Package 'fExtremes'

October 13, 2022

Title Rmetrics - Modelling Extreme Events in Finance

Date 2022-08-06

Version 4021.83

Description Provides functions for analysing
and modelling extreme events in financial time Series. The
topics include: (i) data pre-processing, (ii) explorative
data analysis, (iii) peak over threshold modelling, (iv) block
maxima modelling, (v) estimation of VaR and CVaR, and (vi) the
computation of the extreme index.

Depends R (>= 2.15.1)

Imports fBasics, fGarch, graphics, methods, stats, timeDate,
timeSeries

Suggests RUnit, tcltk

LazyData yes

License GPL (>= 2)

URL https://www.rmetrics.org

BugReports https://r-forge.r-project.org/projects/rmetrics

NeedsCompilation no

Author Diethelm Wuertz [aut],
Tobias Setz [aut],
Yohan Chalabi [aut],
Paul J. Northrop [cre, ctb]

Maintainer Paul J. Northrop <p.northrop@ucl.ac.uk>

Repository CRAN

Date/Publication 2022-08-06 14:10:02 UTC

R topics documented:

`fExtremes-package` *Modelling Extreme Events in Finance*

Description

The Rmetrics "fExtremes" package is a collection of functions to analyze and model extreme events in Finance and Insurance.

Details

```
Package:    \tab fExtremes\cr
Type:       \tab Package\cr


License:    \tab GPL Version 2 or later\cr
Copyright:  \tab (c) 1999-2014 Rmetrics Assiciation\cr
URL:        \tab \url{https://www.rmetrics.org}
```

1 Introduction

The `fExtremes` package provides functions for analyzing and modeling extreme events in financial time Series. The topics include: (i) data pre-processing, (ii) explorative data analysis, (iii) peak over threshold modeling, (iv) block maxima modeling, (v) estimation of VaR and CVaR, and (vi) the computation of the extreme index.

2 Data and their Preprocessing

Data Sets:

Data sets used in the examples of the timeSeries packages.

Data Preprocessing:

These are tools for data preprocessing, including functions to separate data beyond a threshold value, to compute blockwise data like block maxima, and to decluster point process data.

```
blockMaxima     extracts block maxima from a vector or a time series
findThreshold   finds upper threshold for a given number of extremes
pointProcess    extracts peaks over Threshold from a vector or a time series
deCluster       de-clusters clustered point process data
```

2 Explorative Data Analysis of Extremes

This section contains a collection of functions for explorative data analysis of extreme values in financial time series. The tools include plot functions for emprical distributions, quantile plots, graphs exploring the properties of exceedances over a threshold, plots for mean/sum ratio and for the development of records. The functions are:

```
emdPlot         plots of empirical distribution function
qqparetoPlot    exponential/Pareto quantile plot
mePlot          plot of mean excesses over a threshold
mrlPlot         another variant, mean residual life plot
mxfPlot         another variant, with confidence intervals
msratioPlot     plot of the ratio of maximum and sum
```

```
recordsPlot     Record development compared with iid data
ssrecordsPlot   another variant, investigates subsamples
sllnPlot        verifies Kolmogorov's strong law of large numbers
lilPlot         verifies Hartman-Wintner's law of the iterated logarithm
```

```
xacfPlot        plots ACF of exceedances over a threshold
```

Parameter Fitting of Mean Excesses:

```
normMeanExcessFit    fits mean excesses with a normal density
ghMeanExcessFit      fits mean excesses with a GH density
hypMeanExcessFit     fits mean excesses with a HYP density
nigMeanExcessFit     fits mean excesses with a NIG density
ghtMeanExcessFit     fits mean excesses with a GHT density
```

3 GPD Peak over Threshold Modeling

GPD Distribution:

A collection of functions to compute the generalized Pareto distribution. The functions compute density, distribution function, quantile function and generate random deviates for the GPD. In addition functions to compute the true moments and to display the distribution and random variates changing parameters interactively are available.

```
dgpd            returns the density of the GPD distribution
pgpd            returns the probability function of the GPD
qgpd            returns quantile function of the GPD distribution
rgpd            generates random variates from the GPD distribution
gpdSlider       displays density or rvs from a GPD
```

GPD Moments:

```
gpdMoments      computes true mean and variance of GDP
```

GPD Parameter Estimation:

This section contains functions to fit and to simulate processes that are generated from the generalized Pareto distribution. Two approaches for parameter estimation are provided: Maximum likelihood estimation and the probability weighted moment method.

```
gpdSim          generates data from the GPD distribution
gpdFit          fits data to the GPD istribution
```

GPD print, plot and summary methods:

```
print           print method for a fitted GPD object
plot            plot method for a fitted GPD object
summary         summary method for a fitted GPD object
```

GDP Tail Risk:

The following functions compute tail risk under the GPD approach.

```
gpdQPlot        estimation of high quantiles
gpdQuantPlot    variation of high quantiles with threshold
gpdRiskMeasures prescribed quantiles and expected shortfalls
gpdSfallPlot    expected shortfall with confidence intervals
gpdShapePlot    variation of GPD shape with threshold
gpdTailPlot     plot of the GPD tail
```

4 GEV Block Maxima Modeling

GEV Distribution:

This section contains functions to fit and to simulate processes that are generated from the generalized extreme value distribution including the Frechet, Gumbel, and Weibull distributions.

```
dgev            returns density of the GEV distribution
pgev            returns probability function of the GEV
qgev            returns quantile function of the GEV distribution
rgev            generates random variates from the GEV distribution
gevSlider       displays density or rvs from a GEV
```

GEV Moments:

```
gevMoments      computes true mean and variance
```

GEV Parameter Estimation:

A collection to simulate and to estimate the parameters of processes generated from GEV distribution.

```
gevSim          generates data from the GEV distribution
gumbelSim       generates data from the Gumbel distribution
gevFit          fits data to the GEV distribution
gumbelFit       fits data to the Gumbel distribution


print           print method for a fitted GEV object
plot            plot method for a fitted GEV object
summary         summary method for a fitted GEV object
```

GEV MDA Estimation:

Here we provide Maximum Domain of Attraction estimators and visualize the results by a Hill plot and a common shape parameter plot from the Pickands, Einmal-Decker-deHaan, and Hill estimators.

```
hillPlot        shape parameter and Hill estimate of the tail index
shaparmPlot     variation of shape parameter with tail depth
```

GEV Risk Estimation:

```
gevrlevelPlot   k-block return level with confidence intervals
```

4 Value at Risk

Two functions to compute Value-at-Risk and conditional Value-at-Risk.

```
VaR             computes Value-at-Risk
CVaR            computes conditional Value-at-Risk
```

5 Extreme Index

A collection of functions to simulate time series with a known extremal index, and to estimate the extremal index by four different kind of methods, the blocks method, the reciprocal mean cluster size method, the runs method, and the method of Ferro and Segers.

```
thetaSim            simulates a time Series with known theta
blockTheta          computes theta from Block Method
clusterTheta        computes theta from Reciprocal Cluster Method
runTheta            computes theta from Run Method
ferrosegersTheta    computes theta according to Ferro and Segers


exindexPlot         calculatess and plots Theta(1,2,3)
exindexesPlot       calculates Theta(1,2) and plots Theta(1)
```

About Rmetrics

The `fExtremes` Rmetrics package is written for educational support in teaching "Computational Finance and Financial Engineering" and licensed under the GPL.

`DataPreprocessing` *Extremes Data Preprocessing*

Description

A collection and description of functions for data preprocessing of extreme values. This includes tools to separate data beyond a threshold value, to compute blockwise data like block maxima, and to decluster point process data.

The functions are:

`blockMaxima`	Block Maxima from a vector or a time series,
`findThreshold`	Upper threshold for a given number of extremes,
`pointProcess`	Peaks over Threshold from a vector or a time series,
`deCluster`	Declusters clustered point process data.

