPD,0.99)
getEL(data,GPD,5)
getVaR(data,GPD,0.99, rob=FALSE)
getEL(data,GPD,5, rob=FALSE)
getCVaR(data,GPD,0.99, rob=FALSE)
```

---

getStartIC-methods *Methods for Function getStartIC in Package 'RobExtremes'*

Description

getStartIC computes the optimally-robust IC to be used as argument ICstart in kStepEstimator.

Usage

```
getStartIC(model, risk, ...)
## S4 method for signature 'L2ScaleShapeUnion,interpolRisk'
getStartIC(model, risk, ...,
   withMakeIC = FALSE, ..debug=FALSE, modifyICwarn = NULL)
## S4 method for signature 'L2LocScaleShapeUnion,interpolRisk'
getStartIC(model, risk, ...,
   withMakeIC = FALSE, ..debug=FALSE, modifyICwarn = NULL)
## S4 method for signature 'ParetoFamily,interpolRisk'
getStartIC(model, risk, ...,
   withMakeIC = FALSE)
```

Arguments

model	normtype of class NormType
risk	normtype of class NormType
...	further arguments to be passed to specific methods.
withMakeIC	logical; if TRUE the IC is passed through makeIC before return.
..debug	logical; if TRUE information for debugging is issued.
modifyICwarn	logical: should a (warning) information be added if modifyIC is applied and hence some optimality information could no longer be valid? Defaults to NULL in which case this value is taken from RobAStBaseOptions.

Details

getStartIC is used internally in functions robest and roptest to compute the optimally robust influence function according to the arguments given to them.

Value

An IC of type HampIC.

Methods

getStartIC signature(model = "L2ScaleShapeUnion", risk = "interpolRisk"): computes the optimally robust influence function by interpolation on a grid (using internal helper function .getPsi).

getStartIC signature(model = "L2LocScaleShapeUnion", risk = "interpolRisk"): computes the optimally robust influence function by interpolation on a grid (using internal helper function .getPsi.wL).

getStartIC signature(model = "ParetoFamily", risk = "interpolRisk"): computes the optimally robust influence function by interpolation on a grid (using internal helper function .getPsi.P).

All of these methods recenter and restandardize the obtained ICs to warrant centeredness and Fisher consistency.

Author(s)

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

See Also

robest,optIC, radiusMinimaxIC

GEV *Generating function for GEV-class*

Description

Generates an object of class "GEV".

Usage

```
GEV(loc = 0, scale = 1, shape = 0, location = loc)
```

Arguments

loc	real number: location parameter of the GEV distribution.
scale	positive real number: scale parameter of the GEV distribution
shape	non-negative real number: shape parameter of the GEV distribution.
location	real number: location of GEV distribution

Value

Object of class "GEV"

Note

The class "GEV" is based on the code provided by the package **evd** by Alec Stephenson.

Author(s)

Nataliya Horbenko <nhorbenko@gmail.com>

See Also

GEV-class, dgpd

Examples

```
(P1 <- GEV(loc = 0, scale = 1, shape = 0))
plot(P1)

E(GEV())
E(P1)
E(P1, function(x){x^2})
var(P1)
sd(P1)
median(P1)
IQR(P1)
mad(P1)
```

GEV-class *Generalized EV distribution*

Description

[borrowed from **evd**]: The GEV distribution function with parameters loc $= a$, scale $= b$, shape $= s$ is

$$G(x) = exp[-1 + s(z-a)/b^{(} - 1/s)]$$

for $1 + s(z-a)/b > 0$, where $b > 0$. If $s = 0$ the distribution is defined by continuity and gives the Gumbel distribution. If $1 + s(z-a)/b \leq 0$, the value z is either greater than the upper end point (if $s < 0$), or less than the lower end point (if $s > 0$).

Objects from the Class

Objects can be created by calls of the form new("GEV", loc, scale,shape). More frequently they are created via the generating function GEV.

Slots

img Object of class "Reals".

param Object of class "GEVParameter".

r rgpd

d dgpd

p pgpd, but vectorized and with special treatment of arguments lower.tail and log.p

q qgpd, but vectorized and with special treatment of arguments lower.tail and log.p

gaps (numeric) matrix or NULL

.withArith logical: used internally to issue warnings as to interpretation of arithmetics

.withSim logical: used internally to issue warnings as to accuracy

.logExact logical: used internally to flag the case where there are explicit formulae for the log version of density, cdf, and quantile function

.lowerExact logical: used internally to flag the case where there are explicit formulae for the lower tail version of cdf and quantile function

Extends

Class "AbscontDistribution", directly.
Class "UnivariateDistribution", by class "AbscontDistribution".
Class "Distribution", by class "AbscontDistribution".

Methods

initialize signature(.Object = "GEV"): initialize method.

shape signature(object = "GEV"): wrapped access method for slot shape of slot param.

loc signature(object = "GEV"): wrapped access method for slot loc of slot param.

location signature(object = "GEV"): alias to loc, to support argument naming of package **VGAM**.

scale signature(x = "GEV"): wrapped access method for slot scale of slot param.

shape<- signature(object = "GEV"): wrapped replace method for slot shape of slot param.

loc<- signature(object = "GEV"): wrapped replace method for slot loc of slot param.

location<- signature(object = "GEV"): alias to loc<-, to support argument naming of package **VGAM**.

scale<- signature(x = "GEV"): wrapped replace method for slot scale of slot param.

+ signature(e1 = "GEV", e2 = "numeric"): exact method for this transformation — stays within this class.

***** signature(e1 = "GEV", e2 = "numeric"): exact method for this transformation — stays within this class if e2>0.

E signature(object = "GEV", fun = "missing", cond = "missing"): exact evaluation using explicit expressions.

var signature(signature(x = "GEV"): exact evaluation using explicit expressions.

median signature(signature(x = "GEV"): exact evaluation using explicit expressions.

IQR signature(signature(x = "GEV"): exact evaluation using explicit expressions.

skewness signature(signature(x = "GEV"): exact evaluation using explicit expressions.

kurtosis signature(signature(x = "GEV"): exact evaluation using explicit expressions.

liesInSupport signature(object = "GEV", x = "numeric"): checks if x lies in the support of the respective distribution.

Note

This class is based on the code provided by the package **evd** by A. G. Stephenson.

Author(s)

Nataliya Horbenko <nhorbenko@gmail.com>

References

Pickands, J. (1975) *Statistical inference using extreme order statistics. _Annals of Statistics_, *3*, 119-131.*

See Also

dgpd, AbscontDistribution-class

Examples

```
(P1 <- new("GEV", loc = 0, scale = 1,shape = 0))
plot(P1)
shape(P1)
loc(P1)
scale(P1) <- 4
loc(P1) <- 2
shape(P1) <- -1 # may be negative!
plot(P1)
```

GEVFamily *Generating function for families of Generalized Extreme Value distributions*

Description

Generates an object of class "GEVFamily" which represents a Generalized EV family.

Usage

```
GEVFamily(loc = 0, scale = 1, shape = 0.5, of.interest = c("scale", "shape"),
          p = NULL, N = NULL, trafo = NULL, start0Est = NULL, withPos = TRUE,
          secLevel = 0.7, withCentL2 = FALSE, withL2derivDistr  = FALSE,
          withMDE = FALSE, ..ignoreTrafo = FALSE, ..withWarningGEV = TRUE)
```

Arguments

loc	real: known/fixed threshold/location parameter
scale	positive real: scale parameter
shape	positive real: shape parameter
of.interest	character: which parameters, transformations are of interest. possibilites are: "scale", "shape", "quantile", "expected loss", "expected shortfall"; a maximum number of two of these may be selected
p	real or NULL: probability needed for quantile and expected shortfall
N	real or NULL: expected frequency for expected loss
trafo	matrix or NULL: transformation of the parameter
start0Est	startEstimator — if NULL PickandsEstimator is used
withPos	logical of length 1: Is shape restricted to positive values?
secLevel	a numeric of length 1: In the ideal GEV model, for each observastion X_i, the expression $1 + \frac{\text{shape}(X_i - \text{loc})}{\text{scale}}$ must be positive, which in principle could be attacked by a single outlier. Hence for sample size n we allow for εn violations, interpreting the violations as outliers. Here $\varepsilon = \texttt{secLevel}/\sqrt{n}$.

withCentL2 logical: shall L2 derivative be centered by substracting the E()? Defaults to FALSE, but higher accuracy can be achieved when set to TRUE.

withL2derivDistr logical: shall the distribution of the L2 derivative be computed? Defaults to FALSE (to speed up computations).

withMDE logical: should Minimum Distance Estimators be used to find a good starting value for the parameter search? Defaults to FALSE (to speed up computations). We have seen cases though, where the use of the then employed PickandsEstimator was drastically misleading and subsequently led to bad estimates where it is used as starting value; so where feasible it is a good idea to also try argument withMDE=TRUE for control purposes.

..ignoreTrafo logical: only used internally in kStepEstimator; do not change this.

..withWarningGEV logical: shall warnings be issued if shape is large?

Details

The slots of the corresponding L2 differentiable parameteric family are filled.

Value

Object of class "GEVFamily"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>
Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>
Nataliya Horbenko <nhorbenko@gmail.com>

References

Kohl, M. (2005) *Numerical Contributions to the Asymptotic Theory of Robustness*. Bayreuth: Dissertation. https://epub.uni-bayreuth.de/id/eprint/839/2/DissMKohl.pdf.

Kohl, M., Ruckdeschel, P., and Rieder, H. (2010): Infinitesimally Robust Estimation in General Smoothly Parametrized Models. *Stat. Methods Appl.*, **19**, 333-354. doi:10.1007/s1026001001330.

Ruckdeschel, P. and Horbenko, N. (2013): Optimally-Robust Estimators in Generalized Pareto Models. *Statistics*. **47**(4), 762-791. doi:10.1080/02331888.2011.628022.

Ruckdeschel, P. and Horbenko, N. (2012): Yet another breakdown point notion: EFSBP –illustrated at scale-shape models. *Metrika*, **75**(8), 1025–1047. doi:10.1007/s0018401103664.

See Also

L2ParamFamily-class, GPareto

Examples

```
(G1 <- GEVFamily())
FisherInfo(G1)
checkL2deriv(G1)
```

GEVFamilyMuUnknown *Generating function for families of Generalized Extreme Value distributions*

Description

Generates an object of class "GEVFamilyMuUnknown" which represents a Generalized EV family with unknown location parameter mu.

Usage

```
GEVFamilyMuUnknown(loc = 0, scale = 1, shape = 0.5, of.interest = c("loc",
              "scale", "shape"), p = NULL, N = NULL, trafo = NULL,
              start0Est = NULL, withPos = TRUE, secLevel = 0.7,
              withCentL2 = FALSE, withL2derivDistr  = FALSE, withMDE = FALSE,
              ..ignoreTrafo = FALSE, ..withWarningGEV = TRUE, ..name = "")
```

Arguments

loc	real: known/fixed threshold/location parameter
scale	positive real: scale parameter
shape	positive real: shape parameter
of.interest	character: which parameters, transformations are of interest. possibilites are: "scale", "shape", "quantile", "expected loss", "expected shortfall"; a maximum number of two of these may be selected
p	real or NULL: probability needed for quantile and expected shortfall
N	real or NULL: expected frequency for expected loss
trafo	matrix or NULL: transformation of the parameter
start0Est	startEstimator — if NULL PickandsEstimator is used
withPos	logical of length 1: Is shape restricted to positive values?
secLevel	a numeric of length 1: In the ideal GEV model, for each observastion X_i, the expression $1 + \frac{\text{shape}(X_i - \text{loc})}{\text{scale}}$ must be positive, which in principle could be attacked by a single outlier. Hence for sample size n we allow for εn violations, interpreting the violations as outliers. Here $\varepsilon =$ secLevel$/\sqrt{n}$.
withCentL2	logical: shall L2 derivative be centered by substracting the E()? Defaults to FALSE, but higher accuracy can be achieved when set to TRUE.
withL2derivDistr	logical: shall the distribution of the L2 derivative be computed? Defaults to FALSE (to speed up computations).

withMDE — logical: should Minimum Distance Estimators be used to find a good starting value for the parameter search? Defaults to FALSE (to speed up computations). We have seen cases though, where the use of the then employed PickandsEstimator was drastically misleading and subsequently led to bad estimates where it is used as starting value; so where feasible it is a good idea to also try argument withMDE=TRUE for control purposes.

..ignoreTrafo — logical: only used internally in kStepEstimator; do not change this.

..withWarningGEV — logical: shall warnings be issued if shape is large?

..name — character: optional alternative name for the parametric family; used in generating interpolating grids.

Details

The slots of the corresponding L2 differentiable parameteric family are filled.

Value

Object of class "GEVFamilyMuUnknown"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>
Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>
Nataliya Horbenko <nhorbenko@gmail.com>

References

Kohl, M. (2005) *Numerical Contributions to the Asymptotic Theory of Robustness*. Bayreuth: Dissertation. https://epub.uni-bayreuth.de/id/eprint/839/2/DissMKohl.pdf.

Kohl, M., Ruckdeschel, P., and Rieder, H. (2010): Infinitesimally Robust Estimation in General Smoothly Parametrized Models. *Stat. Methods Appl.*, **19**, 333-354. doi:10.1007/s1026001001330.

Ruckdeschel, P. and Horbenko, N. (2013): Optimally-Robust Estimators in Generalized Pareto Models. *Statistics*. **47**(4), 762-791. doi:10.1080/02331888.2011.628022.

Ruckdeschel, P. and Horbenko, N. (2012): Yet another breakdown point notion: EFSBP –illustrated at scale-shape models. *Metrika*, **75**(8), 1025–1047. doi:10.1007/s0018401103664.

See Also

L2ParamFamily-class, GPareto

Examples

