Package 'extRemes'

May 8, 2026

Version 2.2-1

Date 2025-05-16

Title Extreme Value Analysis

Author Eric Gilleland [aut, cre] (ORCID:
<https://orcid.org/0000-0002-8058-7643>)

Maintainer Eric Gilleland <eric.gilleland@colostate.edu>

Depends R (>= 2.10.0), Lmoments, distillery (>= 1.0-4)

Imports graphics, stats, methods

Suggests fields

Description General functions for performing extreme value analysis. In particular, allows for inclusion of covariates into the parameters of the extreme-value distributions, as well as estimation through MLE, L-moments, generalized (penalized) MLE (GMLE), as well as Bayes. Inference methods include parametric normal approximation, profile-likelihood, Bayes, and bootstrapping. Some bivariate functionality and dependence checking (e.g., auto-tail dependence function plot, extremal index estimation) is also included. For a tutorial, see Gilleland and Katz (2016) <doi:10.18637/jss.v072.i08> and for bootstrapping, please see Gilleland (2020) <doi:10.1175/JTECH-D-20-0070.1>.

NeedsCompilation no

Repository CRAN

Date/Publication 2025-05-16 19:00:02 UTC

Contents

`extRemes-package` *extRemes – Weather and Climate Applications of Extreme Value Analysis (EVA)*

Description

extRemes is a suite of functions for carrying out analyses on the extreme values of a process of interest; be they block maxima over long blocks or excesses over a high threshold.

Versions >= 2.0-0 of this package differ considerably from the original package (versions <= 1.65), which was largely a package of graphical user interfaces (GUIs) mostly calling functions from the **ismev** package; a companion software package to Coles (2001). The former GUI windows of **extRemes** (<= 1.65) now run the command-line functions of **extRemes** (>= 2.0) and have been moved to a new package called **in2extRemes**.

For assistance using **extRemes** (>= 2.0-0), please see the tutorial at:

doi:10.18637/jss.v072.i08

Extreme Value Statistics:

Extreme value statistics are used primarily to quantify the stochastic behavior of a process at unusually large (or small) values. Particularly, such analyses usually require estimation of the probability of events that are more extreme than any previously observed. Many fields have begun to use extreme value theory and some have been using it for a very long time including meteorology, hydrology, finance and ocean wave modeling to name just a few. See Gilleland and Katz (2011) for a brief introduction to the capabilities of **extRemes**.

Example Datasets:

There are several example datasets included with this toolkit. In each case, it is possible to load these datasets into R using the `data` function. Each data set has its own help file, which can be accessed by `help([name of dataset])`. Data included with **extRemes** are:

Denmint – Denver daily minimum temperature.

Flood.dat – U.S. Flood damage (in terms of monetary loss) ('dat' file used as example of reading in common data using the extRemes dialog).

ftcanmax – Annual maximum precipitation amounts at one rain gauge in Fort Collins, Colorado.

HEAT – Summer maximum (and minimum) temperature at Phoenix Sky Harbor airport.

Ozone4H.dat – Ground-level ozone order statistics from 1997 from 513 monitoring stations in the eastern United States.

PORTw – Maximum and minimum temperature data (and some covariates) for Port Jervis, New York.

Rsum – Frequency of Hurricanes.

SEPTsp – Maximum and minimum temperature data (and some covariates) for Sept-Iles, Quebec.

damage – Hurricane monetary damage.

Denversp – Denver precipitation.

FCwx – data frame giving daily weather data for Fort Collins, Colorado, U.S.A. from 1900 to 1999.

Flood – R source version of the above mentioned 'Flood.dat' dataset.

Fort – Precipitation amounts at one rain gauge in Fort Collins, Colorado.

Peak – Salt River peak stream flow.

Potomac – Potomac River peak stream flow.

Tphap – Daily maximum and minimum temperatures at Phoenix Sky Harbor Airport.

Primary functions available in **extRemes** include:

`fevd`: Fitting extreme value distribution functions (EVDs: GEV, Gumbel, GP, Exponential, PP) to data (block maxima or threshold excesses).

`ci`: Method function for finding confidence intervals for EVD parameters and return levels.

`taildep`: Estimate chi and/or chibar; statistics that inform about tail dependence between two variables.

`atdf`: Auto-tail dependence function and plot. Helps to inform about possible dependence in the extremes of a process. Note that a process that is highly correlated may or may not be dependent in the extremes.

`decluster`: Decluster threshold exceedance in a data set to yield a new related process that is more closely independent in the extremes. Includes two methods for declustering both of which are based on runs declustering.

`extremalindex`: Estimate the extremal index, a measure of dependence in the extremes. Two methods are available, one based on runs declustering and the other is the intervals estiamte of Ferro and Segers (2003).

`devd, pevd, qevd, revd`: Functions for finding the density, cumulative probability distribution (cdf), quantiles and make random draws from EVDs.

`pextRemes, rextRemes, return.level`: Functions for finding the cdf, make random draws from, and find return levels for fitted EVDs.

To see how to cite **extRemes** in publications or elsewhere, use `citation("extRemes")`.

Acknowledgements

Funding for **extRemes** was originally provided by the Weather and Climate Impacts Assessment Science (WCIAS) Program at the National Center for Atmospheric Research (NCAR) in Boulder, Colorado. WCIAS was funded by the National Science Foundation (NSF). Curent funding is provided by the Regional Climate Uncertainty Program (RCUP), an NSF-supported program at NCAR. NCAR is operated by the nonprofit University Corporation for Atmospheric Research (UCAR) under the sponsorship of the NSF. Any opinions, findings, conclusions, or recommendations expressed in this publication/software package are those of the author(s) and do not necessarily reflect the views of the NSF.

References

Coles, S. (2001) *An introduction to statistical modeling of extreme values*, London, U.K.: Springer-Verlag, 208 pp.

Ferro, C. A. T. and Segers, J. (2003). Inference for clusters of extreme values. *Journal of the Royal Statistical Society B*, **65**, 545–556.

Gilleland, E. and Katz, R. W. (2011). New software to analyze how extremes change over time. *Eos*, 11 January, **92**, (2), 13–14, doi:10.18637/jss.v072.i08.

atdf *Auto-Tail Dependence Function*

Description

Computes (and by default plots) estimates of the auto-tail dependence function(s) (atdf) based on either chi (rho) or chibar (rhobar), or both.

Usage

```
atdf(x, u, lag.max = NULL, type = c("all", "rho", "rhobar"), plot = TRUE,
    na.action = na.fail, ...)

## S3 method for class 'atdf'
plot(x, type = NULL, ...)
```

Arguments

x	For atdf: a univariate time series object or a numeric vector. For the plot method function, a list object of class "atdf".
u	numeric between 0 and 1 (non-inclusive) determining the level F^(-1)(u) over which to compute the atdf. Typically, this should be close to 1, but low enough to incorporate enough data.
lag.max	The maximum lag for which to compute the atdf. Default is 10*log10(n), where n is the length of the data. Will be automatically limited to one less than the total number of observations in the series.
type	character string stating which type of atdf to calculate/plot (rho, rhobar or both). If NULL the plot method function will take the type to be whatever was passed to the call to atdf. If "all", then a 2 by 1 panel of two plots are graphed.
plot	logical, should the plot be made or not? If TRUE, output is returned invisibly. If FALSE, output is returned normally.
na.action	function to be called to handle missing values.
...	Further arguments to be passed to the plot method function or to plot. Note that if main, xlab or ylab are used with type "all", then the labels/title will be applied to both plots, which is probably not desirable.

Details

The tail dependence functions are those described in, e.g., Reiss and Thomas (2007) Eq (2.60) for "chi" and Eq (13.25) "chibar", and estimated by Eq (2.62) and Eq (13.28), resp. See also, Sibuya (1960) and Coles (2001) sec. 8.4, as well as other texts on EVT such as Beirlant et al. (2004) sec. 9.4.1 and 10.3.4 and de Haan and Ferreira (2006).

Specifically, for two series X and Y with associated df's F and G, chi, a function of u, is defined as

chi(u) = Pr[Y > G^(-1)(u) | X > F^(-1)(u)] = Pr[V > u | U > u],

where (U,V) = (F(X),G(Y))–i.e., the copula. Define chi = limit as u goes to 1 of chi(u).

The coefficient of tail dependence, chibar(u) was introduced by Coles et al. (1999), and is given by

chibar(u) = 2*log(Pr[U > u])/log(Pr[U > u, V > u]) - 1.

Define chibar = limit as u goes to 1 of chibar(u).

The auto-tail dependence function using chi(u) and/or chibar(u) employs X against itself at different lags.

The associated estimators for the auto-tail dependence functions employed by these functions are based on the above two coefficients of tail dependence, and are given by Reiss and Thomas (2007) Eq (2.65) and (13.28) for a lag h as

rho.hat(u, h) = sum(min(x_i, x_(i+h)) > sort(x)[floor(n*u)])/(n*(1-u)) [based on chi]

and

rhobar.hat(u, h) = 2*log(1 - u)/log(sum(min(x_i,x_(i+h)) > sort(x)[floor(n*u)])/(n - h)) - 1.

Some properties of the above dependence coefficients, chi(u), chi, and chibar(u) and chibar, are that 0 <= chi(u), chi <= 1, where if X and Y are stochastically independent, then chi(u) = 1 - u, and chibar = 0. If X = Y (perfectly dependent), then chi(u) = chi = 1. For chibar(u) and chibar, we have that -1 <= chibar(u), chibar <= 1. If U = V, then chibar = 1. If chi = 0, then chibar < 1 (tail independence with chibar determining the degree of dependence).

Value

A list object of class "atdf" is returned with components:

`call`	The function calling string.
`type`	character naming the type of atdf computed.
`series`	character string naming the series used.
`lag`	numeric vector giving the lags used.
`atdf`	numeric vector or if type is "all", two-column matrix giving the estimated auto-tail dependence function values.

The plot method functoin does not return anything.

Author(s)

Eric Gilleland

References

Beirlant, J., Goegebeur, Y., Teugels, J. and Segers, J. (2004) *Statistics of Extremes: Theory and Applications*. Chichester, West Sussex, England, UK: Wiley, ISBN 9780471976479, 522pp.

Coles, S. (2001) *An introduction to statistical modeling of extreme values*, London, U.K.: Springer-Verlag, 208 pp.

Coles, S., Heffernan, J. E., and Tawn, J. A. (1999) Dependence measures for extreme value analyses. *Extremes*, **2**, 339–365.

de Haan, L. and Ferreira, A. (2006) *Extreme Value Theory: An Introduction*. New York, NY, USA: Springer, 288pp.

Reiss, R.-D. and Thomas, M. (2007) *Statistical Analysis of Extreme Values: with applications to insurance, finance, hydrology and other fields*. Birkhauser, 530pp., 3rd edition.

Sibuya, M. (1960) Bivariate extreme statistics. *Ann. Inst. Math. Statist.*, **11**, 195–210.

See Also

acf, pacf, taildep, taildep.test

Examples

```
z <- arima.sim(n = 63, list(ar = c(0.8897, -0.4858), ma = c(-0.2279, 0.2488)),
               sd = sqrt(0.1796))
hold <- atdf(z, 0.8, plot=FALSE)
par(mfrow=c(2,2))
acf(z, xlab="")
pacf(z, xlab="")
plot(hold, type="chi")
plot(hold, type="chibar")

y <- cbind(z[2:63], z[1:62])
y <- apply(y, 1, max)
hold2 <- atdf(y, 0.8, plot=FALSE)
par(mfrow=c(2,2))
acf(y, xlab="")
pacf(y, xlab="")
plot(hold2, type="chi")
plot(hold2, type="chibar")

## Not run:
data(Fort)
atdf(Fort[,5], 0.9)

data(Tphap)
atdf(Tphap$MaxT, 0.8)

data(PORTw)
atdf(PORTw$TMX1, u=0.9)
atdf(PORTw$TMX1, u=0.8)

## End(Not run)
```

BayesFactor *Estimate Bayes Factor*

Description

Estimate Bayes factor between two models for two "fevd" objects.

Usage

```
BayesFactor(m1, m2, burn.in = 499, FUN = "postmode",
    method = c("laplace", "harmonic"), verbose = FALSE)
```

Arguments

m1, m2	objects of class "fevd" giving the two models to be compared.
burn.in	numeric how many of the first several iterations from the MCMC sample to throw away before estimating the Bayes factor.
FUN	function to be used to determine the estimated parameter values from the MCMC sample. With the exception of the default (posterior mode), the function should operate on a matrix and return a vector of length equal to the number of parameters. If "mean" is given, then colMeans is actually used.
method	Estimation method to be used.
verbose	logical, should progress information be printed to the screen (no longer necessary).

Details

Better options for estimating the Bayes factor from an MCMC sample are planned for the future. The current options are perhaps the two most common, but do suffer from major drawbacks. See Kass and Raftery (1995) for a review.

Value

A list object of class "htest" is returned with components:

statistic	The estimated Bayes factor.
method	character string naming which estimation method was used.
data.name	character vector naming the models being compared.

Author(s)

Eric Gilleland

References

Kass, R. E. and Raftery, A. E. (1995) Bayes factors. *J American Statistical Association*, **90** (430), 773–795.

See Also

fevd

Examples

```
data(PORTw)
fB <- fevd(TMX1, PORTw, method = "Bayesian", iter = 500)
fB2 <- fevd(TMX1, PORTw, location.fun = ~AOindex,
    method = "Bayesian", iter = 500)

BayesFactor(fB, fB2, burn.in = 100, method = "harmonic")
```

---

blockmaxxer *Find Block Maxima*

Description

Find the block maximum of a data set.

Usage

```
blockmaxxer(x, ...)

## S3 method for class 'data.frame'
blockmaxxer(x, ..., which = 1, blocks = NULL,
    blen = NULL, span = NULL)

## S3 method for class 'fevd'
blockmaxxer(x, ...)

## S3 method for class 'matrix'
blockmaxxer(x, ..., which = 1, blocks = NULL,
    blen = NULL, span = NULL)

## S3 method for class 'vector'
blockmaxxer(x, ..., blocks = NULL, blen = NULL,
    span = NULL)
```

Arguments

x	An object of class "fevd" where the fit is for a PP model, a numeric vector, matrix or data frame.
...	optional arguments to max.
which	number or name of a column indicating for which column to take the block maxima. Note, this does not take componentwise maxima (as in the multivariate setting). Instead, it takes the maxima for a single column and returns a vector, data frame or matrix of the block maxima for that column along with the entire row where that maxima occurred.
blocks	numeric (integer or factor) vector indicating the blocks over which to take the maxima. Must be non-NULL if blen and span are NULL.

blen	(optional) may be used instead of the blocks argument, and span must be non-NULL. This determines the length of the blocks to be created. Note, the last block may be smaller or larger than blen. Ignored if blocks is not NULL.
span	(optional) must be specified if blen is non-NULL and blocks is NULL. This is the number of blocks over which to take the maxima, and the returned value will be either a vector of length equal to span or a matrix or data frame with span rows.

Value

vector of length equal to the number of blocks (vector method) or a matrix or data frame with number of rows equal to the number of blocks (matrix and data frame methods).

The fevd method is for finding the block maxima of the data passed to a PP model fit and the blocks are determined by the npy and span components of the fitted object. If the fevd object is not a PP model, the function will error out. This is useful for utilizing the PP model in the GEV with approximate annual maxima. Any covariate values that occur contiguous with the maxima are returned as well.

The aggregate function is used with max in order to take the maxima from each block.