Usage

```
blockMaxima(x, block = c("monthly", "quarterly"), doplot = FALSE)
findThreshold(x, n = floor(0.05*length(as.vector(x))), doplot = FALSE)
pointProcess(x, u = quantile(x, 0.95), doplot = FALSE)
deCluster(x, run = 20, doplot = TRUE)
```

Arguments

block	the block size. A numeric value is interpreted as the number of data values in each successive block. All the data is used, so the last block may not contain block observations. If the data has a times attribute containing (in an object of class "POSIXct", or an object that can be converted to that class, see as.POSIXct) the times/dates of each observation, then block may instead take the character values "month", "quarter", "semester" or "year". By default monthly blocks from daily data are assumed.
doplot	a logical value. Should the results be plotted? By default TRUE.
n	a numeric value or vector giving number of extremes above the threshold. By default, n is set to an integer representing 5% of the data from the whole data set x.
run	parameter to be used in the runs method; any two consecutive threshold exceedances separated by more than this number of observations/days are considered to belong to different clusters.
u	a numeric value at which level the data are to be truncated. By default the threshold value which belongs to the 95% quantile, u=quantile(x,0.95).
x	a numeric data vector from which findThreshold and blockMaxima determine the threshold values and block maxima values. For the function deCluster the argument x represents a numeric vector of threshold exceedances with a times attribute which should be a numeric vector containing either the indices or the times/dates of each exceedance (if times/dates, the attribute should be an object of class "POSIXct" or an object that can be converted to that class; see as.POSIXct).

Details

Computing Block Maxima:

The function blockMaxima calculates block maxima from a vector or a time series, whereas the function blocks is more general and allows for the calculation of an arbitrary function FUN on blocks.

Finding Thresholds:

The function findThreshold finds a threshold so that a given number of extremes lie above. When the data are tied a threshold is found so that at least the specified number of extremes lie above.

De-Clustering Point Processes:

The function deCluster declusters clustered point process data so that Poisson assumption is more tenable over a high threshold.

Value

blockMaxima
returns a timeSeries object or a numeric vector of block maxima data.

`findThreshold`
returns a numeric value or vector of suitable thresholds.

`pointProcess`
returns a timeSeries object or a numeric vector of peaks over a threshold.

`deCluster`
returns a timeSeries object or a numeric vector for the declustered point process.

Author(s)

Some of the functions were implemented from Alec Stephenson's R-package `evir` ported from Alexander McNeil's S library `EVIS`, *Extreme Values in S*, some from Alec Stephenson's R-package `ismev` based on Stuart Coles code from his book, *Introduction to Statistical Modeling of Extreme Values* and some were written by Diethelm Wuertz.

References

Coles S. (2001); *Introduction to Statistical Modelling of Extreme Values*, Springer.

Embrechts, P., Klueppelberg, C., Mikosch, T. (1997); *Modelling Extremal Events*, Springer.

Examples

```
## findThreshold -
# Threshold giving (at least) fifty exceedances for Danish data:
library(timeSeries)
x <- as.timeSeries(data(danishClaims))
findThreshold(x, n = c(10, 50, 100))

## blockMaxima -
# Block Maxima (Minima) for left tail of BMW log returns:
BMW <- as.timeSeries(data(bmwRet))
colnames(BMW) <- "BMW.RET"
head(BMW)
x <- blockMaxima( BMW, block = 65)
head(x)
## Not run:
y <- blockMaxima(-BMW, block = 65)
head(y)
y <- blockMaxima(-BMW, block = "monthly")
head(y)
## End(Not run)

## pointProcess -
# Return Values above threshold in negative BMW log-return data:
PP = pointProcess(x = -BMW, u = quantile(as.vector(x), 0.75))
PP
nrow(PP)

## deCluster -
# Decluster the 200 exceedances of a particular
DC = deCluster(x = PP, run = 15, doplot = TRUE)
```

```
DC
nrow(DC)
```

ExtremeIndex *Extremal Index Estimation*

Description

A collection and description of functions to simulate time series with a known extremal index, and to estimate the extremal index by four different kind of methods, the blocks method, the reciprocal mean cluster size method, the runs method, and the method of Ferro and Segers.

The functions are:

`thetaSim`	Simulates a time Series with known theta,
`blockTheta`	Computes theta from Block Method,
`clusterTheta`	Computes theta from Reciprocal Cluster Method,
`runTheta`	Computes theta from Run Method,
`ferrosegersTheta`	Computes Theta according to Ferro and Segers,
`exindexPlot`	Calculate and Plot Theta(1,2,3),
`exindexesPlot`	Calculate Theta(1,2) and Plot Theta(1).

Usage

```
## S4 method for signature 'fTHETA'
show(object)

thetaSim(model = c("max", "pair"), n = 1000, theta = 0.5)

blockTheta(x, block = 22, quantiles = seq(0.950, 0.995, length = 10),
    title = NULL, description = NULL)
clusterTheta(x, block = 22, quantiles = seq(0.950, 0.995, length = 10),
    title = NULL, description = NULL)
runTheta(x, block = 22, quantiles = seq(0.950, 0.995, length = 10),
    title = NULL, description = NULL)
ferrosegersTheta(x, quantiles = seq(0.950, 0.995, length = 10),
    title = NULL, description = NULL)

exindexPlot(x, block = c("monthly", "quarterly"), start = 5, end = NA,
    doplot = TRUE, plottype = c("thresh", "K"), labels = TRUE, ...)

exindexesPlot(x, block = 22, quantiles = seq(0.950, 0.995, length = 10),
    doplot = TRUE, labels = TRUE, ...)
```

Arguments

block	[*Theta] - an integer value, the block size. Currently only integer specified block sizes are supported. [exindex*Plot] - the block size. Either "monthly", "quarterly" or an integer value. An integer value is interpreted as the number of data values in each successive block. The default value is "monthly" which corresponds for daily data to an approximately 22-day periods.
description	a character string which allows for a brief description.
doplot	a logical, should the results be plotted?
labels	whether or not axes should be labelled. If set to FALSE then user specified labels can be passed through the "..." argument.
model	[thetaSim] - a character string denoting the name of the model. Either "max" or "pair", the first representing the maximimum Frechet series, and the second the paired exponential series.
n	[thetaSim] - an integer value, the length of the time series to be generated.
object	an object of class "fTHETA" as returned by the functions *Theta.
plottype	[exindexPlot] - whether plot is to be by increasing threshold (thresh) or increasing K value (K).
quantiles	[exindexesPlot] - a numeric vector of quantile values.
start, end	[exindexPlot] - start is the lowest value of K at which to plot a point, and end the highest value; K is the number of blocks in which a specified threshold is exceeded.
theta	[thetaSim] - a numeric value between 0 and 1 setting the value of the extremal index for the maximum Frechet time series. (Not used in the case of the paired exponential series.)
title	a character string which allows for a project title.
x	a 'timeSeries' object or any other object which can be transformed by the function as.vector into a numeric vector. "monthly" and "quarterly" blocks require x to be an object of class "timeSeries".
...	additional arguments passed to the plot function.

Value

exindexPlot

returns a data frame of results with the following columns: N, K, un, theta2, and theta. A plot with K on the lower x-axis and threshold Values on the upper x-axis versus the extremal index is displayed.

exindexesPlot

returns a data.frame with four columns: thresholds, theta1, theta2, and theta3. A plot with quantiles on the x-axis and versus the extremal indexes is displayed.

Author(s)

Alexander McNeil, for parts of the `exindexPlot` function, and
Diethelm Wuertz for the `exindexesPlot` function.

References

Embrechts, P., Klueppelberg, C., Mikosch, T. (1997); *Modelling Extremal Events*, Springer. Chapter 8, 413–429.

See Also

`hillPlot`, `gevFit`.

Examples

```
## Extremal Index for the right and left tails
## of the BMW log returns:
   data(bmwRet)
   par(mfrow = c(2, 2), cex = 0.7)
   library(timeSeries)
   exindexPlot( as.timeSeries(bmwRet), block = "quarterly")
   exindexPlot(-as.timeSeries(bmwRet), block = "quarterly")

## Extremal Index for the right and left tails
## of the BMW log returns:
   exindexesPlot( as.timeSeries(bmwRet), block = 65)
   exindexesPlot(-as.timeSeries(bmwRet), block = 65)
```

ExtremesData *Explorative Data Analysis*

Description

A collection and description of functions for explorative data analysis. The tools include plot functions for emprical distributions, quantile plots, graphs exploring the properties of exceedances over a threshold, plots for mean/sum ratio and for the development of records.