```
(G1 <- GEVFamilyMuUnknown())
FisherInfo(G1)
checkL2deriv(G1)
```

GEVParameter-class *Parameter of generalized Pareto distributions*

Description

The class of the parameter of generalized Pareto distribution.

Objects from the Class

Objects can be created by calls of the form `new("GEVParameter", ...)`.

Slots

`loc` real number: location parameter of a GEV distribution.

`scale` real number: scale parameter of a GEV distribution.

`shape` real number: shape parameter of a GEV distribution.

`name` default name is "parameter of a GEV distribution".

Extends

Class `"Parameter"`, directly.
Class `"OptionalParameter"`, by class `"Parameter"`.

Methods

loc `signature(object = "GEVParameter")`: access method for slot `loc`.

location `signature(object = "GEVParameter")`: alias to `loc`, to support argument naming of package **VGAM**.

scale `signature(object = "GEVParameter")`: access method for slot `scale`.

shape `signature(object = "GEVParameter")`: access method for slot `shape`.

loc<- `signature(object = "GEVParameter")`: replace method for slot `loc`.

location<- `signature(object = "GEVParameter")`: alias to `loc<-`, to support argument naming of package **VGAM**.

shape<- `signature(object = "GEVParameter")`: replace method for slot `shape`.

shape<- `signature(object = "GEVParameter")`: replace method for slot `shape`.

Author(s)

Nataliya Horbenko <nhorbenko@gmail.com>

See Also

GEV-class, Parameter-class

Examples

```
P <- new("GEVParameter")
loc(P)
## same as
location(P)
scale(P)
shape(P)

scale(P) <- 2
location(P) <- 4
shape(P) <- -1 # may be negative!
P
```

GPareto *Generating function for GPareto-class*

Description

Generates an object of class "GPareto".

Usage

```
GPareto(loc = 0, scale = 1, shape = 0, location = loc)
```

Arguments

loc	real number: location parameter of the GPareto distribution.
scale	positive real number: scale parameter of the GPareto distribution
shape	non-negative real number: shape parameter of the GPareto distribution.
location	alternative argument name for argument 'loc' — to support argument names of package **VGAM**.

Value

Object of class "GPareto"

Note

The class "GPareto" is based on the code provided by the package **evd** by Alec Stephenson.

Author(s)

Nataliya Horbenko <nhorbenko@gmail.com>

See Also

GPareto-class, dgpd

Examples

```
(P1 <- GPareto(loc = 1, scale = 1, shape = -0.5))
plot(P1)

E(GPareto())
E(P1)
E(P1, function(x){x^2})
var(P1)
sd(P1)
median(P1)
IQR(P1)
mad(P1)
```

GPareto-class *Generalized Pareto distribution*

Description

[borrowed from **evd**]:

The (Three-parameter) generalized Pareto distribution with parameter loc$= a$, scale $= b$, shape $= c$ has density:

$$f(x) = \frac{1}{b}(1 + cz)^{(} - 1/c - 1), \quad z = \frac{x - a}{c}$$

for $x > a$ ($c \geq 0$) and $a \leq x \leq a - b/c (c < 0)$.

Objects from the Class

Objects can be created by calls of the form new("GPareto", loc, scale,shape). More frequently they are created via the generating function GPareto.

Slots

img Object of class "Reals".

param Object of class "GParetoParameter".

r rgpd

d dgpd

p pgpd, but vectorized and with special treatment of arguments lower.tail and log.p

q qgpd, but vectorized and with special treatment of arguments lower.tail and log.p

gaps (numeric) matrix or NULL

.withArith logical: used internally to issue warnings as to interpretation of arithmetics

`.withSim` logical: used internally to issue warnings as to accuracy

`.logExact` logical: used internally to flag the case where there are explicit formulae for the log version of density, cdf, and quantile function

`.lowerExact` logical: used internally to flag the case where there are explicit formulae for the lower tail version of cdf and quantile function

Extends

Class `"AbscontDistribution"`, directly.
Class `"UnivariateDistribution"`, by class `"AbscontDistribution"`.
Class `"Distribution"`, by class `"AbscontDistribution"`.

Methods

initialize `signature(.Object = "GPareto")`: initialize method.

shape `signature(object = "GPareto")`: wrapped access method for slot `shape` of slot `param`.

loc `signature(object = "GPareto")`: wrapped access method for slot `loc` of slot `param`.

location `signature(object = "GPareto")`: alias to `loc`, to support argument naming of package **VGAM**.

scale `signature(x = "GPareto")`: wrapped access method for slot `scale` of slot `param`.

shape<- `signature(object = "GPareto")`: wrapped replace method for slot `shape` of slot `param`.

loc<- `signature(object = "GPareto")`: wrapped replace method for slot `loc` of slot `param`.

location<- `signature(object = "GPareto")`: alias to `loc<-`, to support argument naming of package **VGAM**.

scale<- `signature(x = "GPareto")`: wrapped replace method for slot `scale` of slot `param`.

+ `signature(e1 = "GPareto", e2 = "numeric")`: exact method for this transformation — stays within this class.

***** `signature(e1 = "GPareto", e2 = "numeric")`: exact method for this transformation — stays within this class if `e2>0`.

E `signature(object = "GPareto", fun = "missing", cond = "missing")`: exact evaluation using explicit expressions.

var `signature(signature(x = "GPareto")`: exact evaluation using explicit expressions.

median `signature(signature(x = "GPareto")`: exact evaluation using explicit expressions.

IQR `signature(signature(x = "GPareto")`: exact evaluation using explicit expressions.

skewness `signature(signature(x = "GPareto")`: exact evaluation using explicit expressions.

kurtosis `signature(signature(x = "GPareto")`: exact evaluation using explicit expressions.

liesInSupport `signature(object = "GPareto", x = "numeric")`: checks if `x` lies in the support of the respective distribution.

Note

This class is based on the code provided by the package **evd** by A. G. Stephenson.

Author(s)

Nataliya Horbenko <nhorbenko@gmail.com>

References

Pickands, J. (1975) *Statistical inference using extreme order statistics. _Annals of Statistics_, *3*, 119-131.*

See Also

dgpd, AbscontDistribution-class

Examples

```
(P1 <- new("GPareto", loc = 0, scale = 1,shape = 0))
plot(P1)
shape(P1)
loc(P1)
scale(P1) <- 4
location(P1) <- 2 ## same as loc(P1) <- 2
shape(P1) <- -2 # may be negative
plot(P1)
```

GParetoFamily *Generating function for Generalized Pareto families*

Description

Generates an object of class "GParetoFamily" which represents a Generalized Pareto family.

Usage

```
GParetoFamily(loc = 0, scale = 1, shape = 0.5, of.interest = c("scale", "shape"),
       p = NULL, N = NULL, trafo = NULL, start0Est = NULL, withPos = TRUE,
       secLevel = 0.7,  withCentL2 = FALSE, withL2derivDistr  = FALSE,
       withMDE = FALSE, ..ignoreTrafo = FALSE)
```

Arguments

loc	real: known/fixed threshold/location parameter
scale	positive real: scale parameter
shape	positive real: shape parameter
of.interest	character: which parameters, transformations are of interest. possibilites are: "scale", "shape", "quantile", "expected loss", "expected shortfall"; a maximum number of two of these may be selected
p	real or NULL: probability needed for quantile and expected shortfall

N	real or NULL: expected frequency for expected loss
trafo	matrix or NULL: transformation of the parameter
start0Est	startEstimator — if NULL medkMADhybr is used
withPos	logical of length 1: Is shape restricted to positive values?
secLevel	a numeric of length 1: In the ideal GEV model, for each observastion X_i, the expression $1 + \frac{\text{shape}(X_i - \text{loc})}{\text{scale}}$ must be positive, which in principle could be attacked by a single outlier. Hence for sample size n we allow for εn violations, interpreting the violations as outliers. Here $\varepsilon =$ secLevel$/\sqrt{n}$.
withCentL2	logical: shall L2 derivative be centered by substracting the E()? Defaults to FALSE, but higher accuracy can be achieved when set to TRUE.
withL2derivDistr	logical: shall the distribution of the L2 derivative be computed? Defaults to FALSE (to speed up computations).
withMDE	logical: should Minimum Distance Estimators be used to find a good starting value for the parameter search? Defaults to FALSE (to speed up computations).
..ignoreTrafo	logical: only used internally in kStepEstimator; do not change this.

Details

The slots of the corresponding L2 differentiable parameteric family are filled.

Value

Object of class "GParetoFamily"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>
Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>
Nataliya Horbenko <nhorbenko@gmail.com>

References

Kohl, M. (2005) *Numerical Contributions to the Asymptotic Theory of Robustness*. Bayreuth: Dissertation. https://epub.uni-bayreuth.de/id/eprint/839/2/DissMKohl.pdf.

Kohl, M., Ruckdeschel, P., and Rieder, H. (2010): Infinitesimally Robust Estimation in General Smoothly Parametrized Models. *Stat. Methods Appl.*, **19**, 333-354. doi:10.1007/s1026001001330.

Ruckdeschel, P. and Horbenko, N. (2013): Optimally-Robust Estimators in Generalized Pareto Models. *Statistics*. **47**(4), 762-791. doi:10.1080/02331888.2011.628022.

Ruckdeschel, P. and Horbenko, N. (2012): Yet another breakdown point notion: EFSBP –illustrated at scale-shape models. *Metrika*, **75**(8), 1025–1047. doi:10.1007/s0018401103664.

See Also

L2ParamFamily-class, GPareto

Examples

```
(G1 <- GParetoFamily())
FisherInfo(G1)
checkL2deriv(G1)
```

GParetoParameter-class

Parameter of generalized Pareto distributions

Description

The class of the parameter of generalized Pareto distribution.

Objects from the Class

Objects can be created by calls of the form `new("GParetoParameter", ...)`.

Slots

`loc` real number: location parameter of a generalized Pareto distribution.

`scale` real number: scale parameter of a generalized Pareto distribution.

`shape` real number: shape parameter of a generalized Pareto distribution.

`name` default name is "parameter of a GPareto distribution".

Extends

Class `"Parameter"`, directly.
Class `"OptionalParameter"`, by class `"Parameter"`.

Methods

loc `signature(object = "GParetoParameter")`: access method for slot `loc`.

location `signature(object = "GParetoParameter")`: alias to `loc`, to support argument naming of package **VGAM**.

scale `signature(object = "GParetoParameter")`: access method for slot `scale`.

shape `signature(object = "GParetoParameter")`: access method for slot `shape`.

loc<- `signature(object = "GParetoParameter")`: replace method for slot `loc`.

location<- `signature(object = "GParetoParameter")`: alias to `loc<-`, to support argument naming of package **VGAM**.

shape<- `signature(object = "GParetoParameter")`: replace method for slot `shape`.

shape<- `signature(object = "GParetoParameter")`: replace method for slot `shape`.

Author(s)

Nataliya Horbenko <nhorbenko@gmail.com>

See Also

GPareto-class, Parameter-class

Examples

```
P <- new("GParetoParameter")
loc(P)
## same as
location(P)
scale(P)
shape(P)

scale(P) <- 2
loc(P) <- -5
shape(P) <- -1 # may be negative
P
```

Gumbel *Generating function for Gumbel-class*

Description

Generates an object of class "Gumbel".

Usage

```
Gumbel(loc = 0, scale = 1)
```

Arguments

loc	real number: location parameter of the Gumbel distribution.
scale	positive real number: scale parameter of the Gumbel distribution

Value

Object of class "Gumbel"

Note

The class "Gumbel" is based on the code provided by the package **evd**.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

See Also

Gumbel-class, rgumbel

Examples

```
(G1 <- Gumbel(loc = 1, scale = 2))
plot(G1)
loc(G1)
scale(G1)
loc(G1) <- -1
scale(G1) <- 2
plot(G1)

E(Gumbel()) # Euler's constant
E(G1, function(x){x^2})