Author(s)

Eric Gilleland

See Also

fevd, max, aggregate

Examples

```
data(Fort)

bmFort <- blockmaxxer(Fort, blocks = Fort$year, which="Prec")

plot(Fort$year, Fort$Prec, xlab = "Year", ylab = "Precipitation (inches)",
    cex = 1.25, cex.lab = 1.25,
    col = "darkblue", bg = "lightblue", pch = 21)

points(bmFort$year, bmFort$Prec, col="darkred", cex=1.5)
```

CarcasonneHeat *European Climate Assessment and Dataset*

Description

Blended temperature multiplied by ten (deg Celsius) series of station STAID: 766 in Carcasonne, France.

Usage

```
data("CarcasonneHeat")
```

Format

The format is: int [1:4, 1:12054] 104888 19800101 96 0 104888 19800102 57 0 104888 19800103 ...

Details

European Climate Assessment and Dataset blended temperature (deg Celsius) series of station STAID: 766 in Carcasonne, France. Blended and updated with sources: 104888 907635. See Klein Tank et al. (2002) for more information.

This index was developed by Simone Russo at the European Commission, Joint Research Centre (JRC). Reports, articles, papers, scientific and non-scientific works of any form, including tables, maps, or any other kind of output, in printed or electronic form, based in whole or in part on the data supplied, must reference to Russo et al. (2014).

Author(s)

Simone Russo <simone.russo@jrc.ec.europa.eu>

Source

We acknowledge the data providers in the ECA&D project.

Klein Tank, A.M.G. and Coauthors, 2002. Daily dataset of 20th-century surface air temperature and precipitation series for the European Climate Assessment. Int. J. of Climatol., 22, 1441-1453.

Data and metadata available at https://www.ecad.eu:443/

References

Russo, S. and Coauthors, 2014. Magnitude of extreme heat waves in present climate and their projection in a warming world. *J. Geophys. Res.*, doi:10.1002/2014JD022098.

Examples

```
data(CarcasonneHeat)
str(CarcasonneHeat)

# see help file for hwmi for an example using these data.
```

ci.fevd *Confidence Intervals*

Description

Confidence intervals for parameters and return levels using fevd objects.

Usage

```
## S3 method for class 'fevd'
ci(x, alpha = 0.05, type = c("return.level", "parameter"),
    return.period = 100, which.par, R = 502, ...)

## S3 method for class 'fevd.bayesian'
ci(x, alpha = 0.05, type = c("return.level", "parameter"),
   return.period = 100, which.par = 1, FUN = "mean", burn.in = 499, tscale = FALSE,
    ...)

## S3 method for class 'fevd.lmoments'
ci(x, alpha = 0.05, type = c("return.level", "parameter"),
    return.period = 100, which.par, R = 502, tscale = FALSE,
    return.samples = FALSE, ...)

## S3 method for class 'fevd.mle'
ci(x, alpha = 0.05, type = c("return.level", "parameter"),
    return.period = 100, which.par, R = 502, method = c("normal",
        "boot", "proflik"), xrange = NULL, nint = 20, verbose = FALSE,
    tscale = FALSE, return.samples = FALSE, ...)
```

Arguments

x	list object returned by fevd.
alpha	numeric between 0 and 1 giving the desired significance level (i.e., the (1 - alpha) * 100 percent confidence level; so that the default alpha = 0.05 corresponds to a 95 percent confidence level).
type	character specifying if confidence intervals (CIs) are desired for return level(s) (default) or one or more parameter.
return.period	numeric vector giving the return period(s) for which it is desired to calculate the corresponding return levels.
...	optional arguments to the profliker function. For example, if it is desired to see the plot (recommended), use verbose = TRUE.
which.par	numeric giving the index (indices) for which parameter(s) to calculate CIs. Default is to do all of them.

FUN	character string naming a function to use to estimate the parameters from the MCMC sample. The function is applied to each column of the results component of the returned fevd object.
burn.in	The first burn.in values are thrown out before calculating anything from the MCMC sample.
R	the number of bootstrap iterations to do.
method	character naming which method for obtaining CIs should be used. Default ("normal") uses a normal approximation, and in the case of return levels (or transformed scale) applies the delta method using the parameter covariance matrix. Option "boot" employs a parametric bootstrap that simulates data from the fitted model, and then fits the EVD to each simulated data set to obtain a sample of parameters or return levels. Currently, only the percentile method of calculating the CIs from the sample is available. Finally, "proflik" uses function profliker to calculate the profile-likelihood function for the parameter(s) of interest, and tries to find the upcross level between this function and the appropriate chi-square critical value (see details).
tscale	For the GP df, the scale parameter is a function of the shape parameter and the threshold. When plotting the parameters, for example, against thresholds to find a good threshold for fitting the GP df, it is imperative to transform the scale parameter to one that is independent of the threshold. In particular, tscale = scale - shape * threshold.
xrange, nint	arguments to profliker function.
return.samples	logical; should the bootstrap samples be returned? If so, CIs will not be calculated and only the sample of parameters (return levels) are returned.
verbose	logical; should progress information be printed to the screen? For profile likelihood method (method = "proflik"), if TRUE, the profile-likelihood will also be plotted along with a horizontal line through the chi-square critical value.

Details

Confidence Intervals (ci):

ci: The ci method function will take output from fevd and calculate confidence intervals (or credible intervals in the case of Bayesian estimation) in an appropriate manner based on the estimation method. There is no need for the user to call ci.fevd, ci.fevd.lmoments, ci.fevd.bayesian or ci.fevd.mle; simply use ci and it will access the correct functions.

Currently, for L-moments, the only method available in this software is to apply a parameteric bootstrap, which is also available for the MLE/GMLE methods. A parametric bootstrap is performed via the following steps.

1. Simulate a sample of size n = lenght of the original data from the fitted model.

2. Fit the EVD to the simulated sample and store the resulting parameter estimates (and perhaps any combination of them, such as return levels).

3. Repeat steps 1 and 2 many times (to be precise, R times) to obtain a sample from the population df of the parameters (or combinations thereof).

4. From the sample resulting form the above steps, calculate confidence intervals. In the present code, the only option is to do this by taking the alpha/2 and 1 - alpha/2 quantiles of the sample

(i.e., the percentile method). However, if one uses `return.samples` = TRUE, then the sample is returned instead of confidence intervals allowing one to apply some other method if they so desire.

As far as guidance on how large `R` should be, it is a trial and error decision. Usually, one wants the smallest value (to make it as fast as possible) that still yields accurate results. Generally, this means doing it once with a relatively low number (say `R` = 100), and then doing it again with a higher number, say `R` = 250. If the results are very different, then do it again with an even higher number. Keep doing this until the results do not change drastically.

For MLE/GMLE, the normal approximation (perhaps using the delta method, e.g., for return levels) is used if `method` = "normal". If `method` = "boot", then parametric bootstrap CIs are found. Finally, if `method` = "profliker", then bounds based on the profile likelihood method are found (see below for more details).

For Bayesian estimation, the alpha/2 and 1 - alpha/2 percentiles of the resulting MCMC sample (after removing the first `burn.in` values) are used. If return levels are desired, then they are first calculated for each MCMC iteration, and the same procedure is applied. Note that the MCMC samples are availabel in the `fevd` output for this method, so any other procedure for finding CIs can be done by the savvy user.

Finding CIs based on the profile-likelihood method:

The profile likelihood method is often the best method for finding accurate CIs for the shape parameter and for return levels associated with long return periods (where their distribution functions are generally skewed so that, e.g., the normal approximation is not a good approximation). The profile likelihood for a parameter is obtained by maximizing the likelihood over the other parameters of the model for each of a range (`xrange`) of values. An approximation confidence region can be obtained using the deviance function D = 2 * (l(theta.hat) - l_p(theta)), where l(theta.hat) is the likelihood for the original model evaluated at their estimates and l_p(theta) is the likelihood of the parameter of interest (optimized over the remaining parameters), which approximately follows a chi-square df with degrees of freedom equal ot the number of parameters in the model less the one of interest. The confidence region is then given by

C_alpha = the set of theta_1 s.t. D <= q,

where q is the 1 - alpha quantile of the chi-square df with degrees of freedom equal to 1 and theta_1 is the parameter of interest. If we let m represent the maximum value of the profile likelihood (i.e., m = max(l_p(theta))), then consider a horizontal line through m - q. All values of theta_1 that yield a profile likelihood value above this horizontal line are within the confidence region, C_alpha (i.e., the range of these values represents the (1 - alpha) * 100 percent CI for the parameter of interest). For combinations of parameters, such as return levels, the same technique is applied by transforming the parameters in the likelihood to reflect the desired combination.

To use the profile-likelihood approach, it is necessary to choose an `xrange` argument that covers the entire confidence interval and beyond (at least a little), and the `nint` argument may be important here too (this argument gives the number of points to try in fitting a spline function to the profile likelihood, and smaller values curiously tend to be better, but not too small! Smaller values are also more efficient). Further, one should really look at the plot of the profile-likelihood to make sure that this is the case, and that resulting CIs are accurately estimated (perhaps using the `locator` function to be sure). Nevertheless, an attempt is made to find the limits automatically. To look at the plot along with the horizontal line, m - q, and vertical lines through the MLE (thin black dashed) and the CIs (thick dashed blue), use the `verbose` = TRUE argument in the call to `ci`. This is not an explicit argument, but available nonetheless (see examples below).

See any text on EVA/EVT for more details (e.g., Coles 2001; Beirlant et al 2004; de Haan and Ferreira 2006).

Value

Either a numeric vector of length 3 (if only one parameter/return level is used) or a matrix. In either case, they will have class "ci".

Author(s)

Eric Gilleland

References

Beirlant, J., Goegebeur, Y., Teugels, J. and Segers, J. (2004). *Statistics of Extremes: Theory and Applications*. Chichester, West Sussex, England, UK: Wiley, ISBN 9780471976479, 522pp.

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

de Haan, L. and Ferreira, A. (2006). *Extreme Value Theory: An Introduction*. New York, NY, USA: Springer, 288pp.

See Also

fevd, ci.rl.ns.fevd.bayesian, distillery::ci

Examples

```
data(Fort)

fit <- fevd(Prec, Fort, threshold = 2, type = "GP",
    units = "inches", verbose = TRUE)

ci(fit, type = "parameter")

## Not run:
ci(fit, type = "return.level", method = "proflik",
    xrange = c(3.5,7.75), verbose = TRUE)
# Can check using locator(2).

ci(fit, method = "boot")


## End(Not run)
```

`ci.rl.ns.fevd.bayesian` *Confidence/Credible Intervals for Effective Return Levels*

Description

Calculates credible intervals based on the upper and lower alpha/2 quantiles of the MCMC sample for effective return levels from a non-stationary EVD fit using Bayesian estimation, or find normal approximation confidence intervals if estimation method is MLE.

Usage

```
## S3 method for class 'rl.ns.fevd.bayesian'
ci(x, alpha = 0.05, return.period = 100, FUN = "mean",
                burn.in = 499, ..., qcov = NULL, qcov.base = NULL,
                verbose = FALSE)

## S3 method for class 'rl.ns.fevd.mle'
ci(x, alpha = 0.05, return.period = 100, method =
                c("normal"), verbose = FALSE, qcov = NULL, qcov.base =
                NULL, ...)
```

Arguments

`x`	An object of class "fevd".
`alpha`	Confidence level (numeric).
`return.period`	numeric giving the desired return period. Must have length one!
`FUN`	character string naming the function to use to calculate the estimated return levels from the posterior sample (default takes the posterior mean).
`burn.in`	The first `burn.in` iterations will be removed from the posterior sample before calculating anything.
`method`	Currently only "normal" method is implemented.
`verbose`	logical, should progress information be printed to the screen? Currently not used by the MLE method.
`...`	Not used.
`qcov, qcov.base`	Matrix giving specific covariate values. `qcov.base` is used if difference betwen effective return levels for two (or more) sets of covariates is desired, where it is rl(qcov) - rl(qcov.base). See `make.qcov` for more details. If not supplied, effective return levels are calculated for all of the original covariate values used for the fit. If `qcov.base` is not NULL but `qcov` is NULL, then `qcov` takes on the values of `qcov.base` and `qcov.base` is set to NULL, and a warning message is produced.

Details

Return levels are calculated for all coavariates supplied by qcov (and, if desired, qcov.base) for all values of the posterior sample (less the burn.in), or for all values of the original covariates used for the fit (if qcov and qcov.base are NULL). The estimates aree taken from the sample according to FUN and credible intervals are returned according to alpha.

Value

A three-column matrix is returned with the estimated effective return levels in the middle and lower and upper to the left and right.

Author(s)

Eric Gilleland

See Also

make.qcov, fevd, ci.fevd, return.level

Examples

```
data(Fort)
fit <- fevd(Prec, threshold = 2, data = Fort,
    location.fun = ~cos(2 * pi * day /365.25),
    type = "PP", verbose = TRUE)

v <- make.qcov(fit, vals=list(mu1 = c(cos(2 * pi * 1 /365.25),
    cos(2 * pi * 120 /365.25), cos(2 * pi * 360 /365.25))))

ci(fit, return.period = 100, qcov = v)

## Not run:
fit <- fevd(Prec, threshold = 2, data = Fort,
    location.fun = ~cos(2 * day /365.25),
    type = "PP", method = "Bayesian", verbose = TRUE)

ci(fit, return.period = 100, qcov = v)

## End(Not run)
```

---

damage *Hurricane Damage Data*

Description

Estimated economic damage (billions USD) caused by hurricanes.

Usage

```
data(damage)
```

Format

A data frame with 144 observations on the following 3 variables.

obs a numeric vector that simply gives the line numbers.

Year a numeric vector giving the years in which the specific hurricane occurred.

Dam a numeric vector giving the total estimated economic damage in billions of U.S. dollars.

Details

More information on these data can be found in Pielke and Landsea (1998) or Katz (2002).

References

Katz, R. W. (2002) Stochastic modeling of hurricane damage. *Journal of Applied Meteorology*, **41**, 754–762.

Pielke, R. A. Jr. and Landsea, C. W. (1998) Normalized hurricane damages in the United States: 1925-95. *Weather and Forecasting*, **13**, (3), 621–631.

Examples

```
data(damage)
plot( damage[,1], damage[,3], xlab="", ylab="Economic Damage", type="l", lwd=2)

# Fig. 3 of Katz (2002).
plot( damage[,"Year"], log( damage[,"Dam"]), xlab="Year", ylab="ln(Damage)", ylim=c(-10,5))

# Fig. 4 of Katz (2002).
qqnorm( log( damage[,"Dam"]), ylim=c(-10,5))
```

---

datagrabber.declustered

Get Original Data from an R Object

Description

Get the original data set used to obtain the resulting R object for which a method function exists.

Usage

```
## S3 method for class 'declustered'
datagrabber(x, ...)

## S3 method for class 'extremalindex'
datagrabber(x, ...)

## S3 method for class 'fevd'
datagrabber(x, response = TRUE,
    cov.data = TRUE, ...)
```

Arguments

x	An R object that has a method function for datagrabber.
response, cov.data	logical; should the response data be returned? Should the covariate data be returned?
...	optional arguments to get. This may eventually become deprecated as scoping gets mixed up, and is currently not actually used.

Details

Accesses the original data set from a fitted fevd object or from declustered data (objects of class "declustered") or from extremalindex.

Value

The original pertinent data in whatever form it takes.

Author(s)

Eric Gilleland

See Also

d[distillery::datagrabber, extremalindex, decluster, fevd, get

Examples

```
y <- rnorm(100, mean=40, sd=20)
y <- apply(cbind(y[1:99], y[2:100]), 1, max)
bl <- rep(1:3, each=33)

ydc <- decluster(y, quantile(y, probs=c(0.95)), r=1, blocks=bl)

yorig <- datagrabber(ydc)
all(y - yorig == 0)
```

decluster *Decluster Data Above a Threshold*

Description

Decluster data above a given threshold to try to make them independent.