The functions are:

`emdPlot`	Plot of empirical distribution function,
`qqparetoPlot`	Exponential/Pareto quantile plot,
`mePlot`	Plot of mean excesses over a threshold,
`mrlPlot`	another variant, mean residual life plot,
`mxfPlot`	another variant, with confidence intervals,
`msratioPlot`	Plot of the ratio of maximum and sum,
`recordsPlot`	Record development compared with iid data,
`ssrecordsPlot`	another variant, investigates subsamples,
`sllnPlot`	verifies Kolmogorov's strong law of large numbers,

lilPlot	verifies Hartman-Wintner's law of the iterated logarithm,
xacfPlot	ACF of exceedances over a threshold,
normMeanExcessFit	fits mean excesses with a normal density,
ghMeanExcessFit	fits mean excesses with a GH density,
hypMeanExcessFit	fits mean excesses with a HYP density,
nigMeanExcessFit	fits mean excesses with a NIG density,
ghtMeanExcessFit	fits mean excesses with a GHT density.

Usage

```
emdPlot(x, doplot = TRUE, plottype = c("xy", "x", "y", " "),
    labels = TRUE, ...)

qqparetoPlot(x, xi = 0, trim = NULL, threshold = NULL, doplot = TRUE,
    labels = TRUE, ...)

mePlot(x, doplot = TRUE, labels = TRUE, ...)
mrlPlot(x, ci = 0.95, umin = mean(x), umax = max(x), nint = 100, doplot = TRUE,
     plottype = c("autoscale", ""), labels = TRUE, ...)
mxfPlot(x, u = quantile(x, 0.05), doplot = TRUE, labels = TRUE, ...)

msratioPlot(x, p = 1:4, doplot = TRUE, labels = TRUE, ...)

recordsPlot(x, ci = 0.95, doplot = TRUE, labels = TRUE, ...)
ssrecordsPlot(x, subsamples = 10, doplot = TRUE, plottype = c("lin", "log"),
    labels = TRUE, ...)

sllnPlot(x, doplot = TRUE, labels = TRUE, ...)
lilPlot(x, doplot = TRUE, labels = TRUE, ...)

xacfPlot(x, u = quantile(x, 0.95), lag.max = 15, doplot = TRUE,
    which = c("all", 1, 2, 3, 4), labels = TRUE, ...)

normMeanExcessFit(x, doplot = TRUE, trace = TRUE, ...)
ghMeanExcessFit(x, doplot = TRUE, trace = TRUE, ...)
hypMeanExcessFit(x, doplot = TRUE, trace = TRUE, ...)
nigMeanExcessFit(x, doplot = TRUE, trace = TRUE, ...)
ghtMeanExcessFit(x, doplot = TRUE, trace = TRUE, ...)
```

Arguments

ci	[recordsPlot] - a confidence level. By default 0.95, i.e. 95%.
doplot	a logical value. Should the results be plotted? By default TRUE.
labels	a logical value. Whether or not x- and y-axes should be automatically labelled and a default main title should be added to the plot. By default TRUE.
lag.max	[xacfPlot] - maximum number of lags at which to calculate the autocorrelation functions.

The default value is 15.

`nint` [mrlPlot] -
the number of intervals, see `umin` and `umax`. The default value is 100.

`p` [msratioPlot] -
the power exponents, a numeric vector. By default a sequence from 1 to 4 in unit integer steps.

`plottype` [emdPlot] -
which axes should be on a log scale: `"x"` x-axis only; `"y"` y-axis only; `"xy"` both axes; `""` neither axis.
[msratioPlot] -
a logical, if set to `"autoscale"`, then the scale of the plots are automatically determined, any other string allows user specified scale information through the `...` argument.
[ssrecordsPlot] -
one from two options can be select either `"lin"` or `"log"`. The default creates a linear plot.

`subsamples` [ssrecordsPlot] -
the number of subsamples, by default 10, an integer value.

`threshold, trim`
[qPlot][xacfPlot] -
a numeric value at which data are to be left-truncated, value at which data are to be right-truncated or the threshold value, by default 95%.

`trace` a logical flag, by default `TRUE`. Should the calculations be traced?

`u` a numeric value at which level the data are to be truncated. By default the threshold value which belongs to the 95% quantile, `u=quantile(x,0.95)`.

`umin, umax` [mrlPlot] -
range of threshold values. If `umin` and/or `umax` are not available, then by default they are set to the following values: `umin=mean(x)` and `umax=max(x)`.

`which` [xacfPlot] -
a numeric or character value, if `which="all"` then all four plots are displayed, if `which` is an integer between one and four, then the first, second, third or fourth plot will be displayed.

`x, y` numeric data vectors or in the case of x an object to be plotted.

`xi` the shape parameter of the generalized Pareto distribution.

`...` additional arguments passed to the FUN or plot function.

Details

Empirical Distribution Function:

The function `emdPlot` is a simple explanatory function. A straight line on the double log scale indicates Pareto tail behaviour.

Quantile–Quantile Pareto Plot:

qqparetoPlot creates a quantile-quantile plot for threshold data. If xi is zero the reference distribution is the exponential; if xi is non-zero the reference distribution is the generalized Pareto with that parameter value expressed by xi. In the case of the exponential, the plot is interpreted as follows: Concave departures from a straight line are a sign of heavy-tailed behaviour, convex departures show thin-tailed behaviour.

Mean Excess Function Plot:

Three variants to plot the mean excess function are available: A sample mean excess plot over increasing thresholds, and two mean excess function plots with confidence intervals for discrimination in the tails of a distribution. In general, an upward trend in a mean excess function plot shows heavy-tailed behaviour. In particular, a straight line with positive gradient above some threshold is a sign of Pareto behaviour in tail. A downward trend shows thin-tailed behaviour whereas a line with zero gradient shows an exponential tail. Here are some hints: Because upper plotting points are the average of a handful of extreme excesses, these may be omitted for a prettier plot. For mrlPlot and mxfPlot the upper tail is investigated; for the lower tail reverse the sign of the data vector.

Plot of the Maximum/Sum Ratio:

The ratio of maximum and sum is a simple tool for detecting heavy tails of a distribution and for giving a rough estimate of the order of its finite moments. Sharp increases in the curves of a msratioPlot are a sign for heavy tail behaviour.

Plot of the Development of Records:

These are functions that investigate the development of records in a dataset and calculate the expected behaviour for iid data. recordsPlot counts records and reports the observations at which they occur. In addition subsamples can be investigated with the help of the function ssrecordsPlot.

Plot of Kolmogorov's and Hartman-Wintner's Laws:

The function sllnPlot verifies Kolmogorov's strong law of large numbers, and the function lilPlot verifies Hartman-Wintner's law of the iterated logarithm.

ACF Plot of Exceedances over a Threshold:

This function plots the autocorrelation functions of heights and distances of exceedances over a threshold.

Value

The functions return a plot.

Note

The plots are labeled by default with a x-label, a y-label and a main title. If the argument labels is set to FALSE neither a x-label, a y-label nor a main title will be added to the graph. To add user

defined label strings just use the function `title(xlab="\dots", ylab="\dots", main="\dots")`.

Author(s)

Some of the functions were implemented from Alec Stephenson's R-package `evir` ported from Alexander McNeil's S library `EVIS`, *Extreme Values in S*, some from Alec Stephenson's R-package `ismev` based on Stuart Coles code from his book, *Introduction to Statistical Modeling of Extreme Values* and some were written by Diethelm Wuertz.

References

Coles S. (2001); *Introduction to Statistical Modelling of Extreme Values*, Springer.

Embrechts, P., Klueppelberg, C., Mikosch, T. (1997); *Modelling Extremal Events*, Springer.