## The function is currently defined as
function(loc = 0, scale = 1){
  new("Gumbel", loc = loc, scale = scale)
}
```

Gumbel-class *Gumbel distribution*

Description

The Gumbel cumulative distribution function with location parameter loc $= \mu$ and scale parameter scale $= \sigma$ is

$$F(x) = \exp(-\exp[-(x-\mu)/\sigma])$$

for all real x, where $\sigma > 0$; c.f. rgumbel. This distribution is also known as extreme value distribution of type I; confer Chapter~22 of Johnson et al. (1995).

Objects from the Class

Objects can be created by calls of the form new("Gumbel", loc, scale). More frequently they are created via the generating function Gumbel.

Slots

img Object of class "Reals".

param Object of class "GumbelParameter".

r rgumbel

d dgumbel

p pgumbel

q qgumbel

gaps (numeric) matrix or NULL

`.withArith` logical: used internally to issue warnings as to interpretation of arithmetics

`.withSim` logical: used internally to issue warnings as to accuracy

`.logExact` logical: used internally to flag the case where there are explicit formulae for the log version of density, cdf, and quantile function

`.lowerExact` logical: used internally to flag the case where there are explicit formulae for the lower tail version of cdf and quantile function

`Symmetry` object of class `"DistributionSymmetry"`; used internally to avoid unnecessary calculations.

Extends

Class `"AbscontDistribution"`, directly.
Class `"UnivariateDistribution"`, by class `"AbscontDistribution"`.
Class `"Distribution"`, by class `"AbscontDistribution"`.

Methods

initialize `signature(.Object = "Gumbel")`: initialize method.

loc `signature(object = "Gumbel")`: wrapped access method for slot `loc` of slot `param`.

scale `signature(x = "Gumbel")`: wrapped access method for slot `scale` of slot `param`.

loc<- `signature(object = "Gumbel")`: wrapped replace method for slot `loc` of slot `param`.

scale<- `signature(x = "Gumbel")`: wrapped replace method for slot `scale` of slot `param`.

`+ signature(e1 = "Gumbel", e2 = "numeric")`: result again of class `"Gumbel"`; exact.

`* signature(e1 = "Gumbel", e2 = "numeric")`: result again of class `"Gumbel"`; exact.

E `signature(object = "Gumbel", fun = "missing", cond = "missing")`: exact evaluation of expectation using explicit expressions.

var `signature(x = "Gumbel")`: exact evaluation of expectation using explicit expressions.

skewness `signature(x = "Gumbel")`: exact evaluation of expectation using explicit expressions.

kurtosis `signature(x = "Gumbel")`: exact evaluation of expectation using explicit expressions.

median `signature(x = "Gumbel")`: exact evaluation of expectation using explicit expressions.

IQR `signature(x = "Gumbel")`: exact evaluation of expectation using explicit expressions.

liesInSupport `signature(object = "Gumbel", x = "numeric")`: checks if `x` lies in the support of the respective distribution.

Note

This class is based on the code provided by the package **evd**.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

Johnson et al. (1995) *Continuous Univariate Distributions. Vol. 2. 2nd ed.* New York: Wiley.

See Also

rgumbel, AbscontDistribution-class

Examples

```
(G1 <- new("Gumbel", loc = 1, scale = 2))
plot(G1)
loc(G1)
scale(G1)
loc(G1) <- -1
scale(G1) <- 2
plot(G1)
```

GumbelLocationFamily *Generating function for Gumbel location families*

Description

Generates an object of class "L2LocationFamily" which represents a Gumbel location family.

Usage

```
GumbelLocationFamily(loc = 0, scale = 1, trafo)
```

Arguments

loc	location parameter
scale	scale parameter
trafo	function in param or matrix: transformation of the parameter

Details

The slots of the corresponding L2 differentiable parameteric family are filled.

Value

Object of class "L2LocationFamily"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

Kohl, M. (2005) *Numerical Contributions to the Asymptotic Theory of Robustness*. Bayreuth: Dissertation.

See Also

L2ParamFamily-class, Gumbel-class

Examples

```
##current implementation is:
theta <- 0
names(theta) <- "loc"
GL <- ParamFamily(name = "Gumbel location family",
          param = ParamFamParameter(name = "location parameter", main = theta),
          startPar = function(x,...) c(min(x),max(x)),
          distribution = Gumbel(loc = 0, scale = 1), ## scale known!
          modifyParam = function(theta){ Gumbel(loc = theta, scale = 1) },
          props = paste(c("The Gumbel location family is invariant under",
                    "the group of transformations 'g(x) = x + loc'",
                    "with location parameter 'loc'"), collapse = " "))
GL

(G1 <- GumbelLocationFamily())
plot(G1)
Map(L2deriv(G1)[[1]])
checkL2deriv(G1)
```

GumbelParameter-class *Paramter of Gumbel distributions*

Description

The class of the parameter of Gumbel distributions.

Objects from the Class

Objects can be created by calls of the form `new("GumbelParameter", ...)`.

Slots

`loc` real number: location parameter of a Gumbel distribution.

`scale` positive real number: scale parameter of a Gumbel distribution.

`name` default name is "parameter of a Gumbel distribution".

Extends

Class `"Parameter"`, directly.
Class `"OptionalParameter"`, by class `"Parameter"`.

Methods

loc `signature(object = "GumbelParameter")`: access method for slot `loc`.

scale `signature(x = "GumbelParameter")`: access method for slot `scale`.

loc<- `signature(object = "GumbelParameter")`: replace method for slot `loc`.

scale<- `signature(x = "GumbelParameter")`: replace method for slot `scale`.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

See Also

Gumbel-class, Parameter-class

Examples

```
new("GumbelParameter")
```

InternalEstimatorReturnClasses
Internal Estimator Return Classes in 'RobExtremes'

Description

S4 classes for return values of estimators in package **RobExtremes** defined for internal purposes.

Described classes

The S4 classes described here are `GPDEstimate`, `GEVEstimate`, `GPDMCEstimate`, `GEVMCEstimate`, `GPDMDEstimate`, `GEVMDEstimate`, `GPDLDEstimate`, `GEVLDEstimate`, `GPDkStepEstimate`, `GEVkStepEstimate`, `GPDORobEstimate`, `GEVORobEstimate`, `GPDML.ALEstimate`, `GEVML.ALEstimate`, `GPDCvMMD.ALEstimate`, `GEVCvMMD.ALEstimate`.

Objects from the Class

These classes are used internally to provide specific S4 methods for different estimators later on; thus, there are no generating functions.

Slots

All slots are inherited from parent classes.

Extends

Classes GPDEstimate, GEVEstimate extend class Estimate, directly.
Class GPDMCEstimate extends classes GPDEstimate, MCEstimate, directly.
Class GEVMCEstimate extends classes GEVEstimate, MCEstimate, directly.
Class GPDMDEstimate extends classes GPDEstimate, MDEstimate, directly.
Class GEVMDEstimate extends classes GEVEstimate, MDEstimate, directly.
Class GPDMCALEstimate extends classes GPDEstimate, MCALEstimate, directly.
Class GEVMCALEstimate extends classes GEVEstimate, MCALEstimate, directly.
Class GPDLDEstimate extends classes GPDEstimate, LDEstimate, directly.
Class GEVLDEstimate extends classes GEVEstimate, LDEstimate, directly.
Class GPDkStepEstimate extends classes GPDEstimate, kStepEstimate, directly.
Class GEVkStepEstimate extends classes GEVEstimate, kStepEstimate, directly.
Class GPDORobEstimate extends classes GPDkStepEstimate, ORobEstimate, directly.
Class GEVORobEstimate extends classes GEVkStepEstimate, ORobEstimate, directly.
Class GPDML.ALEstimate extends classes GPDEstimate, ML.ALEstimate, directly.
Class GEVML.ALEstimate extends classes GEVEstimate, ML.ALEstimate, directly.
Class GPDCvMMD.ALEstimate extends classes GPDEstimate, CvMMD.ALEstimate, directly.
Class GEVCvMMD.ALEstimate extends classes GEVEstimate, CvMMD.ALEstimate, directly.

Author(s)

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

See Also

Estimate-class, MCEstimate-class, kStepEstimate-class, LDEstimate-class

---

interpolateSn *Function to compute LD (location-dispersion) estimates*

Description

Function LDEstimator provides a general way to compute estimates for a given parametric family of probability measures (with a scale and shape parameter) which can be obtained by matching location and dispersion functionals against empirical counterparts.

Usage

```
getShapeGrid(gridsize=1000, centralvalue=0.7,
             withPos=TRUE, cutoff.at.0=1e-4, fac = 2)

getSnGrid(xiGrid = getShapeGrid(), PFam=GParetoFamily(), low=0,
                      upp=1.01, accuracy = 10000, GridFileName="SnGrid.Rdata",
                      withPrint = FALSE)
```

Arguments

gridsize	integer; the size of the grid to be created.
centralvalue	numeric of length 1: the central value of the grid (for details see below).
withPos	logical of length 1; are negative values for the shape forbidden?
cutoff.at.0	numeric of length 1: How close may we come to 0?
fac	a scaling factor used for the respective grid values (see below).
xiGrid	numeric; grid of shape values.
PFam	an object of class "ParamFamily". The parametric family at which to evaluate the LDEstimator; the respective (main) parameter must contain "scale" and "shape".
low	numeric; argument for Sn.
upp	numeric; argument for Sn.
accuracy	numeric; argument for Sn.
GridFileName	character; if GridFileName!="", the pure y-grid values are saved under this filename.
withPrint	logical of length 1: shall current shape value be printed out?

Details

getShapeGrid is a helper function to produce an unequally spaced grid of shape values xi, with the rationale that we need values close to some typical values more often than values at the border. The code starts with an equally spaced grid of size gridsize from 0.5 to 1-0.25/gridsize. This is reflected at 0.5, and a grid of respective quantiles of Norm(mean=centralvalue, sd=fac) is produced—with the heuristic rational that most estimators will be asymptotically normal around a typical value. If withPos is TRUE, negative values are cut off and replaced by respective higher quantiles of the corresponding normal; similarly, values to close to 0 are replaced by values between the cutoff value and the next admissible value and again by respective higher normal quantiles.

getSnGrid is a helper function to produce a grid of Sn values for a given grid of shape values and scale equal to 1 in a given shape-scale family. This result of this function can then be used to speed up calls to Sn (or to medSn) by providing particular methods for Sn. For an example of such a particular method see the body of getMethod("Sn", "GPareto") where object sng[["Generalized Pareto Family"]] is just the result of a call getSnGrid(xiGrid = getShapeGrid(), PFam=GParetoFamily()) which has been stored in the namespace of package distrMod.

Value

getShapeGrid	a numeric grid of xi-values.
getSnGrid	a grid, i.e.; a matrix with columns xi and Sn–the respective interpolation grid).

Author(s)

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

Examples

```
## (empirical) Data
getShapeGrid(50)
head(getShapeGrid(withPos=FALSE))

## Not run:
### code used for the grid stored in the namespace of distrMod:
getSnGrid()

## End(Not run)
```

---

ismevgpdgevdiag-methods

Methods for Diagnostic Functions in Package 'RobExtremes'

Description

We provide wrapper to the diagnostic plots `gpd.diag` and `gev.diag` of package **ismev**, as well as to profilers `gpd.prof`, `gpd.profxi` and `gev.prof`, `gev.profxi`.

Usage

```
gpd.diag(z,...)
## S4 method for signature 'gpd.fit'
gpd.diag(z)
## S4 method for signature 'GPDEstimate'
gpd.diag(z, npy = 365)
gev.diag(z)
## S4 method for signature 'gev.fit'
gev.diag(z)
## S4 method for signature 'GEVEstimate'
gev.diag(z)
gpd.prof(z,...)
## S4 method for signature 'gpd.fit'
gpd.prof(z, m, xlow, xup, npy = 365, conf = 0.95, nint = 100)
## S4 method for signature 'GPDEstimate'
gpd.prof(z, m, xlow, xup, npy = 365, conf = 0.95, nint = 100)
gev.prof(z,...)
## S4 method for signature 'gev.fit'
gev.prof(z, m, xlow, xup, conf = 0.95, nint = 100)
## S4 method for signature 'GEVEstimate'
gev.prof(z, m, xlow, xup, conf = 0.95, nint = 100)
gpd.profxi(z,...)
## S4 method for signature 'gpd.fit'
gpd.profxi(z,  xlow, xup, conf = 0.95, nint = 100)
## S4 method for signature 'GPDEstimate'
gpd.profxi(z,  xlow, xup, npy = 365, conf = 0.95, nint = 100)
```