Usage

```
decluster(x, threshold, ...)

## S3 method for class 'data.frame'
decluster(x, threshold, ..., which.cols, method = c("runs", "intervals"),
    clusterfun = "max")

## Default S3 method:
decluster(x, threshold, ..., method = c("runs", "intervals"),
    clusterfun = "max")

## S3 method for class 'intervals'
decluster(x, threshold, ..., clusterfun = "max", groups = NULL, replace.with,
    na.action = na.fail)

## S3 method for class 'runs'
decluster(x, threshold, ..., data, r = 1, clusterfun = "max", groups = NULL,
    replace.with, na.action = na.fail)

## S3 method for class 'declustered'
plot(x, which.plot = c("scatter", "atdf"), qu = 0.85, xlab = NULL,
    ylab = NULL, main = NULL, col = "gray", ...)

## S3 method for class 'declustered'
print(x, ...)
```

Arguments

x	An R data set to be declustered. Can be a data frame or a numeric vector. If a data frame, then which.cols must be specified. plot and print: an object returned by decluster.
data	A data frame containing the data.
threshold	numeric of length one or the size of the data over which (non-inclusive) data are to be declustered.
qu	quantile for u argument in the call to atdf.

which.cols	numeric of length one or two. The first component tells which column is the one to decluster, and the second component tells which, if any, column is to serve as groups.
which.plot	character string naming the type of plot to make.
method	character string naming the declustering method to employ.
clusterfun	character string naming a function to be applied to the clusters (the returned value is used). Typically, for extreme value analysis (EVA), this will be the cluster maximum (default), but other options are ok as long as they return a single number.
groups	numeric of length x giving natural groupings that should be considered as separate clusters. For example, suppose data cover only summer months across several years. It would probably not make sense to decluster the data across years (i.e., a new cluster should be defined if they occur in different years).
r	integer run length stating how many threshold deficits should be used to define a new cluster.
replace.with	number, NaN, Inf, -Inf, or NA. What should the remaining values in the cluster be replaced with? The default replaces them with threshold, which for most EVA purposes is ideal.
na.action	function to be called to handle missing values.
xlab, ylab, main, col	optioal arguments to the plot function. If not used, then reasonable default values are used.
...	optional arguments to decluster.runs or clusterfun. plot: optional arguments to plot. Not used by print.

Details

Runs declustering (see Coles, 2001 sec. 5.3.2): Extremes separated by fewer than r non-extremes belong to the same cluster.

Intervals declustering (Ferro and Segers, 2003): Extremes separated by fewer than r non-extremes belong to the same cluster, where r is the nc-th largest interexceedance time and nc, the number of clusters, is estimated from the extremal index, theta, and the times between extremes. Setting theta = 1 causes each extreme to form a separate cluster.

The print statement will report the resulting extremal index estimate based on either the runs or intervals estimate depending on the method argument as well as the number of clusters and run length. For runs declustering, the run length is the same as the argument given by the user, and for intervals method, it is an estimated run length for the resulting declustered data. Note that if the declustered data are independent, the extremal index should be close to one (if not equal to 1).

Value

A numeric vector of class "declustered" is returned with various attributes including:

call	the function call.

`data.name`	character string giving the name of the data.
`decluster.function`	value of `clusterfun` argument. This is a function.
`method`	character string naming the method. Same as input argument.
`threshold`	threshold used for declustering.
`groups`	character string naming the data used for the groups when applicable.
`run.length`	the run length used (or estimated if "intervals" method employed).
`na.action`	function used to handle missing values. Same as input argument.
`clusters`	muneric giving the clusters of threshold exceedances.

Author(s)

Eric Gilleland

References

Coles, S. (2001) *An introduction to statistical modeling of extreme values*, London, U.K.: Springer-Verlag, 208 pp.

Ferro, C. A. T. and Segers, J. (2003). Inference for clusters of extreme values. *Journal of the Royal Statistical Society B*, **65**, 545–556.

See Also

`extremalindex`, `fevd`

Examples

```
y <- rnorm(100, mean=40, sd=20)
y <- apply(cbind(y[1:99], y[2:100]), 1, max)
bl <- rep(1:3, each=33)

ydc <- decluster(y, quantile(y, probs=c(0.75)), r=1, groups=bl)
ydc

plot(ydc)

## Not run:
look <- decluster(-Tphap$MinT, threshold=-73)
look
plot(look)

# The code cannot currently grab data of the type of above.
# Better:
y <- -Tphap$MinT
look <- decluster(y, threshold=-73)
look
plot(look)

# Even better.  Use a non-constant threshold.
```

```
u <- -70 - 7 *(Tphap$Year - 48)/42
look <- decluster(y, threshold=u)
look
plot(look)

# Better still: account for the fact that there are huge
# gaps in data from one year to another.
bl <- Tphap$Year - 47
look <- decluster(y, threshold=u, groups=bl)
look
plot(look)


# Now try the above with intervals declustering and compare
look2 <- decluster(y, threshold=u, method="intervals", groups=bl)
look2
dev.new()
plot(look2)
# Looks about the same,
# but note that the run length is estimated to be 5.
# Same resulting number of clusters, however.
# May result in different estimate of the extremal
# index.


#
fit <- fevd(look, threshold=u, type="GP", time.units="62/year")
fit
plot(fit)

# cf.
fit2 <- fevd(-MinT~1, Tphap, threshold=u, type="GP", time.units="62/year")
fit2
dev.new()
plot(fit2)

#
fit <- fevd(look, threshold=u, type="PP", time.units="62/year")
fit
plot(fit)

# cf.
fit2 <- fevd(-MinT~1, Tphap, threshold=u, type="PP", time.units="62/year")
fit2
dev.new()
plot(fit2)



## End(Not run)
```

Denmint *Denver Minimum Temperature*

Description

Daily minimum temperature (degrees centigrade) for Denver, Colorado from 1949 through 1999.

Usage

```
data(Denmint)
```

Format

A data frame with 18564 observations on the following 5 variables.

Time a numeric vector indicating the line number (time from first entry to the last).

Year a numeric vector giving the year.

Mon a numeric vector giving the month of each year.

Day a numeric vector giving the day of the month.

Min a numeric vector giving the minimum temperature in degrees Fahrenheit.

Source

Originally, the data came from the Colorado Climate Center at Colorado State University. The Colorado state climatologist office no longer provides these data without charge. They can be obtained from the NOAA/NCDC web site, but there are slight differences (i.e., some missing values for temperature).

Examples

```
data(Denmint)
plot( Denmint[,3], Denmint[,5], xlab="", xaxt="n", ylab="Minimum Temperature (deg. F)")
axis(1,at=1:12,labels=c("Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"))
```

Denversp *Denver July hourly precipitation amount.*

Description

Hourly precipitation (mm) for Denver, Colorado in the month of July from 1949 to 1990.

Usage

```
data(Denversp)
```

Format

A data frame with 31247 observations on the following 4 variables.

Year a numeric vector giving the number of years from 1900.

Day a numeric vector giving the day of the month.

Hour a numeric vector giving the hour of the day (1 to 24).

Prec a numeric vector giving the precipitation amount (mm).

Details

These observations are part of an hourly precipitation dataset for the United States that has been critically assessed by Collander et al. (1993). The Denver hourly precipitation dataset is examined further by Katz and Parlange (1995). Summer precipitation in this region near the eastern edge of the Rocky Mountains is predominantly of local convective origin (Katz and Parlange (1005)).

Source

Katz, R. W. and Parlange, M. B. (1995) Generalizations of chain-dependent processes: Application to hourly precipitation, *Water Resources Research* **31**, (5), 1331–1341.

References

Collander, R. S., Tollerud, E. I., Li, L., and Vironst-Lazar, A. (1993) Hourly precipitation data and station histories: A research assessment, in *Preprints, Eighth Symposium on Meteorological Observations and Instrumentation, American Meteorological Society*, Boston, 153–158.

Examples

```
data(Denversp)
plot( Denversp[,1], Denversp[,4], xlab="", ylab="Hourly precipitation (mm)", xaxt="n")
axis(1,at=c(50,60,70,80,90),labels=c("1950","1960","1970","1980","1990"))
```

---

devd *Extreme Value Distributions*

Description

Density, distribution function (df), quantile function and random generation for the generalized extreme value and generalized Pareto distributions.

Usage

```
devd(x, loc = 0, scale = 1, shape = 0, threshold = 0, log = FALSE,
    type = c("GEV", "GP"))

pevd(q, loc = 0, scale = 1, shape = 0, threshold = 0, lambda = 1,
    npy, type = c("GEV", "GP", "PP", "Gumbel", "Frechet", "Weibull",
        "Exponential", "Beta", "Pareto"), lower.tail = TRUE, log.p = FALSE)

qevd(p, loc = 0, scale = 1, shape = 0, threshold = 0,
   type = c("GEV", "GP", "PP", "Gumbel", "Frechet", "Weibull", "Exponential", "Beta",
    "Pareto"), lower.tail = TRUE)

revd(n, loc = 0, scale = 1, shape = 0, threshold = 0,
    type = c("GEV", "GP"))
```

Arguments

x, q	numeric vector of quantiles.
p	numeric vector of probabilities. Must be between 0 and 1 (non-inclusive).
n	number of observations to draw.
npy	Number of points per period (period is usually year). Currently not used.
lambda	Event frequency base rate. Currently not used.
loc, scale, shape	location, scale and shape parameters. Each may be a vector of same length as x (devd or length n for revd. Must be length 1 for pevd and qevd.
threshold	numeric giving the threshold for the GP df. May be a vector of same length as x (devd or length n for revd. Must be length 1 for pevd and qevd.
log, log.p	logical; if TRUE, probabilites p are given as log(p).
lower.tail	logical; if TRUE (default), probabilities are P[X <= x] otherwise, P[X > x].
type	character; one of "GEV" or "GP" describing whether to use the GEV or GP.

Details

The extreme value distributions (EVD's) are generalized extreme value (GEV) or generalized Pareto (GP); if type is "PP", then pevd changes it to "GEV". The point process characterization is an equivalent form, but is not handled here. The GEV df is given by

Pr(X <= x) = G(x) = exp[-(1 + shape * (x - location)/scale)^(-1/shape)]

for 1 + shape*(x - location) > 0 and scale > 0. It the shape parameter is zero, then the df is defined by continuity and simplies to

G(x) = exp(-exp((x - location)/scale)).

The GEV df is often called a family of df's because it encompasses the three types of EVD's: Gumbel (shape = 0, light tail), Frechet (shape > 0, heavy tail) and the reverse Weibull (shape < 0, bounded upper tail at location - scale/shape). It was first found by R. von Mises (1936) and also independently noted later by meteorologist A. F. Jenkins (1955). It enjoys theretical support for modeling maxima taken over large blocks of a series of data.

The generalized Pareo df is given by (Pickands, 1975)

Pr(X <= x) = F(x) = 1 - [1 + shape * (x - threshold)/scale]^(-1/shape)

where 1 + shape * (x - threshold)/scale > 0, scale > 0, and x > threshold. If shape = 0, then the GP df is defined by continuity and becomes

F(x) = 1 - exp(-(x - threshold)/scale).

There is an approximate relationship between the GEV and GP df's where the GP df is approximately the tail df for the GEV df. In particular, the scale parameter of the GP is a function of the threshold (denote it scale.u), and is equivalent to scale + shape*(threshold - location) where scale, shape and location are parameters from the "equivalent" GEV df. Similar to the GEV df, the shape parameter determines the tail behavior, where shape = 0 gives rise to the exponential df (light tail), shape > 0 the Pareto df (heavy tail) and shape < 0 the Beta df (bounded upper tail at location - scale.u/shape). Theoretical justification supports the use of the GP df family for modeling excesses over a high threshold (i.e., y = x - threshold). It is assumed here that x, q describe x (not y = x - threshold). Similarly, the random draws are y + threshold.

See Coles (2001) and Reiss and Thomas (2007) for a very accessible text on extreme value analysis and for more theoretical texts, see for example, Beirlant et al. (2004), de Haan and Ferreira (2006), as well as Reiss and Thomas (2007).

Value

'devd' gives the density function, 'pevd' gives the distribution function, 'qevd' gives the quantile function, and 'revd' generates random deviates for the GEV or GP df depending on the type argument.

Note

There is a similarity between the location parameter of the GEV df and the threshold for the GP df. For clarity, two separate arguments are emplyed here to distinguish the two instead of, for example, just using the location parameter to describe both.

Author(s)

Eric Gilleland

References

Beirlant, J., Goegebeur, Y., Teugels, J. and Segers, J. (2004) *Statistics of Extremes: Theory and Applications*. Chichester, West Sussex, England, UK: Wiley, ISBN 9780471976479, 522pp.

Coles, S. (2001) *An introduction to statistical modeling of extreme values*, London, U.K.: Springer-Verlag, 208 pp.

de Haan, L. and Ferreira, A. (2006) *Extreme Value Theory: An Introduction*. New York, NY, USA: Springer, 288pp.

Jenkinson, A. F. (1955) The frequency distribution of the annual maximum (or minimum) of meteorological elements. *Quart. J. R. Met. Soc.*, **81**, 158–171.

Pickands, J. (1975) Statistical inference using extreme order statistics. *Annals of Statistics*, **3**, 119–131.

Reiss, R.-D. and Thomas, M. (2007) *Statistical Analysis of Extreme Values: with applications to insurance, finance, hydrology and other fields*. Birkhauser, 530pp., 3rd edition.

von Mises, R. (1936) La distribution de la plus grande de n valeurs, *Rev. Math. Union Interbalcanique* **1**, 141–160.