Examples

```
## Danish fire insurance data:
   data(danishClaims)
   library(timeSeries)
   danishClaims = as.timeSeries(danishClaims)

## emdPlot -
   # Show Pareto tail behaviour:
   par(mfrow = c(2, 2), cex = 0.7)
   emdPlot(danishClaims)

## qqparetoPlot -
   # QQ-Plot of heavy-tailed Danish fire insurance data:
   qqparetoPlot(danishClaims, xi = 0.7)

## mePlot -
   # Sample mean excess plot of heavy-tailed Danish fire:
   mePlot(danishClaims)

## ssrecordsPlot -
   # Record fire insurance losses in Denmark:
   ssrecordsPlot(danishClaims, subsamples = 10)
```

GevDistribution *Generalized Extreme Value Distribution*

Description

Density, distribution function, quantile function, random number generation, and true moments for the GEV including the Frechet, Gumbel, and Weibull distributions.

The GEV distribution functions are:

dgev	density of the GEV distribution,
pgev	probability function of the GEV distribution,
qgev	quantile function of the GEV distribution,
rgev	random variates from the GEV distribution,
gevMoments	computes true mean and variance,
gevSlider	displays density or rvs from a GEV.

Usage

```
dgev(x, xi = 1, mu = 0, beta = 1, log = FALSE)
pgev(q, xi = 1, mu = 0, beta = 1, lower.tail = TRUE)
qgev(p, xi = 1, mu = 0, beta = 1, lower.tail = TRUE)
rgev(n, xi = 1, mu = 0, beta = 1)

gevMoments(xi = 0, mu = 0, beta = 1)

gevSlider(method = c("dist", "rvs"))
```

Arguments

log	a logical, if TRUE, the log density is returned.
lower.tail	a logical, if TRUE, the default, then probabilities are P[X <= x], otherwise, P[X > x].
method	a character string denoting what should be displayed. Either the density and "dist" or random variates "rvs".
n	the number of observations.
p	a numeric vector of probabilities. [hillPlot] - probability required when option quantile is chosen.
q	a numeric vector of quantiles.
x	a numeric vector of quantiles.
xi, mu, beta	xi is the shape parameter, mu the location parameter, and beta is the scale parameter. The default values are xi=1, mu=0, and beta=1. Note, if xi=0 the distribution is of type Gumbel.

Value

d* returns the density,
p* returns the probability,
q* returns the quantiles, and
r* generates random variates.

All values are numeric vectors.

Author(s)

Alec Stephenson for R's evd and evir package, and
Diethelm Wuertz for this R-port.

References

Coles S. (2001); *Introduction to Statistical Modelling of Extreme Values*, Springer.

Embrechts, P., Klueppelberg, C., Mikosch, T. (1997); *Modelling Extremal Events*, Springer.

Examples

```
## rgev -
   # Create and plot 1000 Weibull distributed rdv:
   r = rgev(n = 1000, xi = -1)
   plot(r, type = "l", col = "steelblue", main = "Weibull Series")
   grid()

## dgev -
   # Plot empirical density and compare with true density:
   hist(r[abs(r)<10], nclass = 25, freq = FALSE, xlab = "r",
     xlim = c(-5,5), ylim = c(0,1.1), main = "Density")
   box()
   x = seq(-5, 5, by = 0.01)
   lines(x, dgev(x, xi = -1), col = "steelblue")

## pgev -
   # Plot df and compare with true df:
   plot(sort(r), (1:length(r)/length(r)),
     xlim = c(-3, 6), ylim = c(0, 1.1),
     cex = 0.5, ylab = "p", xlab = "q", main = "Probability")
   grid()
   q = seq(-5, 5, by = 0.1)
   lines(q, pgev(q, xi = -1), col = "steelblue")

## qgev -
   # Compute quantiles, a test:
   qgev(pgev(seq(-5, 5, 0.25), xi = -1), xi = -1)

## gevMoments:
   # Returns true mean and variance:
   gevMoments(xi = 0, mu = 0, beta = 1)

## Slider:
   # gevSlider(method = "dist")
   # gevSlider(method = "rvs")
```

`GevMdaEstimation` *Generalized Extreme Value Modelling*

Description

A collection and description functions to estimate the parameters of the GEV distribution. To model the GEV three types of approaches for parameter estimation are provided: Maximum likelihood

estimation, probability weighted moment method, and estimation by the MDA approach. MDA includes functions for the Pickands, Einmal-Decker-deHaan, and Hill estimators together with several plot variants.

Maximum Domain of Attraction estimators:

`hillPlot`	shape parameter and Hill estimate of the tail index,
`shaparmPlot`	variation of shape parameter with tail depth.

Usage

```
hillPlot(x, start = 15, ci = 0.95,
    doplot = TRUE, plottype = c("alpha", "xi"), labels = TRUE, ...)
shaparmPlot(x, p = 0.01*(1:10), xiRange = NULL, alphaRange = NULL,
    doplot = TRUE, plottype = c("both", "upper"))

shaparmPickands(x, p = 0.05, xiRange = NULL,
    doplot = TRUE, plottype = c("both", "upper"), labels = TRUE, ...)
shaparmHill(x, p = 0.05, xiRange = NULL,
    doplot = TRUE, plottype = c("both", "upper"), labels = TRUE, ...)
shaparmDEHaan(x, p = 0.05, xiRange = NULL,
    doplot = TRUE, plottype = c("both", "upper"), labels = TRUE, ...)
```

Arguments

`alphaRange, xiRange`
: [saparmPlot] -
plotting ranges for `alpha` and `xi`. By default the values are automatically selected.

`ci`
: [hillPlot] -
probability for asymptotic confidence band; for no confidence band set `ci` to zero.

`doplot`
: a logical. Should the results be plotted?
[shaparmPlot] -
a vector of logicals of the same lengths as tails defining for which tail depths plots should be created, by default plots will be generated for a tail depth of 5 percent. By default `c(FALSE, FALSE, FALSE, FALSE, TRUE, FALSE, FALSE, FALSE, FALSE, FALSE)`.

`labels`
: [hillPlot] -
whether or not axes should be labelled.

`plottype`
: [hillPlot] -
whether `alpha`, `xi` (1/alpha) or `quantile` (a quantile estimate) should be plotted.

`p`
: [qgev] -
a numeric vector of probabilities. [hillPlot] -
probability required when option `quantile` is chosen.

`start` [hillPlot] -
lowest number of order statistics at which to plot a point.

`x` [dgev][devd] -
a numeric vector of quantiles.
[gevFit] -
data vector. In the case of `method="mle"` the interpretation depends on the value of block: if no block size is specified then data are interpreted as block maxima; if block size is set, then data are interpreted as raw data and block maxima are calculated.
[hillPlot][shaparmPlot] -
the data from which to calculate the shape parameter, a numeric vector.
[print][plot] -
a fitted object of class `"gevFit"`.

`...` [gevFit] -
control parameters optionally passed to the optimization function. Parameters for the optimization function are passed to components of the `control` argument of `optim`.
[hillPlot] -
other graphics parameters.
[plot][summary] -
arguments passed to the plot function.

Details

Parameter Estimation:

`gevFit` and `gumbelFit` estimate the parameters either by the probability weighted moment method, `method="pwm"` or by maximum log likelihood estimation `method="mle"`. The summary method produces diagnostic plots for fitted GEV or Gumbel models.

Methods:

`print.gev`, `plot.gev` and `summary.gev` are print, plot, and summary methods for a fitted object of class `gev`. Concerning the summary method, the data are converted to unit exponentially distributed residuals under null hypothesis that GEV fits. Two diagnostics for iid exponential data are offered. The plot method provides two different residual plots for assessing the fitted GEV model. Two diagnostics for iid exponential data are offered.

Return Level Plot:

`gevrlevelPlot` calculates and plots the k-block return level and 95% confidence interval based on a GEV model for block maxima, where `k` is specified by the user. The k-block return level is that level exceeded once every `k` blocks, on average. The GEV likelihood is reparameterized in terms of the unknown return level and profile likelihood arguments are used to construct a confidence interval.

Hill Plot:

The function `hillPlot` investigates the shape parameter and plots the Hill estimate of the tail index of heavy-tailed data, or of an associated quantile estimate. This plot is usually calculated from the alpha perspective. For a generalized Pareto analysis of heavy-tailed data using the `gpdFit` function, it helps to plot the Hill estimates for `xi`.