```
gev.profxi(z,...)
## S4 method for signature 'gev.fit'
gev.profxi(z, xlow, xup, conf = 0.95, nint = 100)
## S4 method for signature 'GEVEstimate'
gev.profxi(z, xlow, xup, conf = 0.95, nint = 100)
```

Arguments

z	an argument of class gpd.fit, gev.fit (recovering the original calling convention from package **ismev** or of class GEVFamily or GParetoFamily.
m	The return level (i.e.\ the profile likelihood is for the value that is exceeded with probability 1/m).
...	further parameters to be passed on the specific methods.
xlow, xup	The least and greatest value at which to evaluate the profile likelihood.
npy	The number of observations per year.
conf	The confidence coefficient of the plotted profile confidence interval.
nint	The number of points at which the profile likelihood is evaluated.

Details

We provide a coercing of our fits of S4-classes "GPDEstimate" and "GEVEstimate" to the (S3-)classes gpd.fit and gev.fit of package **ismev** (the latter being cast to an S4 class, internally, in our package.

Value

For gpd.fit, gev.fit (quoted from package **ismev**: For stationary models four plots are produced; a probability plot, a quantile plot, a return level plot and a histogram of data with fitted density.

For non-stationary models two plots are produced; a residual probability plot and a residual quantile plot.

For gpd.prof, gev.prof (quoted from package **ismev**:

A plot of the profile likelihood is produced, with a horizontal line representing a profile confidence interval with confidence coefficient conf.

Author(s)

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

References

ismev: An Introduction to Statistical Modeling of Extreme Values. R package version 1.39. https://CRAN.R-project.org/package=ismev; original S functions written by Janet E. Heffernan with R port and R documentation provided by Alec G. Stephenson. (2012).

Coles, S. (2001). *An introduction to statistical modeling of extreme values.* London: Springer.

Examples

```
if(require(ismev)){
  ## from ismev
  data(portpirie)
  data(rain)

  detach(package:ismev)
  ppfit <- ismev::gev.fit(portpirie[,2])
  gev.diag(ppfit)
  ##
  (mlE <- MLEstimator(portpirie[,2], GEVFamilyMuUnknown(withPos=FALSE)))
  gev.diag(mlE)


  ## not tested on CRAN because it takes some time...
  gev.prof(mlE, m = 10, 4.1, 5)
  gev.profxi(mlE, -0.3, 0.3)

  rnfit <- ismev::gpd.fit(rain,10)
  gpd.diag(rnfit)
  ##
  mlE2 <- MLEstimator(rain[rain>10], GParetoFamily(loc=10))
  gpd.diag(mlE2)

  gpd.prof(mlE2, m = 10, 55, 77)
  gpd.profxi(mlE2, -0.02, 0.02)

}
```

---

kMAD *Asymmetric Median of Absolute Deviations for Skewed Distributions*

Description

Function for the computation of asymmetric median absolute deviation (kMAD) It coincides with ordinary median absolute deviation (MAD) for $k = 1$.

Usage

```
kMAD(x,k,...)
## S4 method for signature 'numeric,numeric'
kMAD(x, k = 1, na.rm = TRUE,
                 eps = .Machine$double.eps, ... )
## S4 method for signature 'UnivariateDistribution,numeric'
kMAD(x, k = 1, up = NULL, ... )
```

Arguments

x	a numeric vector or a distribution.
k	numeric; tunning parameter for asymmetrical MAD; has to be of length 1 and larger than 1.
na.rm	logical; if TRUE then NA values are stripped from x before computation takes place.
eps	numeric; accuracy up to which to state equality of two numeric values
up	numeric; upper bound for search interval; important in distributions without left/right endpoint.
...	additional arguments for other functions; not used so far;

Details

For kMAD (asymmetrial MAD) is a root of the equation:

$$\mathrm{kMAD}(F,k) = \inf\{t > 0 \mid F(m+kt) - F(m-t) \geq 1/2\}$$

, where F is the cumulative distribution function, m is the median of F.

Author(s)

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>, Nataliya Horbenko <nhorbenko@gmail.com>

References

Ruckdeschel, P., Horbenko, N. (2010): Robustness Properties for Generalized Pareto Distributions. ITWM Report 182.

See Also

mad

Examples

```
x <- rnorm(100)
kMAD(x,k=10)
kMAD(Norm(),k=10)
```

LDEstimate-class *LDEstimate-class.*

Description

Class of Location Dispersion estimates.

Objects from the Class

Objects can be created by calls of the form `new("LDEstimate", ...)`. More frequently they are created via the generating function `LDEstimator`.

Slots

`name` Object of class `"character"`: name of the estimator.

`estimate` Object of class `"ANY"`: estimate.

`estimate.call` Object of class `"call"`: call by which estimate was produced.

`dispersion` Object of class `"numeric"`: the value of the fitted dispersion.

`location` Object of class `"numeric"`: the value of the fitted location.

`Infos` object of class `"matrix"` with two columns named `method` and `message`: additional informations.

`asvar` object of class `"OptionalMatrix"` which may contain the asymptotic (co)variance of the estimator.

`samplesize` object of class `"numeric"` — the samplesize at which the estimate was evaluated.

`nuis.idx` object of class `"OptionalNumeric"`: indices of `estimate` belonging to the nuisance part

`fixed` object of class `"OptionalNumeric"`: the fixed and known part of the parameter.

`trafo` object of class `"list"`: a list with components `fct` and `mat` (see below).

`untransformed.estimate` Object of class `"ANY"`: untransformed estimate.

`untransformed.asvar` object of class `"OptionalNumericOrMatrix"` which may contain the asymptotic (co)variance of the untransformed estimator.

`completecases` object of class `"logical"` — complete cases at which the estimate was evaluated.

Extends

Class `"Estimate"`, directly.

Methods

dispersion `signature(object = "LDEstimate")`: accessor function for slot `dispersion`.

location `signature(object = "LDEstimate")`: accessor function for slot `location`.

show `signature(object = "LDEstimate")`

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>,
Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

See Also

Estimate-class, LDEstimator, MCEstimator

Examples

```
## (empirical) Data
x <- rgamma(50, scale = 0.5, shape = 3)

## parametric family of probability measures
G <- GammaFamily(scale = 1, shape = 2)

(S <- medQn(x, G))
dispersion(S)
location(S)
```

LDEstimator *Function to compute LD (location-dispersion) estimates*

Description

Function LDEstimator provides a general way to compute estimates for a given parametric family of probability measures (with a scale and shape parameter) which can be obtained by matching location and dispersion functionals against empirical counterparts.

Usage

```
LDEstimator(x, loc.est, disp.est, loc.fctal, disp.fctal, ParamFamily,
            loc.est.ctrl = NULL, loc.fctal.ctrl=NULL,
            disp.est.ctrl = NULL, disp.fctal.ctrl=NULL,
            q.lo =1e-3, q.up=15, log.q =TRUE,
            name, Infos, asvar = NULL, nuis.idx = NULL,
            trafo = NULL, fixed = NULL, asvar.fct  = NULL, na.rm = TRUE,
            ..., .withEvalAsVar = FALSE, vdbg = FALSE)
medkMAD(x, ParamFamily, k=1, q.lo =1e-3, q.up=15, nuis.idx = NULL,
        trafo = NULL, fixed = NULL, asvar.fct = NULL, na.rm = TRUE,
        ..., .withEvalAsVar = FALSE, vdbg = FALSE)
medkMADhybr(x, ParamFamily, k=1, q.lo =1e-3, q.up=15, KK = 20, nuis.idx = NULL,
        trafo = NULL, fixed = NULL,  asvar.fct = NULL, na.rm = TRUE,
        ..., .withEvalAsVar = FALSE)
medSn(x, ParamFamily, q.lo =1e-3, q.up=10, nuis.idx = NULL,
      trafo = NULL, fixed = NULL, asvar.fct = NULL, na.rm = TRUE,
      accuracy = 100, ..., .withEvalAsVar = FALSE)
medQn(x, ParamFamily, q.lo =1e-3, q.up=15, nuis.idx = NULL,
```

```
trafo = NULL, fixed = NULL, asvar.fct = NULL, na.rm = TRUE,
..., .withEvalAsVar = FALSE)
```

Arguments

x	(empirical) data
ParamFamily	an object of class "ParamFamily". The parametric family at which to evaluate the LDEstimator; the respective (main) parameter must contain "scale" and "shape".
loc.est	a function expecting x (a numeric vector) as first argument; location estimator.
disp.est	a function expecting x (a numeric vector) as first argument; dispersion estimator; may only take non-negative values.
loc.fctal	a function expecting a distribution object as first argument; location functional.
disp.fctal	a function expecting a distribution object as first argument; dispersion functional; may only take non-negative values.
loc.est.ctrl	a list (or NULL); optional additional arguments for the location estimator.
disp.est.ctrl	a list (or NULL); optional additional arguments for the dispersion estimator.
loc.fctal.ctrl	a list (or NULL); optional additional arguments for the location functional.
disp.fctal.ctrl	a list (or NULL); optional additional arguments for the dispersion functional.
k	numeric; additional parameter for kMAD; must be positive and of length 1.
KK	numeric; Maximal number of trials with different k in medkMADhybr .
q.lo	numeric; lower bound for search intervall in shape parameter.
q.up	numeric; upper bound for search intervall in shape parameter.
log.q	logical; shall the zero search be done on log-scale?
name	optional name for estimator.
Infos	character: optional informations about estimator
asvar	optionally the asymptotic (co)variance of the estimator
nuis.idx	optionally the indices of the estimate belonging to nuisance parameter
fixed	optionally (numeric) the fixed part of the parameter
trafo	an object of class MatrixorFunction – a transformation for the main parameter
asvar.fct	optionally: a function to determine the corresponding asymptotic variance; if given, asvar.fct takes arguments L2Fam((the parametric model as object of class L2ParamFamily)) and param (the parameter value as object of class ParamFamParameter); arguments are called by name; asvar.fct may also process further arguments passed through the ... argument
na.rm	logical: if TRUE, the estimator is evaluated at complete.cases(x).
accuracy	numeric: argument to be passed on to Sn.
...	further arguments to be passed to location estimator and functional and dispersion estimator and functional.
vdbg	logical; if TRUE, debugging information is shown.
.withEvalAsVar	logical: shall slot asVar be evaluated (if asvar.fct is given) or just the call be returned?

Details

The arguments `loc.est`, `disp.est` (location and dispersion estimators) have to be functions with first argument `x` (a numeric vector with the empirical data) and additional, optional individual arguments to be passed on in the respective calls as lists `loc.est.ctrl`, `disp.est.ctrl`, and global additional arguments through the `...` argument. Similarly, arguments `loc.fctal`, `disp.fctal` (location and dispersion functionals) have to be functions with first argument an object of class `UnivariateDistribution`, and additional, optional individual arguments to be passed on in the respective calls as lists `loc.fctal.ctrl`, `disp.fctal.ctrl`, and global additional arguments again through the `...` argument. Uses `.LDMatch` internally.

Value

An object of S4-class `"Estimate"`.

Note

The values for `q.lo` and `q.up` are a bit delicate and have to be found, model by model, by try and error. As a rule, `medSn` is rather slow, as the evaluation of the `Sn` functional is quite expensive. So if `medSn` is the estimator of choice, it pays off, for a given shape-scale family, to evaluate `medSn` on a grid of shape-values (with scale 1) and then to use an interpolation techniques in a particular method to replace the default one for this shape-scale family. As an example, we have done so for the GPD family.

Author(s)

Nataliya Horbenko <nhorbenko@gmail.com>,
Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

References

Marazzi, A. and Ruffieux, C. (1999): The truncated mean of asymmetric distribution. *Computational Statistics and Data Analysis* **32**, 79-100.

Ruckdeschel, P. and Horbenko, N. (2013): Optimally-Robust Estimators in Generalized Pareto Models. *Statistics*. **47**(4), 762-791. doi:10.1080/02331888.2011.628022.

Ruckdeschel, P. and Horbenko, N. (2012): Yet another breakdown point notion: EFSBP –illustrated at scale-shape models. *Metrika*, **75**(8), 1025-1047. doi:10.1007/s0018401103664.

See Also

`ParamFamily-class`, `ParamFamily`, `Estimate-class`

Examples

```
## (empirical) Data
set.seed(123)
x <- rgamma(50, scale = 0.5, shape = 3)
```

```
## parametric family of probability measures
G <- GammaFamily(scale = 1, shape = 2)

medQn(x = x, ParamFamily = G)
medSn(x = x, ParamFamily = G, q.lo = 0.5, q.up = 4)


## not tested on CRAN because it takes time...
## without speedup for Sn:
LDEstimator(x, loc.est = median, disp.est = Sn, loc.fctal = median,
            disp.fctal = getMethod("Sn","UnivariateDistribution"),
            ParamFamily = G, disp.est.ctrl = list(constant=1))

medkMAD(x = x, ParamFamily = G)
medkMADhybr(x = x, ParamFamily = G)

medkMAD(x = x, k=10, ParamFamily = G)