See Also

fevd

Examples

```
## GEV df (Frechet type)
devd(2:4, 1, 0.5, 0.8) # pdf
pevd(2:4, 1, 0.5, 0.8) # cdf
qevd(seq(1e-8,1-1e-8,,20), 1, 0.5, 0.8) # quantiles
revd(10, 1, 0.5, 0.8) # random draws

## GP df
devd(2:4, scale=0.5, shape=0.8, threshold=1, type="GP")
pevd(2:4, scale=0.5, shape=0.8, threshold=1, type="GP")
qevd(seq(1e-8,1-1e-8,,20), scale=0.5, shape=0.8, threshold=1, type="GP")
revd(10, scale=0.5, shape=0.8, threshold=1, type="GP")

## Not run:
# The fickleness of extremes.
z1 <- revd(100, 1, 0.5, 0.8)
hist(z1, breaks="FD", freq=FALSE, xlab="GEV distributed random variables", col="darkblue")
lines(seq(0,max(z1),,200), devd(seq(0,max(z1),,200), 1, 0.5, 0.8), lwd=1.5, col="yellow")
lines(seq(0,max(z1),,200), devd(seq(0,max(z1),,200), 1, 0.5, 0.8), lwd=1.5, lty=2)

z2 <- revd(100, 1, 0.5, 0.8)
qqplot(z1, z2)

z3 <- revd(100, scale=0.5, shape=0.8, threshold=1, type="GP")
# Or, equivalently
z4 <- revd(100, 5, 0.5, 0.8, 1, type="GP") # the "5" is ignored.
qqplot(z3, z4)

# Just for fun.
qqplot(z1, z3)

# Compare
par(mfrow=c(2,2))
plot(density(z1), xlim=c(0,100), ylim=c(0,1))
plot(density(z2), xlim=c(0,100), ylim=c(0,1))
plot(density(z3), xlim=c(0,100), ylim=c(0,1))
plot(density(z4), xlim=c(0,100), ylim=c(0,1))

# Three types
x <- seq(0,10,,200)
par(mfrow=c(1,2))
plot(x, devd(x, 1, 1, -0.5), type="l", col="blue", lwd=1.5,
```

```
    ylab="GEV df")
# Note upper bound at 1 - 1/(-0.5) = 3 in above plot.

lines(x, devd(x, 1, 1, 0), col="lightblue", lwd=1.5)
lines(x, devd(x, 1, 1, 0.5), col="darkblue", lwd=1.5)
legend("topright", legend=c("(reverse) Weibull", "Gumbel", "Frechet"),
    col=c("blue", "lightblue", "darkblue"), bty="n", lty=1, lwd=1.5)

plot(x, devd(x, 1, 1, -0.5, 1, type="GP"), type="l", col="blue", lwd=1.5,
    ylab="GP df")
lines(x, devd(x, 1, 1, 0, 1, type="GP"), col="lightblue", lwd=1.5)
lines(x, devd(x, 1, 1, 0.5, 1, type="GP"), col="darkblue", lwd=1.5)
legend("topright", legend=c("Beta", "Exponential", "Pareto"),
    col=c("blue", "lightblue", "darkblue"), bty="n", lty=1, lwd=1.5)

# Emphasize the tail differences more by using different scale parameters.
par(mfrow=c(1,2))
plot(x, devd(x, 1, 0.5, -0.5), type="l", col="blue", lwd=1.5,
    ylab="GEV df")
lines(x, devd(x, 1, 1, 0), col="lightblue", lwd=1.5)
lines(x, devd(x, 1, 2, 0.5), col="darkblue", lwd=1.5)
legend("topright", legend=c("(reverse) Weibull", "Gumbel", "Frechet"),
    col=c("blue", "lightblue", "darkblue"), bty="n", lty=1, lwd=1.5)

plot(x, devd(x, 1, 0.5, -0.5, 1, type="GP"), type="l", col="blue", lwd=1.5,
    ylab="GP df")
lines(x, devd(x, 1, 1, 0, 1, type="GP"), col="lightblue", lwd=1.5)
lines(x, devd(x, 1, 2, 0.5, 1, type="GP"), col="darkblue", lwd=1.5)
legend("topright", legend=c("Beta", "Exponential", "Pareto"),
    col=c("blue", "lightblue", "darkblue"), bty="n", lty=1, lwd=1.5)


## End(Not run)
```

`distill.fevd` *Distill Parameter Information*

Description

Distill parameter information (and possibly other pertinent inforamtion) from fevd objects.

Usage

```
## S3 method for class 'fevd'
distill(x, ...)

## S3 method for class 'fevd.bayesian'
distill(x, cov = TRUE, FUN = "mean", burn.in = 499, ...)

## S3 method for class 'fevd.lmoments'
```

```
distill(x, ...)

## S3 method for class 'fevd.mle'
distill(x, cov = TRUE, ...)
```

Arguments

x	list object returned by fevd.
...	Not used.
cov	logical; should the parameter covariance be returned with the parameters (if TRUE, they are returned as a vector concatenated to the end of the returned value).
FUN	character string naming a function to use to estimate the parameters from the MCMC sample. The function is applied to each column of the results component of the returned fevd object.
burn.in	The first burn.in values are thrown out before calculating anything from the MCMC sample.

Details

Obtaining just the basic information from the fits:

distill: The distill method function works on fevd output to obtain only pertinent information and output it in a very user-friendly format (i.e., a single vector). Mostly, this simply means returning the parameter estimates, but for some methods, more information (e.g., the optimized negative log-likelihood value and parameter covariances) can also be returned. In the case of the parameter covariances (returned if cov = TRUE), if np is the number of parameters in the model, the covariance matrix can be obtained by peeling off the last np^2 values of the vector, call it v, and using v <- matrix(v, np, np).

As with ci, only distill need be called by the user. The appropriate choice of the other functions is automatically determined from the fevd fitted object.

Value

numeric vector giving the parameter values, and if estimation method is MLE/GMLE, then the negative log-likelihood. If the estimation method is MLE/GMLE or Bayesian, then the parameter covariance values (collapsed with c) are concatenated to the end as well.

Author(s)

Eric Gilleland

See Also

fevd, ci.fevd, distillery::distill

Examples

```
data(Fort)

fit <- fevd(Prec, Fort, threshold=0.395, type="PP", units="inches", verbose=TRUE)
fit

distill(fit)
```

---

erlevd *Effective Return Levels*

Description

Find the so-called effective return levels for non-stationary extreme value distributions (EVDs).

Usage

```
erlevd(x, period = 100)
```

Arguments

x	A list object of class "fevd".
period	number stating for what return period the effective return levels should be calculated.

Details

Return levels are the same as the quantiles for the GEV df. For the GP df, they are very similar to the quantiles, but with the event frequency taken into consideration. Effective return levels are the return levels obtained for given parameter/threshold values of a non-stationary model. For example, suppose the df for data are modeled as a GEV(location(t) = mu0 + mu1 * t, scale, shape), where 't' is time. Then for any specific given time, 't', return levels can be found. This is done for each value of the covariate(s) used to fit the model to the data. See, for example, Gilleland and Katz (2011) for more details.

This function is called by the `plot` method function for "fevd" objects when the models are non-stationary.

Value

A vector of length equal to the length of the data used to obtain the fit. When `x` is from a PP fit with blocks, a vector of length equal to the number of blocks.

Author(s)

Eric Gilleland

References

Gilleland, E. and Katz, R. W. (2011). New software to analyze how extremes change over time. *Eos*, 11 January, **92**, (2), 13–14.

See Also

fevd, rlevd, rextRemes, pextRemes, plot.fevd

Examples

```
data(PORTw)

fit <- fevd(TMX1, PORTw, location.fun=~AOindex, units="deg C")
fit
tmp <- erlevd(fit, period=20)

## Not run:
# Currently, the ci function does not work for effective
# return levels.  There were coding issues encountered.
# But, could try:
#
z <- rextRemes(fit, n=500)
dim(z)
# 500 randomly drawn samples from the
# fitted model.  Each row is a sample
# of data from the fitted model of the
# same length as the data.  Each column
# is a separate sample.

sam <- numeric(0)
for( i in 1:500) {
    cat(i, " ")
    dat <- data.frame(z=z[,i], AOindex=PORTw$AOindex)
    res <- fevd(z, dat, location.fun=~AOindex)
    sam <- cbind(sam, c(erlevd(res)))
}
cat("\n")

dim(sam)

a <- 0.05
res <- apply(sam, 1, quantile, probs=c(a/2, 1 - a/2))
nm <- rownames(res)

res <- cbind(res[1,], tmp, res[2,])
colnames(res) <- c(nm[1], "Estimated 20-year eff. ret. level", nm[2])
res


## End(Not run)
```

`extremalindex` *Extemal Index*

Description

Estimate the extremal index.

Usage

```
extremalindex(x, threshold, method = c("intervals", "runs"), run.length = 1,
    na.action = na.fail, ...)

## S3 method for class 'extremalindex'
ci(x, alpha = 0.05, R = 502, return.samples = FALSE, ...)

## S3 method for class 'extremalindex'
print(x, ...)
```

Arguments

`x`	A data vector. `ci` and `print`: output from `extremalindex`.
`threshold`	numeric of length one or the length of `x` giving the value above which (non-inclusive) the extremal index should be calculated.
`method`	character string stating which method should be used to estimate the extremal index.
`run.length`	For runs declustering only, an integer giving the number of threshold deficits to be considered as starting a new cluster.
`na.action`	function to handle missing values.
`alpha`	number between zero and one giving the (1 - alpha) * 100 percent confidence level. For example, alpha = 0.05 corresponds to 95 percent confidence; alpha is the significance level (or probability of type I errors) for hypothesis tests based on the CIs.
`R`	Number of replicate samples to use in the bootstrap procedure.
`return.samples`	logical; if TRUE, the bootstrap replicate samples will be returned instead of CIs. This is useful, for example, if one wishes to find CIs using a better method than the one used here (percentile method).
`...`	optional arguments to `decluster`. Not used by `ci` or `print`.

Details

The extremal index is a useful indicator of how much clustering of exceedances of a threshold occurs in the limit of the distribution. For independent data, theta = 1, (though the converse is does not hold) and if theta < 1, then there is some dependency (clustering) in the limit.

There are many possible estimators of the extremal index. The ones used here are runs declustering (e.g., Coles, 2001 sec. 5.3.2) and the intervals estimator described in Ferro and Segers (2003). It is unbiased in the mean and can be used to estimate the number of clusters, which is also done by this function.

Value

A numeric vector of length three and class "extremalindex" is returned giving the estimated extremal index, the number of clusters and the run length. Also has attributes including:

`cluster`	the resulting clusters.
`method`	Same as argument above.
`data.name`	character vector giving the name of the data used, and possibly the data frame or matrix and column name, if applicable.
`data.call`	character string giving the actual argument passed in for x. May be the same as data.name.
`call`	the function call.
`na.action`	function used for handling missing values. Same as argument above.
`threshold`	the threshold used.

Author(s)

Eric Gilleland

References

Coles, S. (2001) *An introduction to statistical modeling of extreme values*, London, U.K.: Springer-Verlag, 208 pp.

Ferro, C. A. T. and Segers, J. (2003). Inference for clusters of extreme values. *Journal of the Royal Statistical Society B*, **65**, 545–556.

See Also

`decluster`, `fevd`

Examples

```
data(Fort)

extremalindex(Fort$Prec, 0.395, method="runs", run.length=9, blocks=Fort$year)

## Not run:
tmp <- extremalindex(Fort$Prec, 0.395, method="runs", run.length=9, blocks=Fort$year)
tmp
ci(tmp)

tmp <- extremalindex(Fort$Prec, 0.395, method="intervals", run.length=9, blocks=Fort$year)
tmp
ci(tmp)
```

```
## End(Not run)
```

FCwx *Fort Collins, Colorado Weather Data*

Description

Weather data from Fort Collins, Colorado, U.S.A. from 1900 to 1999.

Usage

```
data(FCwx)
```

Format

The format is: chr "FCwx"

Details

Data frame with components:

Year: integer years from 1900 to 1999,

Mn: integer months from 1 to 12,

Dy: integer days of the month (i.e., from 1 to 28, 29, 30 or 31 depending on the month/year),

MxT: integer valued daily maximum temperature (degrees Fahrenheit),

MnT: integer valued daily minimum temperature (degrees Fahrenheit),

Prec: numeric giving the daily accumulated precipitation (inches),

Snow: numeric daily accumulated snow amount,

SnCv: numeric daily snow cover amount

Source

Originally from the Colorado Climate Center at Colorado State University. The Colorado state climatologist office no longer provides this data without charge. The data can be obtained from the NOAA/NCDC web site, but there are slight differences (i.e., some missing values for temperature).

References

Katz, R. W., Parlange, M. B. and Naveau, P. (2002) Statistics of extremes in hydrology. *Advances in Water Resources*, **25**, 1287–1304.

Examples

```
data(FCwx)
str(FCwx)
plot(FCwx$Mn, FCwx$Prec)
plot(1:36524, FCwx$MxT, type="l")
```

fevd *Fit An Extreme Value Distribution (EVD) to Data*

Description

Fit a univariate extreme value distribution functions (e.g., GEV, GP, PP, Gumbel, or Exponential) to data; possibly with covariates in the parameters.

Usage

```
fevd(x, data, threshold = NULL, threshold.fun = ~1, location.fun = ~1,
    scale.fun = ~1, shape.fun = ~1, use.phi = FALSE,
    type = c("GEV", "GP", "PP", "Gumbel", "Exponential"),
    method = c("MLE", "GMLE", "Bayesian", "Lmoments"), initial = NULL,
    span, units = NULL, time.units = "days", period.basis = "year",
    na.action = na.fail, optim.args = NULL, priorFun = NULL,
    priorParams = NULL, proposalFun = NULL, proposalParams = NULL,
    iter = 9999, weights = 1, blocks = NULL, verbose = FALSE)

## S3 method for class 'fevd'
plot(x, type = c("primary", "probprob", "qq", "qq2",
    "Zplot", "hist", "density", "rl", "trace"),
    rperiods = c(2, 5, 10, 20, 50, 80, 100, 120, 200, 250, 300, 500, 800),
    a = 0, hist.args = NULL, density.args = NULL, d = NULL, ...)

## S3 method for class 'fevd.bayesian'
plot(x, type = c("primary", "probprob", "qq", "qq2",
    "Zplot", "hist", "density", "rl", "trace"),
    rperiods = c(2, 5, 10, 20, 50, 80, 100, 120, 200, 250, 300, 500, 800),
    a = 0, hist.args = NULL, density.args = NULL, burn.in = 499, d = NULL, ...)

## S3 method for class 'fevd.lmoments'
plot(x, type = c("primary", "probprob", "qq", "qq2",
    "Zplot", "hist", "density", "rl", "trace"),
    rperiods = c(2, 5, 10, 20, 50, 80, 100, 120, 200, 250, 300, 500, 800),
    a = 0, hist.args = NULL, density.args = NULL, d = NULL, ...)

## S3 method for class 'fevd.mle'
plot(x, type = c("primary", "probprob", "qq", "qq2",
    "Zplot", "hist", "density", "rl", "trace"),
    rperiods = c(2, 5, 10, 20, 50, 80, 100, 120, 200, 250, 300, 500, 800),
    a = 0, hist.args = NULL, density.args = NULL, period = "year",
    prange = NULL, d = NULL, ...)

## S3 method for class 'fevd'
print(x, ...)
```

```
## S3 method for class 'fevd'
summary(object, ...)

## S3 method for class 'fevd.bayesian'
summary(object, FUN = "mean", burn.in = 499, ...)

## S3 method for class 'fevd.lmoments'
summary(object, ...)

## S3 method for class 'fevd.mle'
summary(object, ...)
```

Arguments

x	fevd: x can be a numeric vector, the name of a column of data or a formula giving the data to which the EVD is to be fit. In the case of the latter two, the data argument must be specified, and must have appropriately named columns. plot and print method functions: any list object returned by fevd.
object	A list object of class "fevd" as returned by fevd.
data	A data frame object with named columns giving the data to be fit, as well as any data necessary for modeling non-stationarity through the threshold and/or any of the parameters.
threshold	numeric (single or vector). If fitting a peak over threshold (POT) model (i.e., type = "PP", "GP", "Exponential") this is the threshold over which (non-inclusive) data (or excesses) are used to estimate the parameters of the distribution function. If the length is greater than 1, then the length must be equal to either the length of x (or number of rows of data) or to the number of unique arguments in threshold.fun.
threshold.fun	formula describing a model for the thresholds using columns from data. Any valid formula will work. data must be supplied if this argument is anything other than ~ 1. Not for use with method "Lmoments".
location.fun, scale.fun, shape.fun	formula describing a model for each parameter using columns from data. data must be supplied if any of these arguments are anything other than ~ 1.
use.phi	logical; should the log of the scale parameter be used in the numerical optimization (for method "MLE", "GMLE" and "Bayesian" only)? For the ML and GML estimation, this may make things more stable for some data.
type	fevd: character stating which EVD to fit. Default is to fit the generalized extreme value (GEV) distribution function (df). plot method function: character describing which plot(s) is (are) desired. Default is "primary", which makes a 2 by 2 panel of plots including the QQ plot of the data quantiles against the fitted model quantiles (type "qq"), a QQ plot ("qq2") of quantiles from model-simulated data against the data, a density plot of the data along with the model fitted density (type "density") and a return level plot (type "rl"). In the case of a stationary (fixed) model, the return level plot will show return levels calculated for return periods given by return.period,

along with associated CIs (calculated using default `method` arguments depending on the estimation method used in the fit. For non-stationary models, the data are plotted as a line along with associated effective return levels for return periods of 2, 20 and 100 years (unless `return.period` is specified by the user to other values. Other possible values for `type` include "hist", which is similar to "density", but shows the histogram for the data and "trace", which is not used for L-moment fits. In the case of MLE/GMLE, the trace yields a panel of plots that show the negative log-likelihood and gradient negative log-likelihood (note that the MLE gradient is currently used even for GMLE) for each of the estimated parameter(s); allowing one parameter to vary according to `prange`, while the others remain fixed at their estimated values. In the case of Bayesian estimation, the "trace" option creates a panel of plots showing the posterior df and MCMC trace for each parameter.