Shape Parameter Plot:

The function `shaparmPlot` investigates the shape parameter and plots for the upper and lower tails the shape parameter as a function of the taildepth. Three approaches are considered, the *Pickands* estimator, the *Hill* estimator, and the *Decker-Einmal-deHaan* estimator.

Value

`gevSim`
returns a vector of data points from the simulated series.

`gevFit`
returns an object of class `gev` describing the fit.

`print.summary`
prints a report of the parameter fit.

`summary`
performs diagnostic analysis. The method provides two different residual plots for assessing the fitted GEV model.

`gevrlevelPlot`
returns a vector containing the lower 95% bound of the confidence interval, the estimated return level and the upper 95% bound.

`hillPlot`
displays a plot.

`shaparmPlot`
returns a list with one or two entries, depending on the selection of the input variable `both.tails`. The two entries `upper` and `lower` determine the position of the tail. Each of the two variables is again a list with entries `pickands`, `hill`, and `dehaan`. If one of the three methods will be discarded the printout will display zeroes.

Note

GEV Parameter Estimation:

If method `"mle"` is selected the parameter fitting in `gevFit` is passed to the internal function `gev.mle` or `gumbel.mle` depending on the value of `gumbel`, `FALSE` or `TRUE`. On the other hand, if method `"pwm"` is selected the parameter fitting in `gevFit` is passed to the internal function `gev.pwm` or `gumbel.pwm` again depending on the value of `gumbel`, `FALSE` or `TRUE`.

Author(s)

Alec Stephenson for R's evd and evir package, and
Diethelm Wuertz for this R-port.

References

Coles S. (2001); *Introduction to Statistical Modelling of Extreme Values*, Springer.

Embrechts, P., Klueppelberg, C., Mikosch, T. (1997); *Modelling Extremal Events*, Springer.

Examples

```
## Load Data:
   library(timeSeries)
   x = as.timeSeries(data(danishClaims))
   colnames(x) <- "Danish"
   head(x)

## hillPlot -
   # Hill plot of heavy-tailed Danish fire insurance data
   par(mfrow = c(1, 1))
   hillPlot(x, plottype = "xi")
   grid()
```

GevModelling *Generalized Extreme Value Modelling*

Description

A collection and description functions to estimate the parameters of the GEV distribution. To model the GEV three types of approaches for parameter estimation are provided: Maximum likelihood estimation, probability weighted moment method, and estimation by the MDA approach. MDA includes functions for the Pickands, Einmal-Decker-deHaan, and Hill estimators together with several plot variants.

The GEV modelling functions are:

gevSim	generates data from the GEV distribution,
gumbelSim	generates data from the Gumbel distribution,
gevFit	fits data to the GEV distribution,
gumbelFit	fits data to the Gumbel distribution,
print	print method for a fitted GEV object,
plot	plot method for a fitted GEV object,
summary	summary method for a fitted GEV object,
gevrlevelPlot	k-block return level with confidence intervals.

Usage

```
gevSim(model = list(xi = -0.25, mu = 0, beta = 1), n = 1000, seed = NULL)
gumbelSim(model = list(mu = 0, beta = 1), n = 1000, seed = NULL)

gevFit(x, block = 1, type = c("mle", "pwm"), title = NULL, description = NULL, ...)
gumbelFit(x, block = 1, type = c("mle", "pwm"), title = NULL, description = NULL, ...)

## S4 method for signature 'fGEVFIT'
show(object)
## S3 method for class 'fGEVFIT'
plot(x, which = "ask", ...)
## S3 method for class 'fGEVFIT'
summary(object, doplot = TRUE, which = "all", ...)
```

Arguments

block	block size.
description	a character string which allows for a brief description.
doplot	a logical. Should the results be plotted? [shaparmPlot] - a vector of logicals of the same lengths as tails defining for which tail depths plots should be created, by default plots will be generated for a tail depth of 5 percent. By default c(FALSE, FALSE, FALSE, FALSE, TRUE, FALSE, FALSE, FALSE, FALSE, FALSE).
model	[gevSim][gumbelSim] - a list with components shape, location and scale giving the parameters of the GEV distribution. By default the shape parameter has the value -0.25, the location is zero and the scale is one. To fit random deviates from a Gumbel distribution set shape=0.
n	[gevSim][gumbelSim] - number of generated data points, an integer value. [rgev] - the number of observations.
object	[summary][grlevelPlot] - a fitted object of class "gevFit".
seed	[gevSim] - an integer value to set the seed for the random number generator.
title	[gevFit] - a character string which allows for a project title.
type	a character string denoting the type of parameter estimation, either by maximum likelihood estimation "mle", the default value, or by the probability weighted moment method "pwm".
which	[plot][summary] - a vector of logicals, one for each plot, denoting which plot should be displayed. Alternatively if which="ask" the user will be interactively asked which of the plots should be displayed. By default which="all".

x	[dgev][devd] - a numeric vector of quantiles. [gevFit] - data vector. In the case of method="mle" the interpretation depends on the value of block: if no block size is specified then data are interpreted as block maxima; if block size is set, then data are interpreted as raw data and block maxima are calculated. [hillPlot][shaparmPlot] - the data from which to calculate the shape parameter, a numeric vector. [print][plot] - a fitted object of class "gevFit".
xi, mu, beta	[*gev] - xi is the shape parameter, mu the location parameter, and beta is the scale parameter. The default values are xi=1, mu=0, and beta=1. Note, if xi=0 the distribution is of type Gumbel.
...	[gevFit] - control parameters optionally passed to the optimization function. Parameters for the optimization function are passed to components of the control argument of optim. [hillPlot] - other graphics parameters. [plot][summary] - arguments passed to the plot function.

Details

Parameter Estimation:

gevFit and gumbelFit estimate the parameters either by the probability weighted moment method, method="pwm" or by maximum log likelihood estimation method="mle". The summary method produces diagnostic plots for fitted GEV or Gumbel models.

Methods:

print.gev, plot.gev and summary.gev are print, plot, and summary methods for a fitted object of class gev. Concerning the summary method, the data are converted to unit exponentially distributed residuals under null hypothesis that GEV fits. Two diagnostics for iid exponential data are offered. The plot method provides two different residual plots for assessing the fitted GEV model. Two diagnostics for iid exponential data are offered.

Return Level Plot:

gevrlevelPlot calculates and plots the k-block return level and 95% confidence interval based on a GEV model for block maxima, where k is specified by the user. The k-block return level is that level exceeded once every k blocks, on average. The GEV likelihood is reparameterized in terms of the unknown return level and profile likelihood arguments are used to construct a confidence interval.

Hill Plot:

The function `hillPlot` investigates the shape parameter and plots the Hill estimate of the tail index of heavy-tailed data, or of an associated quantile estimate. This plot is usually calculated from the alpha perspective. For a generalized Pareto analysis of heavy-tailed data using the `gpdFit` function, it helps to plot the Hill estimates for `xi`.

Shape Parameter Plot:

The function `shaparmPlot` investigates the shape parameter and plots for the upper and lower tails the shape parameter as a function of the taildepth. Three approaches are considered, the *Pickands* estimator, the *Hill* estimator, and the *Decker-Einmal-deHaan* estimator.

Value

`gevSim`
returns a vector of data points from the simulated series.

`gevFit`
returns an object of class `gev` describing the fit.

`print.summary`
prints a report of the parameter fit.

`summary`
performs diagnostic analysis. The method provides two different residual plots for assessing the fitted GEV model.

`gevrlevelPlot`
returns a vector containing the lower 95% bound of the confidence interval, the estimated return level and the upper 95% bound.

`hillPlot`
displays a plot.

`shaparmPlot`
returns a list with one or two entries, depending on the selection of the input variable `both.tails`. The two entries `upper` and `lower` determine the position of the tail. Each of the two variables is again a list with entries `pickands`, `hill`, and `dehaan`. If one of the three methods will be discarded the printout will display zeroes.