##not at all robust:
LDEstimator(x, loc.est = mean, disp.est = sd,
               loc.fctal = E, disp.fctal = sd,
            ParamFamily = G)
```

`movToRef-methods` *Methods for Functions moving from and to reference parameter in Package 'RobExtremes'*

Description

In `optIC` a gain in accuracy can be obtained when computing the optimally-robust ICs at a reference parameter of the model (instead of an arbtirary one). To this end, `moveL2Fam2RefParam` moved the model to the reference parameter and `moveICBackFromRefParam` moves the obtained optimal IC back to the original parameter.

Usage

```
moveL2Fam2RefParam(L2Fam, ...)
       moveICBackFromRefParam(IC, L2Fam,...)
```

Arguments

`L2Fam`	object of class `L2ParamFamily`
`IC`	IC of class `HampIC`
`...`	further arguments to be passed to particular methods

Details

`moveL2Fam2RefParam` and `moveICBackFromRefParam` are used internally in functions `robest` and `roptest` to compute the optimally robust influence function according to the arguments given to them.

Value

`moveL2Fam2RefParam`
: the L2 Family transformed to reference parameter.

`moveICBackFromRefParam`
: the backtransformed IC.

Methods

moveL2Fam2RefParam `signature(L2Fam = "L2ScaleShapeUnion")`: moves `L2Fam` to scale `1` (and, if existing location to `0`).

moveICBackFromRefParam `signature(IC = "IC", L2Fam = "L2ScaleShapeUnion")`: moves IC in `IC` back to original location and scale in `L2Fam` (and in addition changes Lagrange multipliers accordingly), rescaling risk where necessary.

moveICBackFromRefParam `signature(IC = "IC", L2Fam = "L2LocScaleShapeUnion")`: moves IC in `IC` back to original location and scale in `L2Fam` (and in addition changes Lagrange multipliers accordingly), rescaling risk where necessary.

Author(s)

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

See Also

robest,optIC, radiusMinimaxIC

Pareto *Generating function for Pareto-class*

Description

Generates an object of class `"Pareto"`.

Usage

```
Pareto(shape = 1, Min = 1)
```

Arguments

`shape`	positive real number: shape parameter of the Pareto distribution.
`Min`	positive real number: Min parameter of the Pareto distribution

Value

Object of class `"Pareto"`

Note

The class "Pareto" is based on the code provided by the package **actuar** by Vincent Goulet and Mathieu Pigeon.

Author(s)

Nataliya Horbenko <nhorbenko@gmail.com>

See Also

Pareto-class, dpareto1

Examples

```
(P1 <- Pareto(shape = 1, Min = 1))
plot(P1)

E(Pareto())
E(P1)
E(P1, function(x){x^2})
var(P1)
sd(P1)
median(P1)
IQR(P1)
mad(P1)
```

Pareto-class *Pareto distribution*

Description

[borrowed from **actuar**]:

The (Single-parameter) Pareto distribution with parameter shape $= \alpha$ has density:

$$f(x) = \frac{\alpha\theta^{\alpha}}{x^{\alpha+1}}$$

for $x > \theta$, $\alpha > 0$ and $\theta > 0$.

Although there appears to be two parameters, only shape is a true parameter. The value of min $= \theta$ must be set in advance.

Objects from the Class

Objects can be created by calls of the form new("Pareto", shape, Min). More frequently they are created via the generating function Pareto.

Slots

`img` Object of class `"Reals"`.

`param` Object of class `"ParetoParameter"`.

`r` rpareto1

`d` dpareto1

`p` ppareto1

`q` qpareto1

`gaps` (numeric) matrix or `NULL`

`.withArith` logical: used internally to issue warnings as to interpretation of arithmetics

`.withSim` logical: used internally to issue warnings as to accuracy

`.logExact` logical: used internally to flag the case where there are explicit formulae for the log version of density, cdf, and quantile function

`.lowerExact` logical: used internally to flag the case where there are explicit formulae for the lower tail version of cdf and quantile function

Extends

Class `"AbscontDistribution"`, directly.
Class `"UnivariateDistribution"`, by class `"AbscontDistribution"`.
Class `"Distribution"`, by class `"AbscontDistribution"`.

Methods

initialize `signature(.Object = "Pareto")`: initialize method.

shape `signature(object = "Pareto")`: wrapped access method for slot `shape` of slot `param`.

Min `signature(x = "Pareto")`: wrapped access method for slot `Min` of slot `param`.

scale `signature(x = "Pareto")`: wrapped access method for slot `Min` of slot `param`.

shape<- `signature(object = "Pareto")`: wrapped replace method for slot `shape` of slot `param`.

Min<- `signature(x = "Pareto")`: wrapped replace method for slot `Min` of slot `param`.

E `signature(object = "Pareto", fun = "missing", cond = "missing")`: exact evaluation using explicit expressions.

var `signature(signature(x = "Pareto")`: exact evaluation using explicit expressions.

median `signature(signature(x = "Pareto")`: exact evaluation using explicit expressions.

IQR `signature(signature(x = "Pareto")`: exact evaluation using explicit expressions.

skewness `signature(signature(x = "Pareto")`: exact evaluation using explicit expressions.

kurtosis `signature(signature(x = "Pareto")`: exact evaluation using explicit expressions.

***** `signature(e1 = "Pareto", e2 = "numeric")`: exact method for this transformation — stays within this class if `e2>0`.

liesInSupport `signature(object = "Pareto", x = "numeric")`: checks if `x` lies in the support of the respective distribution.

Note

This class is based on the code provided by the package **actuar** by Vincent Goulet and Mathieu Pigeon.

Author(s)

Nataliya Horbenko <nhorbenko@gmail.com>

References

Johnson et al. (1995) *Continuous Univariate Distributions. Vol. 2. 2nd ed.* New York: Wiley.
Klugman, S. A., Panjer, H. H. and Willmot, G. E. (2004), *Loss Models, From Data to Decisions, Second Edition*, Wiley.

See Also

dpareto1, AbscontDistribution-class

Examples

```
(P1 <- new("Pareto", shape = 1, Min = 2))
plot(P1)
shape(P1)
Min(P1)
shape(P1) <- 4
Min(P1) <- 2
plot(P1)
```

ParetoFamily *Generating function for Generalized Pareto families*

Description

Generates an object of class "ParetoFamily" which represents a Pareto family.

Usage

```
ParetoFamily(Min = 1, shape = 0.5, trafo = NULL, start0Est = NULL,
                    withCentL2 = FALSE)
```

Arguments

Min	real: known/fixed threshold/location parameter
shape	positive real: shape parameter
trafo	matrix or NULL: transformation of the parameter
start0Est	startEstimator — if NULL log(2)/log(median/Min) is used
withCentL2	logical: shall L2 derivative be centered by substracting the E()? Defaults to FALSE, but higher accuracy can be achieved when set to TRUE.

Details

The slots of the corresponding L2 differentiable parameteric family are filled.

Value

Object of class "ParetoFamily"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>
Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>
Nataliya Horbenko <nhorbenko@gmail.com>

References

Kohl, M. (2005) *Numerical Contributions to the Asymptotic Theory of Robustness*. Bayreuth: Dissertation. https://epub.uni-bayreuth.de/id/eprint/839/2/DissMKohl.pdf.

Kohl, M., Ruckdeschel, P., and Rieder, H. (2010): Infinitesimally Robust Estimation in General Smoothly Parametrized Models. *Stat. Methods Appl.*, **19**, 333-354. doi:10.1007/s1026001001330.

Ruckdeschel, P. and Horbenko, N. (2013): Optimally-Robust Estimators in Generalized Pareto Models. *Statistics*. **47**(4), 762-791. doi:10.1080/02331888.2011.628022.

Ruckdeschel, P. and Horbenko, N. (2012): Yet another breakdown point notion: EFSBP –illustrated at scale-shape models. *Metrika*, **75**(8), 1025–1047. doi:10.1007/s0018401103664.

See Also

L2ParamFamily-class, Pareto

Examples

```
(P1 <- ParetoFamily())
FisherInfo(P1)
checkL2deriv(P1)
```

ParetoParameter-class *Paramter of Pareto distributions*

Description

The class of the parameter of Pareto distributions.

Objects from the Class

Objects can be created by calls of the form `new("ParetoParameter", ...)`.

Slots

`shape` real number: shape parameter of a Pareto distribution.

`Min` positive real number: Min parameter of a Pareto distribution.

`name` default name is "parameter of a Pareto distribution".

Extends

Class `"Parameter"`, directly.
Class `"OptionalParameter"`, by class `"Parameter"`.

Methods

shape `signature(object = "ParetoParameter")`: access method for slot `shape`.

Min `signature(x = "ParetoParameter")`: access method for slot `Min`.

scale `signature(x = "ParetoParameter")`: access method for slot `Min`.

shape<- `signature(object = "ParetoParameter")`: replace method for slot `shape`.

Min<- `signature(x = "ParetoParameter")`: replace method for slot `Min`.

Author(s)

Nataliya Horbenko <nhorbenko@gmail.com>

See Also

Pareto-class, Parameter-class

Examples

```
(P1 <- new("ParetoParameter"))
Min(P1)
shape(P1)

Min(P1) <- 3
shape(P1) <- 4
P1
```

PickandsEstimator *Function to compute Pickands estimates for the GPD and GEVD*

Description

Function PickandsEstimator computes Pickands estimator (for the GPD and GEVD) at real data and returns an object of class Estimate.

Usage

```
PickandsEstimator(x, ParamFamily=GParetoFamily(), alpha=2,
            name, Infos, nuis.idx = NULL,
            trafo = NULL, fixed = NULL, na.rm = TRUE,
            ...)
.PickandsEstimator(x, alpha=2, GPD.l = TRUE)
```

Arguments

x	(empirical) data
alpha	numeric > 1; determines the variant of the Pickands-Estimator based on matching the empirical quantiles to levels $a_1 = 1 - 1/\alpha$ and $a_2 = 1 - 1/\alpha^2$ (in the GPD case) resp. $a_1 = \exp(-1/\alpha)$ and $a_1 = \exp(-1/\alpha^2)$ (in the GEVD case) against the population counter parts. The "classical" Pickands Estimator building up on the median is obtained for alpha=2 for the GPD and for alpha = 1/log(2) for the GEVD. If alpha is missing we set it to the optimal value (see note below).
ParamFamily	an object of class "GParetoFamily" or "GEVFamily".
name	optional name for estimator.
Infos	character: optional informations about estimator
nuis.idx	optionally the indices of the estimate belonging to nuisance parameter
fixed	optionally (numeric) the fixed part of the parameter
trafo	an object of class MatrixorFunction – a transformation for the main parameter
na.rm	logical: if TRUE, the estimator is evaluated at complete.cases(x).
...	not yet used.
GPD.l	logical: if TRUE the variant for GPD is used, else for GEVD.

Details

The actual work is done in .PickandsEstimator. The wrapper PickandsEstimator pre-treats the data, and constructs a respective Estimate object.

Value

`.PickandsEstimator`
: A numeric vector of length 2 with components named `scale` and `shape`.

`PickandsEstimator`
: An object of S4-class `"Estimate"`.

Note

The scale estimate we use, i.e., with scale = β and shape = ξ, we estimate scale by $\beta = \xi a_1/(\alpha^\xi - 1)$, differs from the one given in the original reference, where it was $\beta = \xi a_1^2/(a_2 - 2a_1)$. The one chosen here avoids taking differences $a_2 - 2a_1$ hence does not require $a_2 > 2a_1$; this leads to (functional) breakdown point (bdp)

$$\min(a_1, 1 - a_2, a_2 - a_1)$$

which is independent ξ, whereas the original setting leads to a bdp which is depending on ξ

$$\min(a_1, 1 - a_2, a_2 - 1 + (2\alpha^\xi - 1)^{-1/\xi}) \qquad \text{for GPD}$$

$$\min(a_1, 1 - a_2, a_2 - \exp(-(2\alpha^\xi - 1)^{-1/\xi})) \qquad \text{for GEVD}$$

. As a consequence our setting, the bdp-optimal choice of α for GDP is 2 leading to bdp $1/4$, and 2.248 for GEVD leading to bdp 0.180. For comparison, with the original setting, at $\xi = 0.7$, this gives optimal bdp's 0.070 and 0.060 for GPD and GEVD, respectively. The standard choice of α such that a_1 gives the median ($\alpha = 2$ in the GPD and $\alpha = 1/\log(2)$ in the GEVD) in our setting gives bdp's of $1/4$ and 0.119 for GPD and GEVD, respectively, and in the original setting, at $\xi = 0.7$, gives bdp's 0.064 and 0.023.

Author(s)

Nataliya Horbenko <nhorbenko@gmail.com>,
Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

References

Ruckdeschel, P. and Horbenko, N. (2012): Yet another breakdown point notion: EFSBP –illustrated at scale-shape models. *Metrika*, **75**(8), 1025–1047. doi:10.1007/s0018401103664.

Pickands, J. (1975): Statistical inference using extreme order statistics. *Ann. Stat.* **3**(1), 119–131.

See Also

`ParamFamily-class`, `ParamFamily`, `Estimate-class`

Examples