`method` `fevd`: character naming which type of estimation method to use. Default is to use maximum likelihood estimation (MLE).

`initial` A list object with any named parameter component giving the initial value estimates for starting the numerical optimization (MLE/GMLE) or the MCMC iterations (Bayesian). In the case of MLE/GMLE, it is best to obtain a good intial guess, and in the Bayesian case, it is perhaps better to choose poor initial estimates. If NULL (default), then L-moments estimates and estimates based on Gumbel moments will be calculated, and whichever yields the lowest negative log-likelihood is used. In the case of `type` "PP", an additional MLE/GMLE estimate is made for the generalized Pareto (GP) df, and parameters are converted to those of the Poisson Process (PP) model. Again, the initial estimates yielding the lowest negative log-likelihoo value are used for the initial guess.

`span` single numeric giving the number of years (or other desired temporal unit) in the data set. Only used for POT models, and only important in the estimation for the PP model, but important for subsequent estimates of return levels for any POT model. If missing, it will be calculated using information from `time.units`.

`units` (optional) character giving the units of the data, which if given may be used subsequently (e.g., on plot axis labels, etc.).

`time.units` character string that must be one of "hours", "minutes", "seconds", "days", "months", "years", "m/hour", "m/minute", "m/second", "m/day", "m/month", or "m/year"; where m is a number. If `span` is missing, then this argument is used in determining the value of `span`. It is also returned with the output and used subsequently for plot labelling, etc.

`period.basis` character string giving the units for the period. Used only for plot labelling and naming output vectors from some of the method functions (e.g., for establishing what the period represents for the return period).

`rperiods` numeric vector giving the return period(s) for which it is desired to calculate the corresponding return levels.

`period` character string naming the units for the return period.

`burn.in` The first `burn.in` values are thrown out before calculating anything from the MCMC sample.

`a` when plotting empirical probabilies and such, the function `ppoints` is called, which has this argument `a`.

d — numeric determining how to scale the rate parameter for the point process. If NULL, the function will attempt to scale based on the values of period.basis and time.units, the first of which must be "year" and the second of which must be one of "days", "months", "years", "hours", "minutes" or "seconds". If none of these are the case, then d should be specified, otherwise, it is not necessary.

density.args, hist.args — named list object containing arguments to the density and hist functions, respectively.

na.action — function to be called to handle missing values. Generally, this should remain at the default (na.fail), and the user should take care to impute missing values in an appropriate manner as it may have serious consequences on the results.

optim.args — A list with named components matching exactly any arguments that the user wishes to specify to optim, which is used only for MLE and GMLE methods. By default, the "BFGS" method is used along with grlevd for the gradient argument. Generally, the grlevd function is used for the gr option unless the user specifies otherwise, or the optimization method does not take gradient information.

priorFun — character naming a prior df to use for methods GMLE and Bayesian. The default for GMLE (not including Gumbel or Exponential types) is to use the one suggested by Martins and Stedinger (2000, 2001) on the shape parameter; a beta df on -0.5 to 0.5 with parameters p and q. Must take x as its first argument for method "GMLE". Optional arguments for the default function are p and q (see details section).

The default for Bayesian estimation is to use normal distribution functions. For Bayesian estimation, this function must take theta as its first argument.

Note: if this argument is not NULL and method is set to "MLE", it will be changed to "GMLE".

priorParams — named list containing any prior df parameters (where the list names are the same as the function argument names). Default for GMLE (assuming the default function is used) is to use q = 6 and p = 9. Note that in the Martins and Stedinger (2000, 2001) papers, they use a different EVD parametrization than is used here such that a positive shape parameter gives the upper bounded distribution instead of the heavy-tail one (as emloyed here). To be consistent with these papers, p and q are reversed inside the code so that they have the same interpretation as in the papers.

Default for Bayesian estimation is to use ML estimates for the means of each parameter (may be changed using m, which must be a vector of same length as the number of parameters to be estimated (i.e., if using the default prior df)) and a standard deviation of 10 for all other parameters (again, if using the default prior df, may be changed using v, which must be a vector of length equal to the number of parameters).

proposalFun — For Bayesian estimation only, this is a character naming a function used to generate proposal parameters at each iteration of the MCMC. If NULL (default), a random walk chain is used whereby if theta.i is the current value of the parameter, the proposed new parameter theta.star is given by theta.i + z, where z is drawn at random from a normal df.

proposalParams	A named list object describing any optional arguments to the proposalFun function. All functions must take argument p, which must be a vector of the parameters, and ind, which is used to identify which parameter is to be proposed. The default proposalFun function takes additional arguments mean and sd, which must be vectors of length equal to the number of parameters in the model (default is to use zero for the mean of z for every parameter and 0.1 for its standard deviation).
iter	Used only for Bayesian estimation, this is the number of MCMC iterations to do.
weights	numeric of length 1 or n giving weights to be applied in the likelihood calculations (e.g., if there are data points to be weighted more/less heavily than others).
blocks	An optional list containing information required to fit point process models in a computationally-efficient manner by using only the exceedances and not the observations below the threshold(s). See details for further information.
FUN	character string naming a function to use to estimate the parameters from the MCMC sample. The function is applied to each column of the results component of the returned fevd object.
verbose	logical; should progress information be printed to the screen? If TRUE, for MLE/GMLE, the argument trace will be set to 6 in the call to optim.
prange	matrix whose columns are numeric vectors of length two for each parameter in the model giving the parameter range over which trace plots should be made. Default is to use either +/- 2 * std. err. of the parameter (first choice) or, if the standard error cannot be calculated, then +/- 2 * log2(abs(parameter)). Typically, these values seem to work very well for these plots.
...	Not used by most functions here. Optional arguments to plot for the various plot method functions. In the case of the summary method functions, the logical argument silent may be passed to suppress (if TRUE) printing any information to the screen.

Details

See text books on extreme value analysis (EVA) for more on univariate EVA (e.g., Coles, 2001 and Reiss and Thomas, 2007 give fairly accessible introductions to the topic for most audiences; and Beirlant et al., 2004, de Haan and Ferreira, 2006, as well as Reiss and Thomas, 2007 give more complete theoretical treatments). The extreme value distributions (EVDs) have theoretical support for analyzing extreme values of a process. In particular, the generalized extreme value (GEV) df is appropriate for modeling block maxima (for large blocks, such as annual maxima), the generalized Pareto (GP) df models threshold excesses (i.e., x - u | x > u and u a high threshold).

The GEV df is given by

Pr(X <= x) = G(x) = exp[-(1 + shape*(x - location)/scale)^(-1/shape)]

for 1 + shape*(x - location) > 0 and scale > 0. If the shape parameter is zero, then the df is defined by continuity and simplies to

G(x) = exp(-exp((x - location)/scale)).

The GEV df is often called a family of distribution functions because it encompasses the three types of EVDs: Gumbel (shape = 0, light tail), Frechet (shape > 0, heavy tail) and the reverse

Weibull (shape < 0, bounded upper tail at location - scale/shape). It was first found by R. von Mises (1936) and also independently noted later by meteorologist A. F. Jenkins (1955). It enjoys theretical support for modeling maxima taken over large blocks of a series of data.

The generalized Pareo df is given by (Pickands, 1975)

Pr(X <= x) = F(x) = 1 - [1 + shape*(x - threshold)/scale]^(-1/shape)

where 1 + shape*(x - threshold)/scale > 0, scale > 0, and x > threshold. If shape = 0, then the GP df is defined by continuity and becomes

F(x) = 1 - exp(-(x - threshold)/scale).

There is an approximate relationship between the GEV and GP distribution functions where the GP df is approximately the tail df for the GEV df. In particular, the scale parameter of the GP is a function of the threshold (denote it scale.u), and is equivalent to scale + shape*(threshold - location) where scale, shape and location are parameters from the "equivalent" GEV df. Similar to the GEV df, the shape parameter determines the tail behavior, where shape = 0 gives rise to the exponential df (light tail), shape > 0 the Pareto df (heavy tail) and shape < 0 the Beta df (bounded upper tail at location - scale.u/shape). Theoretical justification supports the use of the GP df family for modeling excesses over a high threshold (i.e., y = x - threshold). It is assumed here that `x`, `q` describe x (not y = x - threshold). Similarly, the random draws are y + threshold.

If interest is in minima or deficits under a low threshold, all of the above applies to the negative of the data (e.g., - max(-X_1,...,-X_n) = min(X_1, ..., X_n)) and `fevd` can be used so long as the user first negates the data, and subsequently realizes that the return levels (and location parameter) given will be the negative of the desired return levels (and location parameter), etc.

The study of extremes often involves a paucity of data, and for small sample sizes, L-moments may give better estimates than competing methods, but penalized MLE (cf. Coles and Dixon, 1999; Martins and Stedinger, 2000; 2001) may give better estimates than the L-moments for such samples. Martins and Stedinger (2000; 2001) use the terminology generalized MLE, which is also used here.

Non-stationary models:

The current code does not allow for non-stationary models with L-moments estimation.

For MLE/GMLE (see El Adlouni et al 2007 for using GMLE in fitting models whose parameters vary) and Bayesian estimation, linear models for the parameters may be fit using formulas, in which case the `data` argument must be supplied. Specifically, the models allowed for a set of covariates, y, are:

location(y) = mu0 + mu1 * f1(y) + mu2 * f2(y) + ...

scale(y) = sig0 + sig1 * g1(y) + sig2 * g2(y) + ...

log(scale(y)) = phi(y) = phi0 + phi1 * g1(y) + phi2 * g2(y) + ...

shape(y) = xi0 + xi1 * h1(y) + xi2 * h2(y) + ...

For non-stationary fitting it is recommended that the covariates within the generalized linear models are (at least approximately) centered and scaled (see examples below). It is generally ill-advised to include covariates in the shape parameter, but there are situations where it makes sense.

Non-stationary modeling is accomplished with `fevd` by using formulas via the arguments: `threshold.fun`, `location.fun`, `scale.fun` and `shape.fun`. See examples to see how to do this.

Initial Value Estimates:

In the case of MLE/GMLE, it can be very important to get good initial estimates (e.g., see the examples below). `fevd` attempts to find such estimates, but it is also possible for the user to supply

their own initial estimates as a list object using the `initial` argument, whereby the components of the list are named according to which parameter(s) they are associated with. In particular, if the model is non-stationary, with covariates in the location (e.g., mu(t) = mu0 + mu1 * t), then `initial` may have a component named "location" that may contain either a single number (in which case, by default, the initial value for mu1 will be zero) or a vector of length two giving initial values for mu0 and mu1.

For Bayesian estimation, it is good practice to try several starting values at different points to make sure the initial values do not affect the outcome. However, if initial values are not passed in, the MLEs are used (which probably is not a good thing to do, but is more likely to yield good results).

For MLE/GMLE, two (in the case of PP, three) initial estimates are calculated along with their associated likelihood values. The initial estimates that yield the highest likelihood are used. These methods are:

1. L-moment estimates.

2. Let m = mean(xdat) and s = sqrt(6 * var(xdat)) / pi. Then, initial values assigend for the lcoation parameter when either `initial` is NULL or the location component of `initial` is NULL, are m - 0.57722 * s. When `initial` or the scale component of `initial` is NULL, the initial value for the scale parameter is taken to be s, and when `initial` or its shape component is NULL, the initial value for the shape parameter is taken to be 1e-8 (because these initial estimates are moment-based estimates for the Gumbel df, so the initial value is taken to be near zero).

3. In the case of PP, which is often the most difficult model to fit, MLEs are obtained for a GP model, and the resulting parameter estimates are converted to those of the approximately equivalent PP model.

In the case of a non-stationary model, if the default initial estimates are used, then the intercept term for each parameter is given the initial estimate, and all other parameters are set to zero initially. The exception is in the case of PP model fitting where the MLE from the GP fits are used, in which case, these parameter estimates may be non-zero.

The generalized MLE (GMLE) method:

This method places a penalty (or prior df) on the shape parameter to help ensure a better fit. The procedure is nearly identical to MLE, except the likelihood, L, is multiplied by the prior df, p(shape); and because the negative log-likelihood is used, the effect is that of subtracting this term. Currently, there is no supplied function by this package to calculate the gradient for the GMLE case, so in particular, the trace plot is not the trace of the actual negative log-likelihood (or gradient thereof) used in the estimation.

Bayesian Estimation:

It is possible to give your own prior and proposal distribution functions using the appropriate arguments listed above in the arguments section. At each iteration of the chain, the parameters are updated one at a time in random order. The default method uses a random walk chain for the proposal and normal distributions for the parameters.

Plotting output:

`plot`: The `plot` method function will take information from the `fevd` output and make any of various useful plots. The default, regardless of estimation method, is to produce a 2 by 2 panel of plots giving some common diagnostic plots. Possible types (determined by the `type` argument) include:

1. "primary" (default): yields the 2 by 2 panel of plots given by 3, 4, 6 and 7 below.

2. "probprob": Model probabilities against empirical probabilities (obtained from the `ppoints` function). A good fit should yield a straight one-to-one line of points. In the case of a non-stationary model, the data are first transformed to either the Gumbel (block maxima models) or exponential (POT models) scale, and plotted against probabilities from these standardized distribution functions. In the case of a PP model, the parameters are first converted to those of the approximately equivalent GP df, and are plotted against the empirical data threshold excesses probabilities.

3. "qq": Empirical quantiles against model quantiles. Again, a good fit will yield a straight one-to-one line of points. Generally, the qq-plot is preferred to the probability plot in 1 above. As in 2, for the non-stationary case, data are first transformed and plotted against quantiles from the standardized distributions. Also as in 2 above, in the case of the PP model, parameters are converted to those of the GP df and quantiles are from threshold excesses of the data.

4. "qq2": Similar to 3, first data are simulated from the fitted model, and then the qq-plot between them (using the function `qqplot` from this self-same package) is made between them, which also yields confidence bands. Note that for a good fitting model, this should again yield a straight one-to-one line of points, but generally, it will not be as "well-behaved" as the plot in 3. The one-to-one line and a regression line fitting the quantiles is also shown. In the case of a non-stationary model, simulations are obtained by simulating from an appropriate standardized EVD, re-ordered to follow the same ordering as the data to which the model was fit, and then back transformed using the covariates from `data` and the parameter estimates to put the simulated sample back on the original scale of the data. The PP model is handled analogously as in 2 and 3 above.

5. and 6. "Zplot": These are for PP model fits only and are based on Smith and Shively (1995). The Z plot is a diagnostic for determining whether or not the random variable, Zk, defined as the (possibly non-homogeneous) Poisson intensity parameter(s) integrated from exceedance time k - 1 to exceedance time k (beginning the series with k = 1) is independent exponentially distributed with mean 1.

For the Z plot, it is necessary to scale the Poisson intensity parameter appropriately. For example, if the data are given on a daily time scale with an annual period basis, then this parameter should be divided by, for example, 365.25. From the fitted `fevd` object, the function will try to account for the correct scaling based on the two components "period.basis" and "time.units". The former currently must be "year" and the latter must be one of "days", "months", "years", "hours", "minutes" or "seconds". If none of these are valid for your specific data (e.g., if an annual basis is not desired), then use the `d` argument to explicitly specify the correct scaling.