Note

GEV Parameter Estimation:

If method `"mle"` is selected the parameter fitting in `gevFit` is passed to the internal function `gev.mle` or `gumbel.mle` depending on the value of `gumbel`, `FALSE` or `TRUE`. On the other hand, if

method "pwm" is selected the parameter fitting in gevFit is passed to the internal function gev.pwm or gumbel.pwm again depending on the value of gumbel, FALSE or TRUE.

Author(s)

Alec Stephenson for R's evd and evir package, and
Diethelm Wuertz for this R-port.

References

Coles S. (2001); *Introduction to Statistical Modelling of Extreme Values*, Springer.

Embrechts, P., Klueppelberg, C., Mikosch, T. (1997); *Modelling Extremal Events*, Springer.

Examples

```
## gevSim -
   # Simulate GEV Data, use default length n=1000
   x = gevSim(model = list(xi = 0.25, mu = 0 , beta = 1), n = 1000)
   head(x)

## gumbelSim -
   # Simulate GEV Data, use default length n=1000
   x = gumbelSim(model = list(xi = 0.25, mu = 0 , beta = 1))

## gevFit -
   # Fit GEV Data by Probability Weighted Moments:
   fit = gevFit(x, type = "pwm")
   print(fit)

## summary -
   # Summarize Results:
   par(mfcol = c(2, 2))
   summary(fit)
```

GevRisk *Generalized Extreme Value Modelling*

Description

A collection and description functions to estimate the parameters of the GEV distribution. To model the GEV three types of approaches for parameter estimation are provided: Maximum likelihood estimation, probability weighted moment method, and estimation by the MDA approach. MDA includes functions for the Pickands, Einmal-Decker-deHaan, and Hill estimators together with several plot variants.

The GEV modelling functions are:

gevrlevelPlot	k-block return level with confidence intervals.

Usage

```
gevrlevelPlot(object, kBlocks = 20,  ci = c(0.90, 0.95, 0.99),
    plottype = c("plot", "add"), labels = TRUE,...)
```

Arguments

add	[gevrlevelPlot] - whether the return level should be added graphically to a time series plot; if FALSE a graph of the profile likelihood curve showing the return level and its confidence interval is produced.
ci	[hillPlot] - probability for asymptotic confidence band; for no confidence band set ci to zero.
kBlocks	[gevrlevelPlot] - specifies the particular return level to be estimated; default set arbitrarily to 20.
labels	[hillPlot] - whether or not axes should be labelled.
object	[summary][grlevelPlot] - a fitted object of class "gevFit".
plottype	[hillPlot] - whether alpha, xi (1/alpha) or quantile (a quantile estimate) should be plotted.
...	arguments passed to the plot function.

Details

Parameter Estimation:

gevFit and gumbelFit estimate the parameters either by the probability weighted moment method, method="pwm" or by maximum log likelihood estimation method="mle". The summary method produces diagnostic plots for fitted GEV or Gumbel models.

Methods:

print.gev, plot.gev and summary.gev are print, plot, and summary methods for a fitted object of class gev. Concerning the summary method, the data are converted to unit exponentially distributed residuals under null hypothesis that GEV fits. Two diagnostics for iid exponential data are offered. The plot method provides two different residual plots for assessing the fitted GEV model. Two diagnostics for iid exponential data are offered.

Return Level Plot:

gevrlevelPlot calculates and plots the k-block return level and 95% confidence interval based on a GEV model for block maxima, where k is specified by the user. The k-block return level is that level exceeded once every k blocks, on average. The GEV likelihood is reparameterized in terms of the unknown return level and profile likelihood arguments are used to construct a confidence

interval.

Hill Plot:

The function `hillPlot` investigates the shape parameter and plots the Hill estimate of the tail index of heavy-tailed data, or of an associated quantile estimate. This plot is usually calculated from the alpha perspective. For a generalized Pareto analysis of heavy-tailed data using the `gpdFit` function, it helps to plot the Hill estimates for `xi`.

Shape Parameter Plot:

The function `shaparmPlot` investigates the shape parameter and plots for the upper and lower tails the shape parameter as a function of the taildepth. Three approaches are considered, the *Pickands* estimator, the *Hill* estimator, and the *Decker-Einmal-deHaan* estimator.

Value

`gevSim`
returns a vector of data points from the simulated series.

`gevFit`
returns an object of class `gev` describing the fit.

`print.summary`
prints a report of the parameter fit.

`summary`
performs diagnostic analysis. The method provides two different residual plots for assessing the fitted GEV model.

`gevrlevelPlot`
returns a vector containing the lower 95% bound of the confidence interval, the estimated return level and the upper 95% bound.

`hillPlot`
displays a plot.

`shaparmPlot`
returns a list with one or two entries, depending on the selection of the input variable `both.tails`. The two entries `upper` and `lower` determine the position of the tail. Each of the two variables is again a list with entries `pickands`, `hill`, and `dehaan`. If one of the three methods will be discarded the printout will display zeroes.

Note

GEV Parameter Estimation:

If method "mle" is selected the parameter fitting in gevFit is passed to the internal function gev.mle or gumbel.mle depending on the value of gumbel, FALSE or TRUE. On the other hand, if method "pwm" is selected the parameter fitting in gevFit is passed to the internal function gev.pwm or gumbel.pwm again depending on the value of gumbel, FALSE or TRUE.

Author(s)

Alec Stephenson for R's evd and evir package, and
Diethelm Wuertz for this R-port.

References

Coles S. (2001); *Introduction to Statistical Modelling of Extreme Values*, Springer.

Embrechts, P., Klueppelberg, C., Mikosch, T. (1997); *Modelling Extremal Events*, Springer.

Examples

```
## Load Data:
   # BMW Stock Data - negative returns
   library(timeSeries)
   x = -as.timeSeries(data(bmwRet))
   colnames(x)<-"BMW"
   head(x)

## gevFit -
   # Fit GEV to monthly Block Maxima:
   fit = gevFit(x, block = "month")
   print(fit)

## gevrlevelPlot -
   # Return Level Plot:
   gevrlevelPlot(fit)
```

GpdDistribution *Generalized Pareto Distribution*

Description

A collection and description of functions to compute the generalized Pareto distribution. The functions compute density, distribution function, quantile function and generate random deviates for the GPD. In addition functions to compute the true moments and to display the distribution and random variates changing parameters interactively are available.

The GPD distribution functions are:

dgpd	Density of the GPD Distribution,
pgpd	Probability function of the GPD Distribution,

qgpd	Quantile function of the GPD Distribution,
rgpd	random variates from the GPD distribution,
gpdMoments	computes true mean and variance,
gpdSlider	displays density or rvs from a GPD.

Usage

```
dgpd(x, xi = 1, mu = 0, beta = 1, log = FALSE)
pgpd(q, xi = 1, mu = 0, beta = 1, lower.tail = TRUE)
qgpd(p, xi = 1, mu = 0, beta = 1, lower.tail = TRUE)
rgpd(n, xi = 1, mu = 0, beta = 1)

gpdMoments(xi = 1, mu = 0, beta = 1)
gpdSlider(method = c("dist", "rvs"))
```

Arguments

log	a logical, if TRUE, the log density is returned.
lower.tail	a logical, if TRUE, the default, then probabilities are P[X <= x], otherwise, P[X > x].
method	[gpdSlider] - a character string denoting what should be displayed. Either the density and "dist" or random variates "rvs".
n	[rgpd][gpdSim\ - the number of observations to be generated.
p	a vector of probability levels, the desired probability for the quantile estimate (e.g. 0.99 for the 99th percentile).
q	[pgpd] - a numeric vector of quantiles.
x	[dgpd] - a numeric vector of quantiles.
xi, mu, beta	xi is the shape parameter, mu the location parameter, and beta is the scale parameter.

Value

All values are numeric vectors:
d* returns the density,
p* returns the probability,
q* returns the quantiles, and
r* generates random deviates.

Author(s)

Alec Stephenson for the functions from R's evd package,
Alec Stephenson for the functions from R's evir package,
Alexander McNeil for the EVIS functions underlying the evir package,
Diethelm Wuertz for this R-port.

References

Embrechts, P., Klueppelberg, C., Mikosch, T. (1997); *Modelling Extremal Events*, Springer.