```
## (empirical) Data
set.seed(123)
x <- rgpd(50, scale = 0.5, shape = 3)
y <- rgev(50, scale = 0.5, shape = 3)
```

```
## parametric family of probability measures
P <- GParetoFamily(scale = 1, shape = 2)
G <- GEVFamily(scale = 1, shape = 2)
##
PickandsEstimator(x = x, ParamFamily = P)
PickandsEstimator(x = y, ParamFamily = G)
```

QuantileBCCEstimator *Function to compute QuantileBCC estimates for the Weibull Family*

Description

Function QuantileBCCEstimator computes QuantileBCC estimator (for the Weibull) at real data and returns an object of class Estimate.

Usage

```
QuantileBCCEstimator(x, p1 = 1/3, p2 = 2/3,
            name, Infos, nuis.idx = NULL,
            trafo = NULL, fixed = NULL, na.rm = TRUE,
            ...)
.QBCC(x, p1 = 1/3, p2 = 2/3)
```

Arguments

x	(empirical) data
p1, p2	levels of the quantiles; maximal breakdown point is achieved for $p1 = p2 - p1 = 1 - p2 = 1/3$ which is the default.
name	optional name for estimator.
Infos	character: optional informations about estimator
nuis.idx	optionally the indices of the estimate belonging to nuisance parameter
fixed	optionally (numeric) the fixed part of the parameter
trafo	an object of class MatrixorFunction – a transformation for the main parameter
na.rm	logical: if TRUE, the estimator is evaluated at complete.cases(x).
...	not yet used.

Details

The actual work is done in .QBCC. The wrapper QuantileBCCEstimator pre-treats the data, and constructs a respective Estimate object.

Value

.QuantileBCCEstimator
: A numeric vector of length 2 with components named scale and shape.

QuantileBCCEstimator
: An object of S4-class "Estimate".

Author(s)

Nataliya Horbenko <nhorbenko@gmail.com>,
Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

References

Boudt, K., Caliskan, D., Croux, C. (2011): Robust explicit estimators of Weibull parameters. *Metrika*, **73** (2), 187–209.

See Also

ParamFamily-class, ParamFamily, Estimate-class

Examples

```
## (empirical) Data
set.seed(123)
distroptions("withgaps"=FALSE)
x <- rweibull(50, scale = 0.5, shape = 3)
##
QuantileBCCEstimator(x = x)
```

RobExtremesConstants *Built-in Constants in package RobExtremes*

Description

Constants built into **RobExtremes**.

Usage

```
EULERMASCHERONICONSTANT
APERYCONSTANT
```

Details

RobExtremes has a small number of built-in constants.

The following constants are available:

- EULERMASCHERONICONSTANT: the Euler Mascheroni constant

$$\gamma = -\Gamma'(1)$$

  given in http://mathworld.wolfram.com/Euler-MascheroniConstant.html (48);
- APERYCONSTANT: the Apéry constant

$$\zeta(3) = \frac{5}{2}\left(\sum_{k\geq 1} \frac{(-1)^{k-1}}{k^3 \binom{2k}{k}}\right)$$

  as given in http://mathworld.wolfram.com/AperysConstant.html, equation (8);

These are implemented as variables in the **RobExtremes** name space taking appropriate values.

Examples

```
EULERMASCHERONICONSTANT
APERYCONSTANT
```

---

validParameter-methods

Methods for function validParameter in Package 'RobExtremes'

Description

Methods for function `validParameter` in package **RobExtremes** to check whether a new parameter (e.g. "proposed" by an optimization) is valid.

Usage

```
validParameter(object, ...)
## S4 method for signature 'GParetoFamily'
validParameter(object, param, tol=.Machine$double.eps)
## S4 method for signature 'WeibullFamily'
validParameter(object, param, tol=.Machine$double.eps)
## S4 method for signature 'GEVFamily'
validParameter(object, param, tol=.Machine$double.eps)
## S4 method for signature 'ParetoFamily'
validParameter(object, param, tol=.Machine$double.eps)
## S4 method for signature 'GEVFamilyMuUnknown'
validParameter(object, param,
           tol=.Machine$double.eps)
```

Arguments

`object`	an object of class `ParamFamily`
`param`	either a numeric vector or an object of class `ParamFamParameter`
`tol`	accuracy upto which the conditions have to be fulfilled
`...`	additional argument(s) for methods.

Details

method for signature

`GParetoFamily` checks if both parameters are finite by `is.finite`, if their length is 1 or 2 (e.g. if one features as nuisance parameter), and if both are strictly larger than 0 (upto argument `tol`)

`WeibullFamily` checks if both parameters are finite by `is.finite`, if their length is 1 or 2 (e.g. if one features as nuisance parameter), and if both are strictly larger than 0 (upto argument `tol`)

`GEVFamily` checks if both parameters are finite by `is.finite`, if their length is 1 or 2 (e.g. if one features as nuisance parameter), and if both are strictly larger than 0 (upto argument `tol`)

`GParetoFamily` checks if both parameters are finite by `is.finite`, if their length is 1 or 2 (e.g. if one features as nuisance parameter), and if both are strictly larger than 0 (upto argument `tol`)

`GEVFamilyMuUnknown` checks if all parameters are finite by `is.finite`, if their length is in 1,2,3 (e.g. if one features as nuisance parameter), and scale and shape both are strictly larger than 0 (upto argument `tol`)

Value

`logical` of length 1 — valid or not

Examples

```
G <- GParetoFamily()
validParameter(G, c(scale=0.1, shape=2))
validParameter(G, c(scale=-0.1, shape=-2))
```

---

var *Generic Functions for the Computation of Functionals*

Description

Generic functions for the computation of functionals on distributions.

Usage

```
IQR(x, ...)

## S4 method for signature 'Gumbel'
IQR(x)
## S4 method for signature 'GEV'
IQR(x)
## S4 method for signature 'GPareto'
IQR(x)
## S4 method for signature 'Pareto'
IQR(x)

median(x, ...)

## S4 method for signature 'Gumbel'
median(x)
## S4 method for signature 'GEV'
median(x)
## S4 method for signature 'GPareto'
median(x)
## S4 method for signature 'Pareto'
median(x)
```

```
var(x, ...)

## S4 method for signature 'Gumbel'
var(x, ...)
## S4 method for signature 'GEV'
var(x, ...)
## S4 method for signature 'GPareto'
var(x, ...)
## S4 method for signature 'Pareto'
var(x, ...)

skewness(x, ...)
## S4 method for signature 'Gumbel'
skewness(x, ...)
## S4 method for signature 'GEV'
skewness(x, ...)
## S4 method for signature 'GPareto'
skewness(x, ...)
## S4 method for signature 'Pareto'
skewness(x, ...)

kurtosis(x, ...)
## S4 method for signature 'Gumbel'
kurtosis(x, ...)
## S4 method for signature 'GEV'
kurtosis(x, ...)
## S4 method for signature 'GPareto'
kurtosis(x, ...)
## S4 method for signature 'Pareto'
kurtosis(x, ...)

Sn(x, ...)
## S4 method for signature 'ANY'
Sn(x,  ...)
## S4 method for signature 'UnivariateDistribution'
Sn(x, low = 0, upp = 1.01, accuracy = 1000, ...)
## S4 method for signature 'DiscreteDistribution'
Sn(x,  ...)
## S4 method for signature 'AffLinDistribution'
Sn(x,  ...)
## S4 method for signature 'Norm'
Sn(x,  ...)
## S4 method for signature 'GPareto'
Sn(x,  ...)
## S4 method for signature 'Pareto'
Sn(x,  ...)
## S4 method for signature 'GEV'
Sn(x,  ...)
```

```
## S4 method for signature 'Gammad'
Sn(x,  ...)
## S4 method for signature 'Weibull'
Sn(x,  ...)

Qn(x, ...)
## S4 method for signature 'ANY'
Qn(x,  ...)
## S4 method for signature 'UnivariateDistribution'
Qn(x, q00 = NULL, ...)
## S4 method for signature 'AffLinDistribution'
Qn(x, ...)
## S4 method for signature 'DiscreteDistribution'
Qn(x,  ...)
## S4 method for signature 'Norm'
Qn(x,  ...)
```

Arguments

x	object of class "UnivariateDistribution"
...	additional arguments to fun or E
q00	numeric or NULL: determines search interval (from -q00 to q00) for Qn; if NULL (default) q00 is set to 10*q(x)(3/4) internally.
low	numeric; lower bound for search interval for median(abs(x-Y)) where Y (a real constant) runs over the range of x; defaults to 0.
upp	numeric; upper bound for search interval for median(abs(x-Y)) where Y (a real constant) runs over the range of x; defaults to 1.01. Is used internally as upp*(mad(x)+abs(median(x)-Y)).
accuracy	numeric; number of grid points for Sn; defaults to 1000.

Value

The value of the corresponding functional at the distribution in the argument is computed.

Methods

Qn, signature(x = "Any"): interface to the **robustbase**-function Qn — see Qn.

Qn, signature(x = "UnivariateDistribution"): Qn of univariate distributions.

Qn, signature(x = "DiscreteDistribution"): Qn of discrete distributions.

Qn, signature(x = "AffLinDistribution"): abs(x@a) * Qn(x@X0)

Sn, signature(x = "Any"): interface to the **robustbase**-function Qn — see Sn.

Sn, signature(x = "UnivariateDistribution"): Sn of univariate distributions using pseudo-random variables (Thx to N. Horbenko).

Sn, signature(x = "DiscreteDistribution"): Sn of discrete distributions.

Sn, signature(x = "AffLinDistribution"): abs(x@a) * Sn(x@X0)

var, signature(x = "Gumbel"): exact evaluation using explicit expressions.

var, signature(x = "GPareto"): exact evaluation using explicit expressions.

var, signature(x = "GEV"): exact evaluation using explicit expressions.

var, signature(x = "Pareto"): exact evaluation using explicit expressions.

IQR, signature(x = "Gumbel"): exact evaluation using explicit expressions.

IQR, signature(x = "GPareto"): exact evaluation using explicit expressions.

IQR, signature(x = "GEV"): exact evaluation using explicit expressions.

IQR, signature(x = "Pareto"): exact evaluation using explicit expressions.

median, signature(x = "Gumbel"): exact evaluation using explicit expressions.

median, signature(x = "GEV"): exact evaluation using explicit expressions.

median, signature(x = "GPareto"): exact evaluation using explicit expressions.

median, signature(x = "Pareto"): exact evaluation using explicit expressions.

skewness, signature(x = "Gumbel"): exact evaluation using explicit expressions.

skewness, signature(x = "GEV"): exact evaluation using explicit expressions.

skewness, signature(x = "GPareto"): exact evaluation using explicit expressions.

skewness, signature(x = "Pareto"): exact evaluation using explicit expressions.

kurtosis, signature(x = "Gumbel"): exact evaluation using explicit expressions.

kurtosis, signature(x = "GEV"): exact evaluation using explicit expressions.

kurtosis, signature(x = "GPareto"): exact evaluation using explicit expressions.

kurtosis, signature(x = "Pareto"): exact evaluation using explicit expressions.

Sn, signature(x = "Norm"): exact evaluation using explicit expressions.

Sn, signature(x = "GPareto"): speeded up using interpolation grid.

Sn, signature(x = "GEV"): speeded up using interpolation grid.

Sn, signature(x = "Gammad"): speeded up using interpolation grid.

Sn, signature(x = "Weibull"): speeded up using interpolation grid.

Sn, signature(x = "Pareto"): speeded up using interpolation grid.

Qn, signature(x = "Norm"): exact evaluation using explicit expressions.

Caveat

If any of the packages **e1071**, **moments**, **fBasics** is to be used together with **distrEx** (or **RobExtremes**) the latter must be attached *after* any of the first mentioned. Otherwise `kurtosis()` and `skewness()` defined as *methods* in **distrEx** (or **RobExtremes**) may get masked.
To re-mask, you may use `kurtosis <- distrEx::kurtosis; skewness <- distrEx::skewness`.
See also `distrExMASK()`.

Author(s)

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

See Also

Var,
sd, var, IQR,
median, mad, sd,
Sn, Qn

Examples

```
# Variance of Exp(1) distribution
G <- GPareto()
var(G)

#median(Exp())
IQR(G)

## note the timing
system.time(print(Sn(GPareto(shape=0.5,scale=2))))

system.time(print(Sn(as(GPareto(shape=0.5,scale=2),"AbscontDistribution"))))
```

`WeibullFamily` *Generating function for Weibull family*

Description

Generates an object of class `"WeibullFamily"` which represents a Generalized Pareto family.

Usage