7. "hist": A histogram of the data is made, and the model density is shown with a blue dashed line. In the case of non-stationary models, the data are first transformed to an appropriate standardized EVD scale, and the model density line is for the self-same standardized EVD. Currently, this does not work for non-stationary POT models.

8. "density": Same as 5, but the kernel density (using function `density`) for the data is plotted instead of the histogram. In the case of the PP model, block maxima of the data are calculated and the density of these block maxima are compared to the PP in terms of the equivalent GEV df. If the model is non-stationary GEV, then the transformed data (to a stationary Gumbel df) are used. If the model is a non-stationary POT model, then currently this option is not available.

9. "rl": Return level plot. This is done on the log-scale for the abscissa in order that the type of EVD can be discerned from the shape (i.e., heavy tail distributions are concave, light tailed distributions are straight lines, and bounded upper-tailed distributions are convex, asymptoting at the upper bound). 95 percent CIs are also shown (gray dashed lines). In the case of non-stationary models, the data are plotted as a line, and the "effective" return levels (by default the 2-period (i.e.,

the median), 20-period and 100-period are used; period is usually annual) are also shown (see, e.g., Gilleland and Katz, 2011). In the case of the PP model, the equivalent GEV df (stationary model) is assumed and data points are block maxima, where the blocks are determined from information passed in the call to `fevd`. In particular, the `span` argument (which, if not passed by the user, will have been determined by `fevd` using `time.units` along with the number of points per year (which is estimated from `time.units`) are used to find the blocks over which the maxima are taken. For the non-stationary case, the equivalent GP df is assumed and parameters are converted. This helps facilitate a more meaningful plot, e.g., in the presence of a non-constant threshold, but otherwise constant parameters.

10. "trace": In each of cases (b) and (c) below, a 2 by the number of parameters panel of plots are created.

(a) L-moments: Not available for the L-moments estimation.

(b) For MLE/GMLE, the likelihood traces are shown for each parameter of the model, whereby all but one parameter is held fixed at the MLE/GMLE values, and the negative log-likelihood is graphed for varying values of the parameter of interest. Note that this differs greatly from the profile likelihood (see, e.g., `profliker`) where the likelihood is maximized over the remaining parameters. The gradient negative log-likelihoods are also shown for each parameter. These plots may be useful in diagnosing any fitting problems that might arise in practice. For ease of interpretation, the gradients are shown directly below the likleihoods for each parameter.

(c) For Bayesian estimation, the usual trace plots are shown with a gray vertical dashed line showing where the `burn.in` value lies; and a gray dashed horizontal line through the posterior mean. However, the posterior densities are also displayed for each parameter directly above the usual trace plots. It is not currently planned to allow for adding the prior dnsities to the posterior density graphs, as this can be easily implemented by the user, but is more difficult to do generally.

As with `ci` and `distill`, only `plot` need be called by the user. The appropriate choice of the other functions is automatically determined from the `fevd` fitted object.

Note that when `blocks` are provided to `fevd`, certain plots that require the full set of observations (including non-exceedances) cannot be produced.

Summaries and Printing:

`summary` and `print` method functions are available, and give different information depending on the estimation method used. However, in each case, the parameter estimates are printed to the screen. `summary` returns some useful output (similar to distill, but in a list format). The `print` method function need not be called as one can simply type the name of the `fevd` fitted object and return to execute the command (see examples below). The deviance information criterion (DIC) is calculated for the Bayesian estimation method as DIC = D(mean(theta)) + 2 * pd, where pd = mean(D(theta)) - D(mean(theta)), and D(theta) = -2 * log-likelihood evaluated at the parameter values given by theta. The means are taken over the posterior MCMC sample. The default estimation method for the parameter values from the MCMC sample is to take the mean of the sample (after removing the first burn.in samples). A good alternative is to set the `FUN` argument to "postmode" in order to obtain the posterior mode of the sample.

Using Blocks to Reduce Computation in PP Fitting:

If `blocks` is supplied, the user should provide only the exceedances and not all of the data values. For stationary models, the list should contain a component called `nBlocks` indicating the number of observations within a block, where blocks are defined in a manner analogous to that used in GEV models. For nonstationary models, the list may contain one or more of several components. For

nonstationary models with covariates, the list should contain a data component analogous to the data argument, providing values for the blocks. If the threshold varies, the list should contain a threshold component that is analogous to the threshold argument. If some of the observations within any block are missing (assuming missing at random or missing completely at random), the list should contain a proportionMissing component that is a vector with one value per block indicating the proportion of observations missing for the block. To weight the blocks, the list can contain a weights component with one value per block. Warning: to properly analyze nonstationary models, any covariates, thresholds, and weights must be constant within each block.

Value

fevd: A list object of class "fevd" is returned with components:

call	the function call. Used as a default for titles in plots and output printed to the screen (e.g., by summary).
data.name	character vector giving the name of arguments: x and data. This is used by the datagrabber method function, as well as for some plot labeling.
data.pointer, cov.pointer, cov.data, x, x.fun	Not all of these are included, and which ones are depend on how the data are passed to fevd. These may be character strings, vectors or data frames depending again on the original function call. They are used by datagrabber in order to obtain the original data. If x is a column of data, then x.fun is a formula specifying which column. Also, if x is a formula, then x.fun is this self-same formula.
in.data	logical; is the argument x a column of the argument data (TRUE) or not (FALSE)?
method	character string naming which estimation method ws used for the fit. Same as method argument.
type	character string naming which EVD was fit to the data. Same as type argument.
period.basis	character string naming the period for return periods. This is used in plot labeling and naming, e.g., output from ci. Same as the argument passed in.
units	Not present if not passed in by the user. character string naming the data units. Used for plot labeling.
par.models	A list object giving the values of the threshold and parameter function arguments, as well as a logical stating whether the log of the scale parameter is used or not. This is present even if it does not make sense (i.e., for L-moments estimation) because it is used by the is.fixedfevd function for determining whether or not a model is stationary or not.
const.loc, const.scale, const.shape	logicals stating whether the named parameter is constant (TRUE) or not (FALSE). Currently, not used, but could be useful. Still present even for L-moments estimation even though it does not really make sense in this context (will always be true).
n	the length of the data to which the model is fit.
span, npy	For POT models only, this is the estimated number of periods (usually years) and number of points per period, both estimated using time.units
na.action	character naming the function used for handling missing values.

`results` If L-moments are used, then this is a simple vector of length equal to the number of parameters of the model. For MLE/GMLE, this is a list object giving the output from `optim` (in particular, the par component is a vector giving the parameter estimates). For Bayesian estimation, this is a matrix whose rows give the results for each iteration of the MCMC simulations. The columns give the parameters, as well as an additional last column that gives the number of parameters that were updated at each iteration.

`priorFun`, `priorParams`
These are only present for GMLE and Bayesian estimation methods. They give the prior df and optional arguments used in the estimation.

`proposalFun`, `proposalParams`
These are only present for Bayesian estimation, and they give the proposal function used along with optional arguments.

`chain.info` If estimation method is "Bayesian", then this component is a matrix whose first several columns give 0 or 1 for each parameter at each iteration, where 0 indicates that the parameter was not updated and 1 that it was. The first row is NA for these columns. the last two columns give the likelihood and prior values for the current parameter values.

`chain.info` matrix whose first n columns give a one or zero depending on whether the parameter was updated or not, resp. The last two columns give the log-likelihood and prior values associated with the parameters of that sample.

`blocks` Present only if `blocks` is supplied as an argument. Will contain the input information and computed block-level information for use in post-processing the model object, in particular block-wise design matrices.

`print`: Does not return anything. Information is printed to the screen.

`summary`: Depending on the estimation method, either a numeric vector giving the parameter estimates ("Lmoments") or a list object (all other estimation methods) is returned invisibly, and if `silent` is FALSE (default), then summary information is printed to the screen. List components may include:

`par`, `se.theta` numeric vectors giving the parameter and standard error estimates, resp.

`cov.theta` matrix giving the parameter covariances.

`nllh` number giving the value of the negative log-likelihood.

`AIC`, `BIC`, `DIC` numbers giving the Akaike Information Criterion (AIC, Akaike, 1974), the Bayesian Information Criterion (BIC, Schwarz, 1978), and/or the Deviance Information Criterion, resp.

Author(s)

Eric Gilleland

References

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, **19**, 716–723.

Beirlant, J., Goegebeur, Y., Teugels, J. and Segers, J. (2004). *Statistics of Extremes: Theory and Applications*. Chichester, West Sussex, England, UK: Wiley, ISBN 9780471976479, 522pp.

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

Coles, S. G. and Dixon, M. J. (1999). Likelihood-based inference for extreme value models. *Extremes*, **2** (1), 5–23.

El Adlouni, S., Ouarda, T. B. M. J., Zhang, X., Roy, R, and Bobee, B. (2007). Generalized maximum likelihood estimators for the nonstationary generalized extreme value model. *Water Resources Research*, **43**, W03410, doi: 10.1029/2005WR004545, 13 pp.

Gilleland, E. and Katz, R. W. (2011). New software to analyze how extremes change over time. *Eos*, 11 January, **92**, (2), 13–14.

de Haan, L. and Ferreira, A. (2006). *Extreme Value Theory: An Introduction*. New York, NY, USA: Springer, 288pp.

Jenkinson, A. F. (1955). The frequency distribution of the annual maximum (or minimum) of meteorological elements. *Quart. J. R. Met. Soc.*, **81**, 158–171.

Martins, E. S. and Stedinger, J. R. (2000). Generalized maximum likelihood extreme value quantile estimators for hydrologic data. *Water Resources Research*, **36** (3), 737–744.

Martins, E. S. and Stedinger, J. R. (2001). Generalized maximum likelihood Pareto-Poisson estimators for partial duration series. *Water Resources Research*, **37** (10), 2551–2557.

Pickands, J. (1975). Statistical inference using extreme order statistics. *Annals of Statistics*, **3**, 119–131.

Reiss, R.-D. and Thomas, M. (2007). *Statistical Analysis of Extreme Values: with applications to insurance, finance, hydrology and other fields*. Birkhauser, 530pp., 3rd edition.

Schwarz, G. E. (1978). Estimating the dimension of a model. *Annals of Statistics*, **6**, 461–464.

Smith, R. L. and Shively, T. S. (1995). A point process approach to modeling trends in tropospheric ozone. *Atmospheric Environment*, **29**, 3489–3499.

von Mises, R. (1936). La distribution de la plus grande de n valeurs, *Rev. Math. Union Interbalcanique* **1**, 141–160.

See Also

`ci.fevd` for obtaining parameter and return level confidence intervals.

`distill.fevd` for stripping out a vector of parameter estimates and perhaps other pertinent information from an fevd object.

For functions to find the density, probability df, quantiles and simulate data from, an EV df, see: `devd`, `pevd`, `qevd`, `revd`

For functions to find the probability df and simulate random data from a fitted model from `fevd`, see: `pextRemes`, `rextRemes`

For functions to determine if the extreme data are independent or not, see: `extremalindex`, `atdf`

For functions to help choose a threshold, see: `threshrange.plot`, `mrlplot`

To decluster stationary dependent extremes, see: `decluster`

For more on formulas in R, see: `formula`

To grab the parameters of a fitted fevd model, see: findpars

To calculate the parameter covariance, see: optimHess, parcov.fevd

To see more about the **extRemes** method functions described here, see: distillery::ci and distillery::distill

To calculate effective return levels and CI's for MLE and Bayesian estimation of non-stationary models, see ci.rl.ns.fevd.bayesian, ci.rl.ns.fevd.mle and return.level

To obtain the original data set from a fitted fevd object, use: distillery::datagrabber

To calculate the profile likelihood, see: profliker

To test the statistical significance of nested models with additional parameters, see: lr.test

To find effective return levels for non-stationary models, see: erlevd

To determine if an fevd object is stationary or not, use: is.fixedfevd and check.constant

For more about the plots created for fevd fitted objects, see: ppoints, density, hist, qqplot