Examples

```
## rgpd  -
   par(mfrow = c(2, 2), cex = 0.7)
   r = rgpd(n = 1000, xi = 1/4)
   plot(r, type = "l", col = "steelblue", main = "GPD Series")
   grid()

## dgpd -
   # Plot empirical density and compare with true density:
   # Omit values greater than 500 from plot
   hist(r, n = 50, probability = TRUE, xlab = "r",
     col = "steelblue", border = "white",
     xlim = c(-1, 5), ylim = c(0, 1.1), main = "Density")
   box()
   x = seq(-5, 5, by = 0.01)
   lines(x, dgpd(x, xi = 1/4), col = "orange")

## pgpd -
   # Plot df and compare with true df:
   plot(sort(r), (1:length(r)/length(r)),
     xlim = c(-3, 6), ylim = c(0, 1.1), pch = 19,
     cex = 0.5, ylab = "p", xlab = "q", main = "Probability")
   grid()
   q = seq(-5, 5, by = 0.1)
   lines(q, pgpd(q, xi = 1/4), col = "steelblue")

## qgpd -
   # Compute quantiles, a test:
   qgpd(pgpd(seq(-1, 5, 0.25), xi = 1/4 ), xi = 1/4)
```

`GpdModelling` *GPD Distributions for Extreme Value Theory*

Description

A collection and description to functions to fit and to simulate processes that are generated from the generalized Pareto distribution. Two approaches for parameter estimation are provided: Maximum likelihood estimation and the probability weighted moment method.

The GPD modelling functions are:

`gpdSim`	generates data from the GPD,
`gpdFit`	fits empirical or simulated data to the distribution,
`print`	print method for a fitted GPD object of class ...,
`plot`	plot method for a fitted GPD object,
`summary`	summary method for a fitted GPD object.

Usage

```
gpdSim(model = list(xi = 0.25, mu = 0, beta = 1), n = 1000,
    seed = NULL)
gpdFit(x, u = quantile(x, 0.95), type = c("mle", "pwm"), information =
    c("observed", "expected"), title = NULL, description = NULL, ...)

## S4 method for signature 'fGPDFIT'
show(object)
## S3 method for class 'fGPDFIT'
plot(x, which = "ask", ...)
## S3 method for class 'fGPDFIT'
summary(object, doplot = TRUE, which = "all", ...)
```

Arguments

description	a character string which allows for a brief description.
doplot	a logical. Should the results be plotted?
information	whether standard errors should be calculated with "observed" or "expected" information. This only applies to the maximum likelihood method; for the probability-weighted moments method "expected" information is used if possible.
model	[gpdSim] - a list with components shape, location and scale giving the parameters of the GPD distribution. By default the shape parameter has the value 0.25, the location is zero and the scale is one.
n	[rgpd][gpdSim\ - the number of observations to be generated.
object	[summary] - a fitted object of class "gpdFit".
seed	[gpdSim] - an integer value to set the seed for the random number generator.
title	a character string which allows for a project title.
type	a character string selecting the desired estimation method, either "mle" for the maximum likelihood method or "pwm" for the probability weighted moment method. By default, the first will be selected. Note, the function gpd uses "ml".
u	the threshold value.
which	if which is set to "ask" the function will interactively ask which plot should be displayed. By default this value is set to FALSE and then those plots will be displayed for which the elements in the logical vector which ar set to TRUE; by default all four elements are set to "all".
x	[dgpd] - a numeric vector of quantiles. [gpdFit] - the data vector. Note, there are two different names for the first argument x and data depending which function name is used, either gpdFit or the EVIS

	synonym gpd. [print][plot] - a fitted object of class "gpdFit".
xi, mu, beta	xi is the shape parameter, mu the location parameter, and beta is the scale parameter.
...	control parameters and plot parameters optionally passed to the optimization and/or plot function. Parameters for the optimization function are passed to components of the control argument of optim.

Details

Generalized Pareto Distribution:

Compute density, distribution function, quantile function and generates random variates for the Generalized Pareto Distribution.

Simulation:

gpdSim simulates data from a Generalized Pareto distribution.

Parameter Estimation:

gpdFit fits the model parameters either by the probability weighted moment method or the maxim log likelihood method. The function returns an object of class "gpd" representing the fit of a generalized Pareto model to excesses over a high threshold. The fitting functions use the probability weighted moment method, if method method="pwm" was selected, and the the general purpose optimization function optim when the maximum likelihood estimation, method="mle" or method="ml" is chosen.

Methods:

print.gpd, plot.gpd and summary.gpd are print, plot, and summary methods for a fitted object of class gpdFit. The plot method provides four different plots for assessing fitted GPD model.

gpd* Functions:

gpdqPlot calculates quantile estimates and confidence intervals for high quantiles above the threshold in a GPD analysis, and adds a graphical representation to an existing plot. The GPD approximation in the tail is used to estimate quantile. The "wald" method uses the observed Fisher information matrix to calculate confidence interval. The "likelihood" method reparametrizes the likelihood in terms of the unknown quantile and uses profile likelihood arguments to construct a confidence interval.

gpdquantPlot creates a plot showing how the estimate of a high quantile in the tail of a dataset based on the GPD approximation varies with threshold or number of extremes. For every model gpdFit is called. Evaluation may be slow. Confidence intervals by the Wald method may be fastest.

gpdriskmeasures makes a rapid calculation of point estimates of prescribed quantiles and expected shortfalls using the output of the function gpdFit. This function simply calculates point estimates and (at present) makes no attempt to calculate confidence intervals for the risk measures. If confidence levels are required use gpdqPlot and gpdsfallPlot which interact with graphs of the tail of a loss distribution and are much slower.

gpdsfallPlot calculates expected shortfall estimates, in other words tail conditional expectation and confidence intervals for high quantiles above the threshold in a GPD analysis. A graphical representation to an existing plot is added. Expected shortfall is the expected size of the loss, given that a particular quantile of the loss distribution is exceeded. The GPD approximation in the tail is used to estimate expected shortfall. The likelihood is reparametrized in terms of the unknown expected shortfall and profile likelihood arguments are used to construct a confidence interval.

gpdshapePlot creates a plot showing how the estimate of shape varies with threshold or number of extremes. For every model gpdFit is called. Evaluation may be slow.

gpdtailPlot produces a plot of the tail of the underlying distribution of the data.

Value

gpdSim
returns a vector of datapoints from the simulated series.

gpdFit
returns an object of class "gpd" describing the fit including parameter estimates and standard errors.

gpdQuantPlot
returns invisible a table of results.

gpdShapePlot
returns invisible a table of results.

gpdTailPlot
returns invisible a list object containing details of the plot is returned invisibly. This object should be used as the first argument of gpdqPlot or gpdsfallPlot to add quantile estimates or expected shortfall estimates to the plot.

Author(s)

Alec Stephenson for the functions from R's evd package,
Alec Stephenson for the functions from R's evir package,
Alexander McNeil for the EVIS functions underlying the evir package,
Diethelm Wuertz for this R-port.

References

Embrechts, P., Klueppelberg, C., Mikosch, T. (1997); *Modelling Extremal Events*, Springer.

Hosking J.R.M., Wallis J.R., (1987); *Parameter and quantile estimation for the generalized Pareto distribution*, Technometrics 29, 339–349.

Examples

```
## gpdSim  -
   x = gpdSim(model = list(xi = 0.25, mu = 0, beta = 1), n = 1000)
## gpdFit -
   par(mfrow = c(2, 2), cex = 0.7)
   fit = gpdFit(x, u = min(x), type = "pwm")
   print(fit)
   summary(fit)
```

---

gpdRisk *GPD Distributions for Extreme Value Theory*

Description

A collection and description to functions to compute tail risk under the GPD approach.

The GPD modelling functions are:

gpdQPlot	estimation of high quantiles,
gpdQuantPlot	variation of high quantiles with threshold,
gpdRiskMeasures	prescribed quantiles and expected shortfalls,
gpdSfallPlot	expected shortfall with confidence intervals,
gpdShapePlot	variation of shape with threshold,
gpdTailPlot	plot of the tail,
tailPlot	,
tailSlider	,
tailRisk	.