```
WeibullFamily(scale = 1, shape = 0.5, of.interest = c("scale", "shape"),
       p = NULL, N = NULL, trafo = NULL, start0Est = NULL, withPos = TRUE,
       withCentL2 = FALSE, withL2derivDistr  = FALSE, ..ignoreTrafo = FALSE)
```

Arguments

`scale`	positive real: scale parameter
`shape`	positive real: shape parameter
`of.interest`	character: which parameters, transformations are of interest. possibilites are: "scale", "shape", "quantile", "expected loss", "expected shortfall"; a maximum number of two of these may be selected
`p`	real or NULL: probability needed for quantile and expected shortfall
`N`	real or NULL: expected frequency for expected loss
`trafo`	matrix or NULL: transformation of the parameter
`start0Est`	startEstimator — if `NULL` `medkMADhybr` is used
`withPos`	logical of length 1: Is shape restricted to positive values?

withCentL2 logical: shall L2 derivative be centered by substracting the E()? Defaults to FALSE, but higher accuracy can be achieved when set to TRUE.

withL2derivDistr logical: shall the distribution of the L2 derivative be computed? Defaults to FALSE (to speeds up computations).

..ignoreTrafo logical: only used internally in kStepEstimator; do not change this.

Details

The slots of the corresponding L2 differentiable parameteric family are filled.

Value

Object of class "WeibullFamily"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>
Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>
Nataliya Horbenko <nhorbenko@gmail.com>

References

Kohl, M. (2005) *Numerical Contributions to the Asymptotic Theory of Robustness*. Bayreuth: Dissertation. https://epub.uni-bayreuth.de/id/eprint/839/2/DissMKohl.pdf.

Kohl, M., Ruckdeschel, P., and Rieder, H. (2010): Infinitesimally Robust Estimation in General Smoothly Parametrized Models. *Stat. Methods Appl.*, **19**, 333-354. doi:10.1007/s1026001001330.

Ruckdeschel, P. and Horbenko, N. (2013): Optimally-Robust Estimators in Generalized Pareto Models. *Statistics*. **47**(4), 762-791. doi:10.1080/02331888.2011.628022.

Ruckdeschel, P. and Horbenko, N. (2012): Yet another breakdown point notion: EFSBP –illustrated at scale-shape models. *Metrika*, **75**(8), 1025–1047. doi:10.1007/s0018401103664.

See Also

L2ParamFamily-class, Weibull-class

Examples