For general numerical optimization in R, see: optim

Examples

```
z <- revd(100, loc=20, scale=0.5, shape=-0.2)
fit <- fevd(z)
fit
plot(fit)
plot(fit, "trace")

## Not run:
## Fitting the GEV to block maxima.

# Port Jervis, New York winter maximum and minimum
# temperatures (degrees centigrade).
data(PORTw)

# Gumbel
fit0 <- fevd(TMX1, PORTw, type="Gumbel", units="deg C")
fit0
plot(fit0)
plot(fit0, "trace")
return.level(fit0)

# GEV
fit1 <- fevd(TMX1, PORTw, units="deg C")
fit1
plot(fit1)
plot(fit1, "trace")
return.level(fit1)
return.level(fit1, do.ci=TRUE)
ci(fit1, return.period=c(2,20,100)) # Same as above.

lr.test(fit0, fit1)
ci(fit1, type="parameter")
par(mfrow=c(1,1))
```

```
ci(fit1, type="parameter", which.par=3, xrange=c(-0.4,0.01),
    nint=100, method="proflik", verbose=TRUE)

# 100-year return level
ci(fit1, method="proflik", xrange=c(22,28), verbose=TRUE)

plot(fit1, "probprob")
plot(fit1, "qq")
plot(fit1, "hist")
plot(fit1, "hist", ylim=c(0,0.25))

# Non-stationary model.
# Location as a function of AO index.

fit2 <- fevd(TMX1, PORTw, location.fun=~AOindex, units="deg C")
fit2
plot(fit2)
plot(fit2, "trace")
# warnings are not critical here.
# Sometimes the nllh or gradients
# are not defined.

return.level(fit2)

v <- make.qcov(fit2, vals=list(mu1=c(-1, 1)))
return.level(fit2, return.period=c(2, 20, 100), qcov=v)

# Note that first row is for AOindex = -1 and second
# row is for AOindex = 1.

lr.test(fit1, fit2)
# Also compare AIC and BIC

look1 <- summary(fit1, silent=TRUE)
look1 <- c(look1$AIC, look1$BIC)

look2 <- summary(fit2, silent=TRUE)
look2 <- c(look2$AIC, look2$BIC)

# Lower AIC/BIC is better.
names(look1) <- names(look2) <- c("AIC", "BIC")
look1
look2

par(mfrow=c(1,1))
plot(fit2, "rl")


## Fitting the GP df to threshold excesses.

# Hurricane damage data.

data(damage)
```

```
ny <- tabulate(damage$Year)
# Looks like only, at most, 5 obs per year.

ny <- mean(ny[ny > 0])
fit0 <- fevd(Dam, damage, threshold=6, type="Exponential", time.units="2.05/year")
fit0
plot(fit0)
plot(fit0, "trace") # ignore the warning.

fit1 <- fevd(Dam, damage, threshold=6, type="GP", time.units="2.05/year")
fit1
plot(fit1) # ignore the warning.
plot(fit1, "trace")

return.level(fit1)

# lr.test(fit0, fit1)

# Fort Collins, CO precipitation

data(Fort)

## GP df

fit <- fevd(Prec, Fort, threshold=0.395, type="GP", units="inches", verbose=TRUE)
fit
plot(fit)
plot(fit, "trace")

ci(fit, type="parameter")
par(mfrow=c(1,1))
ci(fit, type="return.level", method="proflik", xrange=c(4,7.5), verbose=TRUE)
# Can check using locator(2).

ci(fit, type="parameter", which.par=2, method="proflik", xrange=c(0.1, 0.3),
    verbose=TRUE)
# Can check using locator(2).


## PP model.

fit <- fevd(Prec, Fort, threshold=0.395, type="PP", units="inches", verbose=TRUE)
fit
plot(fit)
plot(fit, "trace")
ci(fit, type="parameter")

# Same thing, but just to try a different optimization method.
# And, for fun, a different way of entering the data set.
fit <- fevd(Fort$Prec, threshold=0.395, type="PP",
    optim.args=list(method="Nelder-Mead"), units="inches", verbose=TRUE)
fit
```

```
plot(fit)
plot(fit, "trace")
ci(fit, type="parameter")

## PP model with blocks argument for computational efficiency # CJP2

system.time(fit <- fevd(Prec, Fort, threshold=0.395, type="PP", units="inches", verbose=TRUE))

FortSub = Fort[Fort$Prec > 0.395, ]
system.time(fit.blocks <- fevd(Prec, FortSub, threshold=0.395,
type="PP", units="inches", blocks = list(nBlocks = 100), verbose=TRUE))
fit.blocks
plot(fit.blocks)
plot(fit.blocks, "trace")
ci(fit.blocks, type="parameter")

#
# Phoenix data
#
# Summer only with 62 days per year.

data(Tphap)

plot(MinT~ Year, data=Tphap)

# GP df
fit <- fevd(-MinT ~1, Tphap, threshold=-73, type="GP", units="deg F",
    time.units="62/year", verbose=TRUE)

fit
plot(fit)
plot(fit, "trace")

# PP
fit <- fevd(-MinT ~1, Tphap, threshold=-73, type="PP", units="deg F", time.units="62/year",
    use.phi=TRUE, optim.args=list(method="BFGS", gr=NULL), verbose=TRUE)
fit
plot(fit)
plot(fit, "trace")


# Non-stationary models

fit <- fevd(Prec, Fort, threshold=0.395,
    scale.fun=~sin(2 * pi * (year - 1900)/365.25) + cos(2 * pi * (year - 1900)/365.25),
    type="GP", use.phi=TRUE, verbose=TRUE)
fit
plot(fit)
plot(fit, "trace")
ci(fit, type="parameter")

# Non-constant threshold.
```

```
# GP
fit <- fevd(Prec, Fort, threshold=0.475, threshold.fun=~I(-0.15 * cos(2 * pi * month / 12)),
    type="GP", verbose=TRUE)
fit
plot(fit)
par(mfrow=c(1,1))
plot(fit, "rl", xlim=c(0, 365))

# PP

fit <- fevd(Prec, Fort, threshold=0.475, threshold.fun=~I(-0.15 * cos(2 * pi * month / 12)),
    type="PP", verbose=TRUE)
fit
plot(fit)

## Bayesian PP with blocks for computational efficiency
## Note that 1999 iterations may not be sufficient; used here to
## minimize time spent fitting.
# CJP2
## CJP2: Eric, CRAN won't like this being run as part of the examples
## as it takes a long time; we'll probably want to wrap this in a \dontrun{}

set.seed(0)
system.time(fit <- fevd(Prec, Fort, threshold=0.395,
    scale.fun=~sin(2 * pi * (year - 1900)/365.25) + cos(2 * pi * (year - 1900)/365.25),
    type="PP", method="Bayesian", iter=1999, use.phi=TRUE, verbose=TRUE))
fit
ci(fit, type="parameter")

set.seed(0)
FortSub <- Fort[Fort$Prec > 0.395, ]
system.time(fit2 <- fevd(Prec, FortSub, threshold=0.395,
    scale.fun=~sin(2 * pi * (year - 1900)/365.25) + cos(2 * pi * (year -
1900)/365.25), type="PP", blocks = list(nBlocks= 100, data =
data.frame(year = 1900:1999)), use.phi=TRUE, method = "Bayesian",
iter=1999, verbose=TRUE))
# an order of magnitude faster

fit2
ci(fit2, type="parameter")

data(ftcanmax)

fit <- fevd(Prec, ftcanmax, units="inches")
fit

plot(fit)
par(mfrow=c(1,1))
plot(fit, "probprob")
plot(fit, "hist")
plot(fit, "hist", ylim=c(0,0.01))
plot(fit, "density", ylim=c(0,0.01))
plot(fit, "trace")
```

```
distill(fit)
distill(fit, cov=FALSE)

fit2 <- fevd(Prec, ftcanmax, location.fun=~Year)
fit2

plot(fit2)
##
# plot(fit2, "trace") # Gives warnings because of some NaNs produced
                      # (nothing to worry about).

lr.test(fit, fit2)

ci(fit)
ci(fit, type="parameter")

fit0 <- fevd(Prec, ftcanmax, type="Gumbel")
fit0

plot(fit0)
lr.test(fit0, fit)
plot(fit0, "trace")

ci(fit, return.period=c(2, 20, 100))
ci(fit, type="return.level", method="proflik", return.period=20, verbose=TRUE)

ci(fit, type="parameter", method="proflik", which.par=3, xrange=c(-0.1,0.5), verbose=TRUE)

# L-moments
fitLM <- fevd(Prec, ftcanmax, method="Lmoments", units="inches")
fitLM # less info.
plot(fitLM)
# above is slightly slower because of the parametric bootstrap
# for finding CIs in return levels.
par(mfrow=c(1,1))
plot(fitLM, "density", ylim=c(0,0.01))

# GP model.
# CJP2 : fixed so have 744/year (31 days *24 hours/day)

data(Denversp)

fitGP <- fevd(Prec, Denversp, threshold=0.5, type="GP", units="mm",
    time.units="744/year", verbose=TRUE)

fitGP
plot(fitGP)
plot(fitGP, "trace")
# you can see the difficulty in getting good numerics here.
# the warnings are not a coding problem, but challenges in
# the likelihood for the data.
```

```
# PP model.
fitPP <- fevd(Prec, Denversp, threshold=0.5, type="PP", units="mm",
    time.units="744/year", verbose=TRUE)

fitPP
plot(fitPP)
plot(fitPP, "trace")

fitPP <- fevd(Prec, Denversp, threshold=0.5, type="PP", optim.args=list(method="Nelder-Mead"),
    time.units="744/year", units="mm", verbose=TRUE)
fitPP
plot(fitPP) # Much better.
plot(fitPP, "trace")
# Better than above, but can see the difficulty!
# Can see the importance of good starting values!

# Try out for small samples
# Using one of the data example from Martins and Stedinger (2000)
z <- c( -0.3955, -0.3948, -0.3913, -0.3161, -0.1657, 0.3129, 0.3386, 0.5979,
    1.4713, 1.8779, 1.9742, 2.0540, 2.6206, 4.9880, 10.3371 )

tmpML <- fevd( z ) # Usual MLE.

# Find 0.999 quantile for the MLE fit.
# "True" 0.999 quantile is around 11.79
p <- tmpML$results$par
qevd( 0.999, loc = p[ 1 ], scale = p[ 2 ], shape = p[ 3 ] )

tmpLM <- fevd(z, method="Lmoments")
p <- tmpLM$results
qevd( 0.999, loc = p[ 1 ], scale = p[ 2 ], shape = p[ 3 ] )

tmpGML <- fevd(z, method="GMLE")
p <- tmpGML$results$par
qevd( 0.999, loc = p[ 1 ], scale = p[ 2 ], shape = p[ 3 ] )

plot(tmpLM)
dev.new()
plot(tmpGML)

# Bayesian
fitB <- fevd(Prec, ftcanmax, method="Bayesian", verbose=TRUE)
fitB
plot(fitB)
plot(fitB, "trace")

# Above looks good for scale and shape, but location does not appear to have found its way.
fitB <- fevd(Prec, ftcanmax, method="Bayesian", priorParams=list(v=c(1, 10, 10)), verbose=TRUE)
fitB
plot(fitB)
plot(fitB, "trace")

# Better, but what if we start with poor initial values?
```

```
fitB <- fevd(Prec, ftcanmax, method="Bayesian", priorParams=list(v=c(0.1, 10, 0.1)),
    initial=list(location=0, scale=0.1, shape=-0.5)), verbose=TRUE)
fitB
plot(fitB)
plot(fitB, "trace")

##
## Non-constant threshold.
##
data(Tphap)

# Negative of minimum temperatures.
plot(-Tphap$MinT)

fitGP2 <- fevd(-MinT ~1, Tphap, threshold=c(-70,-7), threshold.fun=~I((Year - 48)/42), type="GP",
    time.units="62/year", verbose=TRUE)
fitGP2
plot(fitGP2)
plot(fitGP2, "trace")
par(mfrow=c(1,1))
plot(fitGP2, "hist")
plot(fitGP2, "rl")

ci(fitGP2, type="parameter")

##
## Non-stationary models.
##

data(PORTw)

# GEV
fitPORTstdmax <- fevd(PORTw$TMX1, PORTw, scale.fun=~STDTMAX, use.phi=TRUE)
plot(fitPORTstdmax)
plot(fitPORTstdmax, "trace")
# One can see how finding the optimum value numerically can be tricky!

# Bayesian
fitPORTstdmaxB <- fevd(PORTw$TMX1, PORTw, scale.fun=~STDTMAX, use.phi=TRUE,
    method="Bayesian", verbose=TRUE)
fitPORTstdmaxB
plot(fitPORTstdmaxB)
plot(fitPORTstdmaxB, "trace")

# Let us go crazy.
fitCrazy <- fevd(PORTw$TMX1, PORTw, location.fun=~AOindex + STDTMAX, scale.fun=~STDTMAX,
    shape.fun=~STDMIN, use.phi=TRUE)
fitCrazy
plot(fitCrazy)
plot(fitCrazy, "trace")
# With so many parameters, you may need to stretch the device
# using your mouse in order to see them well.
```

```
ci(fitCrazy, type="parameter", which=2) # Hmmm.  NA NA.  Not good.
ci(fitCrazy, type="parameter", which=2, method="proflik", verbose=TRUE)
# Above not quite good enough (try to get better bounds).

ci(fitCrazy, type="parameter", which=2, method="proflik", xrange=c(0, 2), verbose=TRUE)
# Much better.


##
## Center and scale covariate.
##
data(Fort)

fitGPcross <- fevd(Prec, Fort, threshold=0.395,
    scale.fun=~cos(day/365.25) + sin(day/365.25) + I((year - 1900)/99),
    type="GP", use.phi=TRUE, units="inches")

fitGPcross
plot(fitGPcross) # looks good!

# Get a closer look at the effective return levels.
par(mfrow=c(1,1))
plot(fitGPcross, "rl", xlim=c(10000,12000))

lr.test(fitGPfc, fitGPcross)


## End(Not run)
```

`findAllMCMCpars` *Manipulate MCMC Output from fevd Objects*

Description

Manipulates the MCMC sample from an "fevd" object to be in a unified format that can be used in other function calls.

Usage

```
findAllMCMCpars(x, burn.in = 499, qcov = NULL, ...)
```

Arguments

`x`	Object of class "fevd" with component `method` = "Bayesian".
`burn.in`	Burn in period.
`qcov`	Matrix giving specific covariate values. See 'make.qcov' for more details. If not suplied, original covariates are used.
`...`	Not used.

Details

This function was first constructed for use by `postmode`, but might be useful in other areas as well. It evaluates any parameters that vary according to covariates at the values supplied by `qcov` or else at the covariate values used to obtain the original fit (default). If a model does not contain one or more parameters (e.g., the GP does not have a location component), then a column with these values (set to zero) are returned. That is, a matrix with columns corresponding to location, scale, shape and threshold are returned regardless of the model fit so that subsequent calls to functions like `fevd` can be made more easily.

This function is intended more as an internal function, but may still be useful to end users.

This function is very similar to `findpars`, but is only for MCMC samples and returns the entire MCMC sample of parameters. Also, returns a matrix instead of a list.

Value

A matrix is returned whose rows correspond to the MCMC samples (less burn in), and whose columns are "location" (if no location parameter is in the model, this column is still given with all values identical to zero), "scale", "shape" and "threshold".

Author(s)

Eric Gilleland

See Also

`fevd`, `findpars`, `postmode`

`findpars` *Get EVD Parameters*

Description

Obtain the parameters from an fevd object. This function differs greatly from distill.

Usage

```
findpars(x, ...)

## S3 method for class 'fevd'
findpars(x, ...)

## S3 method for class 'fevd.bayesian'
findpars(x, burn.in = 499, FUN = "mean",
    use.blocks = FALSE, ..., qcov = NULL)

## S3 method for class 'fevd.lmoments'
findpars(x, ...)
```

```
## S3 method for class 'fevd.mle'
findpars(x, use.blocks = FALSE, ..., qcov = NULL)
```

Arguments

x	A list object of class "fevd" as returned by fevd.
burn.in	number giving the burn in value. The first 1:burn.in will not be used in obtaining parmaeter estiamtes.
FUN	character string naming a function, or a function, to use to find the parameter estimates from the MCMC sample. Default is to take the posterior mean (after burn in).
use.blocks	logical: If TRUE and x was fit with blocks provided, returns parameters for each block
...	Not used.
qcov	numeric matrix with rows the same length as q and columns equal to the number of parameters (+ 1 for the threshold, if a POT model). This gives any covariate values for a nonstationary model. If NULL, and model is non-stationary, only the intercept terms for modeled parameters are used, and if a non-constant threshold, only the first threshold value is used. Not used if model is stationary.

Details

This function finds the EVD parameters for each value of the covariates in a non-stationary model. In the case of a stationary model, it will return vectors of length equal to the length of the data that simply repeat the parameter(s) value(s).

Note that this differs greatly from distill, which simply returns a vector of the length of the number of parameters in the model. This function returns a named list containing the EVD parameter values possibly for each value of the covariates used to fit the model. For example, if a GEV(location(t), scale, shape) is fit with location(t) = mu0 + mu1 * t, say, then the "location" component of the returned list will have a vector of mu0 + mu1 * t for each value of t used in the model fit.

Value

A list object is returned with components

location, scale, shape
: vector of parameter values (or NULL if the parameter is not in the model). For stationary models, or for parameters that are fixed in the otherwise non-stationary model, the vectors will repeat the parameter value. The length of the vectors equals the length of the data used to fit the models.

Author(s)

Eric Gilleland

See Also

fevd, distillery::distill, parcov.fevd

Examples

```
z <- revd(100, loc=20, scale=0.5, shape=-0.2)
fit <- fevd(z)
fit

findpars(fit)

## Not run:
data(PORTw)
fit <- fevd(TMX1, PORTw, location.fun=~AOindex, units="deg C")
fit

findpars(fit)


## End(Not run)
```

Flood *United States Total Economic Damage Resulting from Floods*

Description

United States total economic damage (in billions of U.S. dollars) caused by floods by hydrologic year from 1932-1997. See Pielke and Downton (2000) for more information.

Usage

```
data(Flood)
```

Format

A data frame with 66 observations on the following 5 variables.

OBS a numeric vector giving the line number.

HYEAR a numeric vector giving the hydrologic year.

USDMG a numeric vector giving total economic damage (in billions of U.S. dollars) caused by floods.

DMGPC a numeric vector giving damage per capita.

LOSSPW a numeric vector giving damage per unit wealth.

Details

From Pielke and Downton (2000):

The National Weather Service (NWS) maintains a national flood damage record from 1903 to the present, and state level data from 1983 to the present. The reported losses are for "significant flood events" and include only direct economic damage that results from flooding caused by ranfall and/or snowmelt. The annual losses are based on "hydrologic years" from October through September. Flood damage per capita is computed by dividing the inflation-adjusted losses for each hydrological year by the estimated population on 1 July of that year (www.census.gov). Flood damage per million dollars of national wealth uses the net stock of fixed reproducible tangible wealth in millions of current dollars (see Pielke and Downton (2000) for more details; see also Katz et al. (2002) for analysis).