Usage

```
gpdQPlot(x, p = 0.99, ci = 0.95, type = c("likelihood", "wald"),
    like.num = 50)
gpdQuantPlot(x, p = 0.99, ci = 0.95, models = 30, start = 15, end = 500,
    doplot = TRUE, plottype = c("normal", "reverse"), labels = TRUE,
    ...)
gpdSfallPlot(x, p = 0.99, ci = 0.95, like.num = 50)
gpdShapePlot(x, ci = 0.95, models = 30, start = 15, end = 500,
    doplot = TRUE, plottype = c("normal", "reverse"), labels = TRUE,
    ...)
gpdTailPlot(object, plottype = c("xy", "x", "y", ""), doplot = TRUE,
    extend = 1.5, labels = TRUE, ...)

gpdRiskMeasures(object, prob = c(0.99, 0.995, 0.999, 0.9995, 0.9999))

tailPlot(object, p = 0.99, ci = 0.95, nLLH = 25, extend = 1.5, grid =
    TRUE, labels = TRUE, ...)
```

```
tailSlider(x)
tailRisk(object, prob = c(0.99, 0.995, 0.999, 0.9995, 0.9999), ...)
```

Arguments

ci	the probability for asymptotic confidence band; for no confidence band set to zero.
doplot	a logical. Should the results be plotted?
extend	optional argument for plots 1 and 2 expressing how far x-axis should extend as a multiple of the largest data value. This argument must take values greater than 1 and is useful for showing estimated quantiles beyond data.
grid	...
labels	optional argument for plots 1 and 2 specifying whether or not axes should be labelled.
like.num	the number of times to evaluate profile likelihood.
models	the number of consecutive gpd models to be fitted.
nLLH	...
object	[summary] - a fitted object of class "gpdFit".
p	a vector of probability levels, the desired probability for the quantile estimate (e.g. 0.99 for the 99th percentile).
reverse	should plot be by increasing threshold (TRUE) or number of extremes (FALSE).
prob	a numeric value.
plottype	a character string.
start, end	the lowest and maximum number of exceedances to be considered.
type	a character string selecting the desired estimation method, either "mle" for the maximum likelihood method or "pwm" for the probability weighted moment method. By default, the first will be selected. Note, the function gpd uses "ml".
x	[dgpd] - a numeric vector of quantiles. [gpdFit] - the data vector. Note, there are two different names for the first argument x and data depending which function name is used, either gpdFit or the EVIS synonym gpd. [print][plot] - a fitted object of class "gpdFit".
...	control parameters and plot parameters optionally passed to the optimization and/or plot function. Parameters for the optimization function are passed to components of the control argument of optim.

Details

Generalized Pareto Distribution:

Compute density, distribution function, quantile function and generates random variates for the Generalized Pareto Distribution.

Simulation:

`gpdSim` simulates data from a Generalized Pareto distribution.

Parameter Estimation:

`gpdFit` fits the model parameters either by the probability weighted moment method or the maxim log likelihood method. The function returns an object of class `"gpd"` representing the fit of a generalized Pareto model to excesses over a high threshold. The fitting functions use the probability weighted moment method, if method `method="pwm"` was selected, and the the general purpose optimization function `optim` when the maximum likelihood estimation, `method="mle"` or `method="ml"` is chosen.

Methods:

`print.gpd`, `plot.gpd` and `summary.gpd` are print, plot, and summary methods for a fitted object of class `gpdFit`. The plot method provides four different plots for assessing fitted GPD model.

gpd* Functions:

`gpdqPlot` calculates quantile estimates and confidence intervals for high quantiles above the threshold in a GPD analysis, and adds a graphical representation to an existing plot. The GPD approximation in the tail is used to estimate quantile. The `"wald"` method uses the observed Fisher information matrix to calculate confidence interval. The `"likelihood"` method reparametrizes the likelihood in terms of the unknown quantile and uses profile likelihood arguments to construct a confidence interval.

`gpdquantPlot` creates a plot showing how the estimate of a high quantile in the tail of a dataset based on the GPD approximation varies with threshold or number of extremes. For every model `gpdFit` is called. Evaluation may be slow. Confidence intervals by the Wald method may be fastest.

`gpdriskmeasures` makes a rapid calculation of point estimates of prescribed quantiles and expected shortfalls using the output of the function `gpdFit`. This function simply calculates point estimates and (at present) makes no attempt to calculate confidence intervals for the risk measures. If confidence levels are required use `gpdqPlot` and `gpdsfallPlot` which interact with graphs of the tail of a loss distribution and are much slower.

`gpdsfallPlot` calculates expected shortfall estimates, in other words tail conditional expectation and confidence intervals for high quantiles above the threshold in a GPD analysis. A graphicalx representation to an existing plot is added. Expected shortfall is the expected size of the loss, given that a particular quantile of the loss distribution is exceeded. The GPD approximation in the tail

is used to estimate expected shortfall. The likelihood is reparametrized in terms of the unknown expected shortfall and profile likelihood arguments are used to construct a confidence interval.

gpdshapePlot creates a plot showing how the estimate of shape varies with threshold or number of extremes. For every model gpdFit is called. Evaluation may be slow.

gpdtailPlot produces a plot of the tail of the underlying distribution of the data.

Value

gpdSim
returns a vector of datapoints from the simulated series.

gpdFit
returns an object of class "gpd" describing the fit including parameter estimates and standard errors.

gpdQuantPlot
returns invisible a table of results.

gpdShapePlot
returns invisible a table of results.

gpdTailPlot
returns invisible a list object containing details of the plot is returned invisibly. This object should be used as the first argument of gpdqPlot or gpdsfallPlot to add quantile estimates or expected shortfall estimates to the plot.

Author(s)

Alec Stephenson for the functions from R's evd package,
Alec Stephenson for the functions from R's evir package,
Alexander McNeil for the EVIS functions underlying the evir package,
Diethelm Wuertz for this R-port.

References

Embrechts, P., Klueppelberg, C., Mikosch, T. (1997); *Modelling Extremal Events*, Springer.

Hosking J.R.M., Wallis J.R., (1987); *Parameter and quantile estimation for the generalized Pareto distribution*, Technometrics 29, 339–349.

Examples

```
## Load Data:
   library(timeSeries)
   danish = as.timeSeries(data(danishClaims))

## Tail Plot:
   x = as.timeSeries(data(danishClaims))
   fit = gpdFit(x, u = 10)
   tailPlot(fit)

## Try Tail Slider:
```

```
   # tailSlider(x)

## Tail Risk:
   tailRisk(fit)
```

`TimeSeriesData` *Time Series Data Sets*

Description

Data sets used in the examples of the fExtremes packages.

Usage

```
bmwRet
danishClaims
```

Format

`bmwRet`. A data frame with 6146 observations on 2 variables. The first column contains dates (Tuesday 2nd January 1973 until Tuesday 23rd July 1996) and the second column contains the respective value of daily log returns on the BMW share price made on each of those dates. These data are an irregular time series because there is no trading at weekends.

`danishClaims`. A data frame with 2167 observations on 2 variables. The first column contains dates and the second column contains the respective value of a fire insurance claim in Denmark made on each of those dates. These data are an irregular time series.

Examples

```
head(bmwRet)
head(danishClaims)
```

`ValueAtRisk` *Value-at-Risk*

Description

A collection and description of functions to compute Value-at-Risk and conditional Value-at-Risk

The functions are:

`VaR`	Computes Value-at-Risk,
`CVaR`	Computes conditional Value-at-Risk.

Usage

```
VaR(x, alpha = 0.05, type = "sample", tail = c("lower", "upper"))
CVaR(x, alpha = 0.05, type = "sample", tail = c("lower", "upper"))
```

Arguments

x	an uni- or multivariate timeSeries object
alpha	a numeric value, the confidence interval.
type	a character string, the type to calculate the value-at-risk.
tail	a character string denoting which tail will be considered, either "lower" or "upper". If tail="lower", then alpha will be converted to alpha=1-alpha.

Value

VaR
CVaR

returns a numeric vector or value with the (conditional) value-at-risk for each time series column.

Author(s)

Diethelm Wuertz for this R-port.

See Also

hillPlot, gevFit.

Index