```
(G1 <- WeibullFamily())
FisherInfo(G1)
checkL2deriv(G1)
```

Index

∗ **E**
RobExtremes-package, 3
∗ **GEV distribution**
GEV, 19
GEVParameter-class, 26
InternalEstimatorReturnClasses, 38
∗ **GEV**
GEV, 19
GEV-class, 20
∗ **GPD distribution**
InternalEstimatorReturnClasses, 38
∗ **GPareto distribution**
GPareto, 27
GParetoParameter-class, 32
∗ **GPareto**
GPareto, 27
GPareto-class, 28
∗ **Generalized Pareto model**
GEVFamily, 22
GEVFamilyMuUnknown, 24
GParetoFamily, 30
∗ **Gumbel distribution**
Gumbel, 33
GumbelParameter-class, 37
∗ **Gumbel location model**
GumbelLocationFamily, 36
∗ **Gumbel**
Gumbel, 33
Gumbel-class, 34
∗ **IQR**
RobExtremes-package, 3
var, 61
∗ **LDEstimator**
RobExtremes-package, 3
∗ **Pareto distribution**
Pareto, 50
ParetoParameter-class, 54
∗ **Pareto model**
ParetoFamily, 53
∗ **Pareto**
Pareto, 50
Pareto-class, 51
∗ **Qn**
RobExtremes-package, 3
var, 61
∗ **S4 condition class**
RobExtremes-package, 3
∗ **S4 distribution class**
GEV-class, 20
GPareto-class, 28
Gumbel-class, 34
Pareto-class, 51
RobExtremes-package, 3
∗ **S4 parameter class**
GEVParameter-class, 26
GParetoParameter-class, 32
GumbelParameter-class, 37
InternalEstimatorReturnClasses, 38
ParetoParameter-class, 54
∗ **Sn**
RobExtremes-package, 3
var, 61
∗ **Weibull model**
WeibullFamily, 65
∗ **absolutely continuous distribution**
GEV, 19
GEV-class, 20
GPareto, 27
GPareto-class, 28
Gumbel, 33
Gumbel-class, 34
Pareto, 50
Pareto-class, 51
∗ **asymptotic risk**
getStartIC-methods, 17
movToRef-methods, 49
∗ **asymptotic variance**
asvarMedkMAD, 9

asvarPickands, 10
asvarQBCC, 12
* **classes**
getStartIC-methods, 17
LDEstimate-class, 45
movToRef-methods, 49
* **distribution**
E, 14
GEV, 19
GEV-class, 20
GEVParameter-class, 26
GPareto, 27
GPareto-class, 28
GParetoParameter-class, 32
Gumbel, 33
Gumbel-class, 34
GumbelParameter-class, 37
Pareto, 50
Pareto-class, 51
ParetoParameter-class, 54
var, 61
* **estimate**
LDEstimate-class, 45
* **estimator**
getCVaR, 16
kMAD, 43
* **expectation**
E, 14
* **extreme value distribution**
GEV-class, 20
GPareto-class, 28
Gumbel-class, 34
Pareto-class, 51
* **functional**
E, 14
RobExtremes-package, 3
var, 61
* **generating function**
GEV, 19
GEVParameter-class, 26
GPareto, 27
GParetoParameter-class, 32
Gumbel, 33
GumbelParameter-class, 37
Pareto, 50
ParetoParameter-class, 54
* **graphics**
ismevgpdgevdiag-methods, 41
* **influence curve**
checkmakeIC-methods, 13
* **integration**
E, 14
var, 61
* **kMAD**
RobExtremes-package, 3
* **kurtosis**
RobExtremes-package, 3
var, 61
* **location model**
GumbelLocationFamily, 36
* **location scale model**
GumbelParameter-class, 37
* **location**
GumbelParameter-class, 37
* **medQn**
RobExtremes-package, 3
* **medSn**
RobExtremes-package, 3
* **median**
RobExtremes-package, 3
var, 61
* **medkMAD**
RobExtremes-package, 3
* **methods**
.checkEstClassForParamFamily-methods, 8
E, 14
var, 61
* **models**
GEVFamily, 22
GEVFamilyMuUnknown, 24
GEVParameter-class, 26
GParetoFamily, 30
GParetoParameter-class, 32
GumbelLocationFamily, 36
GumbelParameter-class, 37
InternalEstimatorReturnClasses, 38
ParetoFamily, 53
ParetoParameter-class, 54
validParameter-methods, 60
WeibullFamily, 65
* **moment**
E, 14
* **package**
RobExtremes-package, 3
* **parameter**

GEVParameter-class, 26
GParetoParameter-class, 32
GumbelParameter-class, 37
ParetoParameter-class, 54
∗ **risk measure**
getCVaR, 16
∗ **risk**
getStartIC-methods, 17
movToRef-methods, 49
∗ **robust**
checkmakeIC-methods, 13
∗ **scale estimator**
kMAD, 43
∗ **scale**
GumbelParameter-class, 37
∗ **skewness**
RobExtremes-package, 3
var, 61
∗ **sysdata**
RobExtremesConstants, 59
∗ **univar**
interpolateSn, 39
LDEstimator, 46
PickandsEstimator, 56
QuantileBCCEstimator, 58
∗ **var**
RobExtremes-package, 3
var, 61
*,GEV,numeric-method *(GEV-class)*, 20
*,GPareto,numeric-method *(GPareto-class)*, 28
*,Gumbel,numeric-method *(Gumbel-class)*, 34
*,Pareto,numeric-method *(Pareto-class)*, 51
+,GEV,numeric-method *(GEV-class)*, 20
+,GPareto,numeric-method *(GPareto-class)*, 28
+,Gumbel,numeric-method *(Gumbel-class)*, 34
.LDMatch, *48*
.PickandsEstimator *(PickandsEstimator)*, 56
.QBCC *(QuantileBCCEstimator)*, 58
.checkEstClassForParamFamily *(.checkEstClassForParamFamily-methods)*, 8
.checkEstClassForParamFamily,GEVFamily,CvMMDEstimate-method *(.checkEstClassForParamFamily-methods)*, 8
.checkEstClassForParamFamily,GEVFamily,Estimate-method *(.checkEstClassForParamFamily-methods)*, 8
.checkEstClassForParamFamily,GEVFamily,LDEstimate-method *(.checkEstClassForParamFamily-methods)*, 8
.checkEstClassForParamFamily,GEVFamily,MCEstimate-method *(.checkEstClassForParamFamily-methods)*, 8
.checkEstClassForParamFamily,GEVFamily,MDEstimate-method *(.checkEstClassForParamFamily-methods)*, 8
.checkEstClassForParamFamily,GEVFamily,MLEstimate-method *(.checkEstClassForParamFamily-methods)*, 8
.checkEstClassForParamFamily,GEVFamily,ORobEstimate-method *(.checkEstClassForParamFamily-methods)*, 8
.checkEstClassForParamFamily,GEVFamily,kStepEstimate-metho *(.checkEstClassForParamFamily-methods)*, 8
.checkEstClassForParamFamily,GEVFamilyMuUnknown,CvMMDEstim *(.checkEstClassForParamFamily-methods)*, 8
.checkEstClassForParamFamily,GEVFamilyMuUnknown,Estimate-m *(.checkEstClassForParamFamily-methods)*, 8
.checkEstClassForParamFamily,GEVFamilyMuUnknown,LDEstimate *(.checkEstClassForParamFamily-methods)*, 8
.checkEstClassForParamFamily,GEVFamilyMuUnknown,MCEstimate *(.checkEstClassForParamFamily-methods)*, 8
.checkEstClassForParamFamily,GEVFamilyMuUnknown,MDEstimate *(.checkEstClassForParamFamily-methods)*, 8
.checkEstClassForParamFamily,GEVFamilyMuUnknown,MLEstimate *(.checkEstClassForParamFamily-methods)*, 8
.checkEstClassForParamFamily,GEVFamilyMuUnknown,ORobEstima *(.checkEstClassForParamFamily-methods)*, 8
.checkEstClassForParamFamily,GEVFamilyMuUnknown,kStepEstim *(.checkEstClassForParamFamily-methods)*, 8
.checkEstClassForParamFamily,GParetoFamily,CvMMDEstimate-m

(.checkEstClassForParamFamily-methods), 8
.checkEstClassForParamFamily,GParetoFamily,EstimateMethod (.checkEstClassForParamFamily-methods), 8
.checkEstClassForParamFamily,GParetoFamily,LDEstimate-method (.checkEstClassForParamFamily-methods), 8
.checkEstClassForParamFamily,GParetoFamily,MCEstimate-method (.checkEstClassForParamFamily-methods), 8
.checkEstClassForParamFamily,GParetoFamily,MDEstimate-method (.checkEstClassForParamFamily-methods), 8
.checkEstClassForParamFamily,GParetoFamily,MLEstimate-method (.checkEstClassForParamFamily-methods), 8
.checkEstClassForParamFamily,GParetoFamily,ORobEstimate-method (.checkEstClassForParamFamily-methods), 8
.checkEstClassForParamFamily,GParetoFamily,kStepEstimate-method (.checkEstClassForParamFamily-methods), 8
.checkEstClassForParamFamily-methods, 8

APERYCONSTANT (RobExtremesConstants), 59
asvarMedkMAD, 9
asvarPickands, 10
asvarQBCC, 12

checkIC, *13*
checkIC,IC,GEVFamily-method (checkmakeIC-methods), 13
checkIC,IC,GEVFamilyMuUnknown-method (checkmakeIC-methods), 13
checkIC,IC,GParetoFamily-method (checkmakeIC-methods), 13
checkIC,IC,ParetoFamily-method (checkmakeIC-methods), 13
checkmakeIC-methods, 13

dgpd, *19*, *22*, *28*, *30*
dispersion (LDEstimate-class), 45
dispersion,LDEstimate-method (LDEstimate-class), 45
distrExIntegrate, *16*
distrExOptions, *15*
DistributionsIntegratingByQuantiles-class (E), 14
dpareto, *51*, *53*

E, 14
E,DistributionsIntegratingByQuantiles,function,missing-method (E), 14
E,GEV,function,missing-method (E), 14
E,GEV,missing,missing-method (E), 14
E,GPareto,function,missing-method (E), 14
E,GPareto,missing,missing-method (E), 14
E,Gumbel,missing,missing-method (E), 14
E,Pareto,function,missing-method (E), 14
E,Pareto,missing,missing-method (E), 14
E-methods (E), 14
EULERMASCHERONICONSTANT (RobExtremesConstants), 59

GammaFamily, *17*
getCVaR, 16
getDiagnostic, *15*
getEL (getCVaR), 16
getShapeGrid (interpolateSn), 39
getSnGrid (interpolateSn), 39
getStartIC (getStartIC-methods), 17
getStartIC,L2LocScaleShapeUnion,interpolRisk-method (getStartIC-methods), 17
getStartIC,L2ScaleShapeUnion,interpolRisk-method (getStartIC-methods), 17
getStartIC,ParetoFamily,interpolRisk-method (getStartIC-methods), 17
getStartIC-methods, 17
getVaR (getCVaR), 16
GEV, 19
GEV-class, 20
gev.diag (ismevgpdgevdiag-methods), 41
gev.diag,gev.fit-method (ismevgpdgevdiag-methods), 41
gev.diag,GEVEstimate-method (ismevgpdgevdiag-methods), 41
gev.diag-methods (ismevgpdgevdiag-methods), 41
gev.prof (ismevgpdgevdiag-methods), 41
gev.prof,gev.fit-method (ismevgpdgevdiag-methods), 41
gev.prof,GEVEstimate-method (ismevgpdgevdiag-methods), 41
gev.profxi (ismevgpdgevdiag-methods), 41

gev.profxi,gev.fit-method
(ismevgpdgevdiag-methods), 41
gev.profxi,GEVEstimate-method
(ismevgpdgevdiag-methods), 41
gev.profxi-methods
(ismevgpdgevdiag-methods), 41
GEVCvMMD.ALEstimate-class
(InternalEstimatorReturnClasses), 38
GEVEstimate-class
(InternalEstimatorReturnClasses), 38
GEVFamily, *17*, 22
GEVFamilyMuUnknown, 24
GEVkStepEstimate-class
(InternalEstimatorReturnClasses), 38
GEVLDEstimate-class
(InternalEstimatorReturnClasses), 38
GEVMCEstimate-class
(InternalEstimatorReturnClasses), 38
GEVMDEstimate-class
(InternalEstimatorReturnClasses), 38
GEVML.ALEstimate-class
(InternalEstimatorReturnClasses), 38
GEVORobEstimate-class
(InternalEstimatorReturnClasses), 38
GEVParameter-class, 26
GPareto, *23*, *25*, 27, *32*
GPareto-class, 28
GParetoFamily, *17*, 30
GParetoParameter-class, 32
gpd.diag (ismevgpdgevdiag-methods), 41
gpd.diag,gpd.fit-method
(ismevgpdgevdiag-methods), 41
gpd.diag,GPDEstimate-method
(ismevgpdgevdiag-methods), 41
gpd.diag-methods
(ismevgpdgevdiag-methods), 41
gpd.prof (ismevgpdgevdiag-methods), 41
gpd.prof,gpd.fit-method
(ismevgpdgevdiag-methods), 41
gpd.prof,GPDEstimate-method
(ismevgpdgevdiag-methods), 41
gpd.profxi (ismevgpdgevdiag-methods), 41
gpd.profxi,gpd.fit-method
(ismevgpdgevdiag-methods), 41
gpd.profxi,GPDEstimate-method
(ismevgpdgevdiag-methods), 41
gpd.profxi-methods
(ismevgpdgevdiag-methods), 41
GPDCvMMD.ALEstimate-class
(InternalEstimatorReturnClasses), 38
GPDEstimate-class
(InternalEstimatorReturnClasses), 38
GPDkStepEstimate-class
(InternalEstimatorReturnClasses), 38
GPDLDEstimate-class
(InternalEstimatorReturnClasses), 38
GPDMCEstimate-class
(InternalEstimatorReturnClasses), 38
GPDMDEstimate-class
(InternalEstimatorReturnClasses), 38
GPDML.ALEstimate-class
(InternalEstimatorReturnClasses), 38
GPDORobEstimate-class
(InternalEstimatorReturnClasses), 38
Gumbel, 33
Gumbel-class, 34
GumbelLocationFamily, 36
GumbelParameter-class, 37

initialize,GEV-method (GEV-class), 20
initialize,GPareto-method
(GPareto-class), 28
initialize,Gumbel-method
(Gumbel-class), 34
initialize,Pareto-method
(Pareto-class), 51
InternalEstimatorReturnClasses, 38
InternalEstimatorReturnClasses-class
(InternalEstimatorReturnClasses), 38
interpolateSn, 39

IQR, *65*
IQR *(*var*)*, 61
IQR,GEV-method *(*var*)*, 61
IQR,GPareto-method *(*var*)*, 61
IQR,Gumbel-method *(*var*)*, 61
IQR,Pareto-method *(*var*)*, 61
IQR-methods *(*var*)*, 61
ismevgpdgevdiag-methods, 41

kMAD, *10*, 43, *47*
kMAD,numeric,numeric-method *(*kMAD*)*, 43
kMAD,UnivariateDistribution,numeric-method *(*kMAD*)*, 43
kMAD-methods *(*kMAD*)*, 43
kurtosis *(*var*)*, 61
kurtosis,GEV-method *(*var*)*, 61
kurtosis,GPareto-method *(*var*)*, 61
kurtosis,Gumbel-method *(*var*)*, 61
kurtosis,Pareto-method *(*var*)*, 61
kurtosis-methods *(*var*)*, 61

LDEstimate-class, 45
LDEstimator, *10*, *46*, 46
liesInSupport,GEV,numeric-method *(*GEV-class*)*, 20
liesInSupport,GPareto,numeric-method *(*GPareto-class*)*, 28
liesInSupport,Gumbel,numeric-method *(*Gumbel-class*)*, 34
liesInSupport,Pareto,numeric-method *(*Pareto-class*)*, 51
loc *(*GumbelParameter-class*)*, 37
loc,GEV-method *(*GEV-class*)*, 20
loc,GEVParameter-method *(*GEVParameter-class*)*, 26
loc,GPareto-method *(*GPareto-class*)*, 28
loc,GParetoParameter-method *(*GParetoParameter-class*)*, 32
loc,Gumbel-method *(*Gumbel-class*)*, 34
loc,GumbelParameter-method *(*GumbelParameter-class*)*, 37
loc<- *(*GumbelParameter-class*)*, 37
loc<-,GEV-method *(*GEV-class*)*, 20
loc<-,GEVParameter-method *(*GEVParameter-class*)*, 26
loc<-,GPareto-method *(*GPareto-class*)*, 28
loc<-,GParetoParameter-method *(*GParetoParameter-class*)*, 32
loc<-,Gumbel-method *(*Gumbel-class*)*, 34
loc<-,GumbelParameter-method *(*GumbelParameter-class*)*, 37
location,GEV-method *(*GEV-class*)*, 20
location,GEVParameter-method *(*GEVParameter-class*)*, 26
location,GPareto-method *(*GPareto-class*)*, 28
location,GParetoParameter-method *(*GParetoParameter-class*)*, 32
location,LDEstimate-method *(*LDEstimate-class*)*, 45
location<-,GEV-method *(*GEV-class*)*, 20
location<-,GEVParameter-method *(*GEVParameter-class*)*, 26
location<-,GPareto-method *(*GPareto-class*)*, 28
location<-,GParetoParameter-method *(*GParetoParameter-class*)*, 32

m1df, *16*
m2df, *16*
mad, *44*, *65*
makeIC, *13*
makeIC,IC,GEVFamily-method *(*checkmakeIC-methods*)*, 13
makeIC,IC,GEVFamilyMuUnknown-method *(*checkmakeIC-methods*)*, 13
makeIC,IC,GParetoFamily-method *(*checkmakeIC-methods*)*, 13
makeIC,IC,ParetoFamily-method *(*checkmakeIC-methods*)*, 13
MCEstimator, *46*
median, *65*
median *(*var*)*, 61
median,GEV-method *(*var*)*, 61
median,GPareto-method *(*var*)*, 61
median,Gumbel-method *(*var*)*, 61
median,Pareto-method *(*var*)*, 61
median-methods *(*var*)*, 61
medkMAD *(*LDEstimator*)*, 46
medkMADhybr, *31*, *65*
medkMADhybr *(*LDEstimator*)*, 46
medQn *(*LDEstimator*)*, 46
medSn *(*LDEstimator*)*, 46
Min,Pareto-method *(*Pareto-class*)*, 51
Min,ParetoParameter-method *(*ParetoParameter-class*)*, 54
Min<-,Pareto-method *(*Pareto-class*)*, 51

Min<-,ParetoParameter-method
(ParetoParameter-class), 54
moveICBackFromRefParam
(movToRef-methods), 49
moveICBackFromRefParam,IC,L2LocScaleShapeUnion-method
(movToRef-methods), 49
moveICBackFromRefParam,IC,L2ScaleShapeUnion-method
(movToRef-methods), 49
moveICBackFromRefParam-methods
(movToRef-methods), 49
moveL2Fam2RefParam (movToRef-methods),
49
moveL2Fam2RefParam,L2ScaleShapeUnion-method
(movToRef-methods), 49
moveL2Fam2RefParam-methods
(movToRef-methods), 49
movToRef-methods, 49

optIC, *19*, *50*

ParamFamily, *48*, *57*, *59*
Pareto, 50, *54*
Pareto-class, 51
ParetoFamily, 53
ParetoParameter-class, 54
PickandsEstimator, *11*, *22*, *24*, 56
print.riskMeasure (getCVaR), 16

Qn, *63*, *65*
Qn (var), 61
Qn,AffLinDistribution-method (var), 61
Qn,ANY-method (var), 61
Qn,DiscreteDistribution-method (var), 61
Qn,Norm-method (var), 61
Qn,UnivariateDistribution-method (var),
61
Qn-methods (var), 61
QuantileBCCEstimator, *12*, 58

radiusMinimaxIC, *19*, *50*
rgumbel, *34*, *36*
robest, *19*, *50*
RobExtremes (RobExtremes-package), 3
RobExtremes-package, 3
RobExtremesConstants, 59

scale,GEV-method (GEV-class), 20
scale,GEVParameter-method
(GEVParameter-class), 26
scale,GPareto-method (GPareto-class), 28
scale,GParetoParameter-method
(GParetoParameter-class), 32
scale,Gumbel-method (Gumbel-class), 34
scale,GumbelParameter-method
(GumbelParameter-class), 37
scale,Pareto-method (Pareto-class), 51
scale,ParetoParameter-method
(ParetoParameter-class), 54
scale<-,GEV-method (GEV-class), 20
scale<-,GEVParameter-method
(GEVParameter-class), 26
scale<-,GPareto-method (GPareto-class),
28
scale<-,GParetoParameter-method
(GParetoParameter-class), 32
scale<-,Gumbel-method (Gumbel-class), 34
scale<-,GumbelParameter-method
(GumbelParameter-class), 37
sd, *65*
shape (ParetoParameter-class), 54
shape,GEV-method (GEV-class), 20
shape,GEVParameter-method
(GEVParameter-class), 26
shape,GPareto-method (GPareto-class), 28
shape,GParetoParameter-method
(GParetoParameter-class), 32
shape,Pareto-method (Pareto-class), 51
shape,ParetoParameter-method
(ParetoParameter-class), 54
shape<- (ParetoParameter-class), 54
shape<-,GEV-method (GEV-class), 20
shape<-,GEVParameter-method
(GEVParameter-class), 26
shape<-,GPareto-method (GPareto-class),
28
shape<-,GParetoParameter-method
(GParetoParameter-class), 32
shape<-,Pareto-method (Pareto-class), 51
shape<-,ParetoParameter-method
(ParetoParameter-class), 54
show,LDEstimate-method
(LDEstimate-class), 45
showDiagnostic, *15*
skewness (var), 61
skewness,GEV-method (var), 61
skewness,GPareto-method (var), 61
skewness,Gumbel-method (var), 61

skewness,Pareto-method *(var)*, 61
skewness-methods *(var)*, 61
Sn, *40*, *47*, *63*, *65*
Sn *(var)*, 61
Sn,AffLinDistribution-method *(var)*, 61
Sn,ANY-method *(var)*, 61
Sn,DiscreteDistribution-method *(var)*, 61
Sn,Gammad-method *(var)*, 61
Sn,GEV-method *(var)*, 61
Sn,GPareto-method *(var)*, 61
Sn,Norm-method *(var)*, 61
Sn,Pareto-method *(var)*, 61
Sn,UnivariateDistribution-method *(var)*, 61
Sn,Weibull-method *(var)*, 61
Sn-methods *(var)*, 61

validParameter *(validParameter-methods)*, 60
validParameter,GEVFamily-method *(validParameter-methods)*, 60
validParameter,GEVFamilyMuUnknown-method *(validParameter-methods)*, 60
validParameter,GParetoFamily-method *(validParameter-methods)*, 60
validParameter,ParetoFamily-method *(validParameter-methods)*, 60
validParameter,WeibullFamily-method *(validParameter-methods)*, 60
validParameter-methods, 60
Var, *65*
var, 61, *65*
var,GEV-method *(var)*, 61
var,GPareto-method *(var)*, 61
var,Gumbel-method *(var)*, 61
var,Pareto-method *(var)*, 61
var-methods *(var)*, 61

WeibullFamily, *17*, 65