Source

National Weather Service

References

Katz, R. W., Parlange, M. B. and Naveau, P. (2002) Statistics of extremes in hydrology, *Advances in Water Resources*, **25**, 1287–1304.

Pielke, R. A. Jr. and Downton, M. W. (2000) Precipitation and damaging floods: trends in the United States, 1932-97, *Journal of Climate*, **13**, (20), 3625–3637.

Examples

```
data(Flood)
plot( Flood[,2], Flood[,3], type="l", lwd=2, xlab="hydrologic year",
    ylab="Total economic damage (billions of U.S. dollars)")
```

Fort *Daily precipitation amounts in Fort Collins, Colorado.*

Description

Daily precipitation amounts (inches) from a single rain gauge in Fort Collins, Colorado. See Katz et al. (2002) Sec. 2.3.1 for more information and analyses.

Usage

```
data(Fort)
```

Format

A data frame with dimension 36524 by 5. Columns are: "obs", "tobs", "month", "day", "year" and "Prec"; where "Prec" is the daily precipitation amount (inches).

Source

Originally from the Colorado Climate Center at Colorado State University. The Colorado state climatologist office no longer provides this data without charge. The data can be obtained from the NOAA/NCDC web site, but there are slight differences (i.e., some missing values for temperature).

References

Katz, R. W., Parlange, M. B. and Naveau, P. (2002) Statistics of extremes in hydrology. *Advances in Water Resources*, **25**, 1287–1304.

Examples

```
data(Fort)
str(Fort)
plot(Fort[,"month"], Fort[,"Prec"], xlab="month", ylab="daily precipitation (inches)")
```

---

fpois — *Fit Homogeneous Poisson to Data and Test Equality of Mean and Variance*

Description

Fit a homogeneous Poisson to data and test whether or not the mean and variance are equal.

Usage

```
fpois(x, na.action = na.fail, ...)

## Default S3 method:
fpois(x, na.action = na.fail, ...)

## S3 method for class 'data.frame'
fpois(x, na.action = na.fail, ..., which.col = 1)

## S3 method for class 'matrix'
fpois(x, na.action = na.fail, ..., which.col = 1)

## S3 method for class 'list'
fpois(x, na.action = na.fail, ..., which.component = 1)
```

Arguments

x	numeric, matrix, data frame or list object containing the data to which the Poisson is to be fit.
na.action	function to be called to handle missing values.

`...` Not used.

`which.col`, `which.component`
column or component (list) number containing the data to which the Poisson is to be fit.

Details

The probability function for the (homogeneous) Poisson distribution is given by:

Pr(N = k) = exp(-lambda) * lambda^k / k!

for k = 0, 1, 2, ...

The rate parameter, lambda, is both the mean and the variance of the Poisson distribution. To test the adequacy of the Poisson fit, therefore, it makes sense to test whether or not the mean equals the variance. R. A. Fisher showed that under the assumption that X_1, ..., X_n follow a Poisson distribution, the statistic given by:

D = (n - 1) * var(X_1) / mean(X_1)

follows a Chi-square distribution with n - 1 degrees of freedom. Therefore, the p-value for the one-sided alternative (greater) is obtained by finding the probability of being greater than D based on a Chi-square distribution with n - 1 degrees of freedom.

Value

A list of class "htest" is returned with components:

`statistic`	The value of the dispersion D
`parameter`	named numeric vector giving the estimated mean, variance, and degrees of freedom.
`alternative`	character string with the value "greater" indicating the one-sided alternative hypothesis.
`p.value`	the p-value for the test.
`method`	character string stating the name of the test.
`data.name`	character naming the data used by the test (if a vector is applied).

Author(s)

Eric Gilleland

See Also

`glm`

Examples

```
data(Rsum)
fpois(Rsum$Ct)

## Not run:
```

```
# Because 'Rsum' is a data frame object,
# the above can also be carried out as:

fpois(Rsum, which.col = 3)

# Or:

fpois(Rsum, which.col = "Ct")

##
## For a non-homogeneous fit, use glm.
##
## For example, to fit the non-homogeneous Poisson model
## Enso as a covariate:
##

fit <- glm(Ct~EN, data = Rsum, family = poisson())
summary(fit)

## End(Not run)
```

`ftcanmax` *Annual Maximum Precipitation: Fort Collins, Colorado*

Description

Annual maximum precipitation (inches) for one rain gauge in Fort Collins, Colorado from 1900 through 1999. See Katz et al. (2002) Sec. 2.3.1 for more information and analyses.

Usage

```
data(ftcanmax)
```

Format

A data frame with 100 observations on the following 2 variables.

Year a numeric vector giving the Year.

Prec a numeric vector giving the annual maximum precipitation amount in inches.

Source

Originally from the Colorado Climate Center at Colorado State University. The Colorado state climatologist office no longer provides this data without charge. The data can be obtained from the NOAA/NCDC web site, but there are slight differences (i.e., some missing values for temperature). The annual maximum precipitation data is taken directly from the daily precipitation data available in this package under the name "Fort".

References

Katz, R. W., Parlange, M. B. and Naveau, P. (2002) Statistics of extremes in hydrology. *Advances in Water Resources*, **25**, 1287–1304.

Examples

```
data(ftcanmax)
str(ftcanmax)
plot(ftcanmax, type="l", lwd=2)
```

HEAT *Summer Maximum and Minimum Temperature: Phoenix, Arizona*

Description

Summer maximum and minimum temperature (degrees Fahrenheit) for July through August 1948 through 1990 at Sky Harbor airport in Phoenix, Arizona.

Usage

```
data(HEAT)
```

Format

A data frame with 43 observations on the following 3 variables.

Year a numeric vector giving the number of years since 1900.

Tmax a numeric vector giving the Summer maximum temperatures in degrees Fahrenheit.

Tmin a numeric vector giving the Summer minimum temperatures in degrees Fahrenheit.

Details

Data is Summer maximum and minimum temperature for the months of July through August from 1948 through 1990.

Source

U.S. National Weather Service Forecast office at the Phoenix Sky Harbor Airport.

References

Balling, R. C., Jr., Skindlov, J. A. and Phillips, D. H. (1990) The impact of increasing summer mean temperatures on extreme maximum and minimum temperatures in Phoenix, Arizona. *Journal of Climate*, **3**, 1491–1494.

Tarleton, L. F. and Katz, R. W. (1995) Statistical explanation for trends in extreme summer temperatures at Phoenix, A.Z. *Journal of Climate*, **8**, (6), 1704–1708.

Examples

```
data(HEAT)
str(HEAT)
plot(HEAT)
```

`hwmi` *Heat Wave Magnitude Index*

Description

This function computes the Heat Wave Magnitude Index and the associated duration and starting date of a heat wave event.

Usage

```
hwmi(yTref, Tref, yTemp, Temp)
```

Arguments

`yTref`	a single numeric value giving the starting year of Tref
`Tref`	a numeric vector of daily maximum temperatures for a 32-year reference period used to calculate threshold and empirical cumulative distribution function (ecdf).
`yTemp`	a single numeric value giving the starting year of Temp
`Temp`	a numeric vector of daily maximum temperature for at least one year (with length not shorter than 365 days) or n-years (each with length not shorter than 365 days) containing the data to which the HWMI is to be calculated.

Details

This function takes a daily maximum temperature time series as input and computes the climate index HWMI (Heat Wave Magnitude Index). The Heat Wave Magnitude Index is defined as the maximum magnitude of the heat waves in a year. A "heat wave" is defined as a sequence of 3 or more days in which the daily maximum temperature is above the 90th percentile of daily maximum temperature for a 31-day running window surrounding this day during the baseline period (e.g., 1981-2010).

Note that the argument `Tref` must have daily maximum temperatures for a 32-year period (e.g., using the baseline period of 1981 to 2010, `Tref` must be for 1980 through 2011). The first and the 32nd years are needed to calculate the daily threshold over the first and the last year of the baseline period (1981-2010).

If HWMI is calculated in the Southern Hemisphere, then, in order not to split a Heat Wave event into two, the year should start on the 1st of July and end on the 30th of June of the following year.

In other words:

1. Tref will be from the 1st of July 1980 up to the 30th of June 2012

2. Similarly for Temp, each 365 days in a year must be taken between the 1st of July and the 30th of June.

Value

A list with the following components: hwmi: a numeric vector containing the hwmi value for each year and the associated duration and starting day (number between 1 and 365) for each heat wave event in a year. thr: a numeric vector containing 365 temperature values representing the daily threshold for the reference period (1981-2010 or any other 30 years period). pdfx: the "n" (n was fixed to 512) coordinates of the points (sum of three daily maximum temperatures) where the density is estimated. pdfy: the estimated density values. These will be non-negative, but can be zero.

Author(s)

Simone Russo <simone.russo@jrc.ec.europa.eu>

References

Russo, S. and Coauthors, 2014. Magnitude of extreme heat waves in present climate and their projection in a warming world. *J. Geophys. Res.*, doi:10.1002/2014JD022098.

Examples

```
data("CarcasonneHeat")

tiid <- CarcasonneHeat[2,]

jan1980 <- which(tiid == 19800101)
jan2003 <- which(tiid == 20030101)
dec2003 <- which(tiid == 20031231)
dec2011 <- which(tiid == 20111231)

Temp <- CarcasonneHeat[3, jan2003:dec2003] / 10
Tref <- CarcasonneHeat[3, jan1980:dec2011] / 10

##hwmi calculation
hwmiFr2003 <- hwmi(1980, Tref, 2003, Temp)

#### Heat Wave occurred in Carcassonne, France, 2003

plot(c(150:270), Temp[150:270], xlim = c(150, 270),
    ylim = c((min(hwmiFr2003$thr[150:270]) -
        sd(hwmiFr2003$thr[150:270])), max(Temp[150:270])),
    xlab = "days", ylab = "temperature", col = 8)

par(new = TRUE)
plot(c(150:270), hwmiFr2003$thr[150:270], type = "l",
    xlim = c(150,270),
    ylim = c((min(hwmiFr2003$thr[150:270]) - sd(hwmiFr2003$thr[150:270])),
    max(Temp)), xlab = "", ylab = "", col = 1, lwd = 2)

par(new = TRUE)
plot(c(hwmiFr2003$hwmi[1,3]:(hwmiFr2003$hwmi[1,3] + hwmiFr2003$hwmi[1,2]-1)),
    Temp[hwmiFr2003$hwmi[1,3]:(hwmiFr2003$hwmi[1,3] + hwmiFr2003$hwmi[1,2] - 1)],
```

```
    xlim = c(150,270),
    ylim = c((min(hwmiFr2003$thr[150:270]) - sd(hwmiFr2003$thr[150:270])),
        max(Temp[150:270])),
    xlab = "", ylab = "", col = 4, type = "b",
    main = "Carcassonne, France, 2003", lwd = 2)

text(175, 42, "hwmi = 3.68", col = 4, font = 2)
text(175, 41, "Duration = 12 days", col = 4, font = 2)
text(175, 40, "Starting day = 214 (02.Aug.2003)", col = 4, font = 2)
```

`hwmid` *Heat Wave Magnitude Index*

Description

This function computes the Heat Wave Magnitude Index and the associated duration and starting date of a heat wave event.

Usage

```
hwmid(yTref, Tref, yTemp, Temp)
```

Arguments

`yTref`	a single numeric value giving the starting year of Tref
`Tref`	a numeric vector of daily maximum temperatures for a 32-year reference period used to calculate threshold, T30ymax and T30ymin as defined below.
`yTemp`	a single numeric value giving the starting year of Temp
`Temp`	a numeric vector of daily maximum (minimum or mean) temperature for at least one year (with length not shorter than 365 days) or n-years (each with length not shorter than 365 days) containing the data to which the HWMId is to be calculated.

Details

This function takes a daily temperature time series as input and computes the climate index HWMId (Heat Wave Magnitude Index daily). The Heat Wave Magnitude Index daily is defined as the maximum magnitude of the heat waves in a year. A "heat wave" is defined as a sequence of 3 or more days in which the daily maximum temperature is above the 90th percentile of daily maximum temperature for a 31-day running window surrounding this day during the baseline period (e.g., 1981-2010).

Note that the argument `Tref` must have daily temperatures for a 32-year period (e.g., using the baseline period of 1981 to 2010, `Tref` must be for 1980 through 2011). The first and the 32nd years are needed to calculate the daily threshold over the first and the last year of the baseline period (1981-2010).

Value

A list with the following components: hwmid: a numeric vector containing the hwmid value for each year and the associated duration and starting day (number between 1 and 365) for each heat wave event in a year. thr: a numeric vector containing 365 temperature values representing the daily threshold for the reference period (1981-2010 or any other 30 years period). T30y75p: a single numeric value giving the 75th percentile value of the time series calculated from Tref and composed of 30-year annual maximum temperatures of the baseline period. T30y25p: a single numeric value giving the 25th percentile value of the time series calculated from Tref and composed of 30-year annual maximum temperatures of the baseline period.

Author(s)

Simone Russo <simone.russo@jrc.ec.europa.eu, simone.russo@isprambiente.it>

References

Russo, S., J. Sillmann, E. Fischer, 2015. Top ten European heatwaves since 1950 and their occurrence in the coming decades. *Environmental Research Letters*, **10**, 124003, doi:10.1088/1748-9326/10/12/124003.

Russo, S. and Coauthors, 2014. Magnitude of extreme heat waves in present climate and their projection in a warming world. *J. Geophys. Res.*, doi:10.1002/2014JD022098.

Examples

```
data("CarcasonneHeat")

tiid <- CarcasonneHeat[2,]

jan1980 <- which(tiid == 19800101)
jan2003 <- which(tiid == 20030101)
dec2003 <- which(tiid == 20031231)
dec2011 <- which(tiid == 20111231)

Temp <- CarcasonneHeat[3, jan2003:dec2003] / 10
Tref <- CarcasonneHeat[3, jan1980:dec2011] / 10


##hwmid calculation
hwmidFr2003 <- hwmid(1980, Tref, 2003, Temp)
hwmiFr2003 <- hwmi(1980, Tref, 2003, Temp)

T30y25p <- hwmidFr2003$T30y25p
T30y75p <- hwmidFr2003$T30y75p
range30y <- (T30y75p - T30y25p)

#daymag<-(Temp[214:225]-hwmidFr2003$T30ymin)/(hwmidFr2003$T30ymax-hwmidFr2003$T30ymin)
#### Heat Wave occurred in Carcassonne, France, 2003

split.screen( rbind( c(0, 1, 0.6, 1), c(0, 0.5, 0, 0.6), c(0.5, 1, 0, 0.6) ) )
screen(1)
```

```
par( mar = c(2, 2, 2, 0) )
plot( c(1:365), Temp[1:365], xlim = c(190, 240), ylim = c(25, 50),
    xlab = "", ylab = "", cex.axis = 1.1, col = 8, font.axis = 2)

par( new = TRUE )
plot( c(150:270), hwmiFr2003$thr[150:270], type = "l",xlim = c(190, 240),
    ylim = c(25, 50), xlab = "", ylab = "", col = 1, lwd = 2, axes = FALSE)

par(new = TRUE)


plot(c(214:216), Temp[214:216], xlim = c(190, 240), ylim = c(25, 50),
    xlab = "", ylab = "", col = 4, type = "b", lwd = 2, axes = FALSE,
    pch = "a", cex = 1.2)

par( new = TRUE)
plot(c(217:219), Temp[217:219], xlim = c(190, 240), ylim = c(25,50),
    xlab = "", ylab = "", col = 4, type = "b", lwd = 2, axes = FALSE,
    pch = "b", cex = 1.2)

par(new=TRUE)
plot(c(220:222), Temp[220:222], xlim = c(190, 240), ylim = c(25, 50),
    xlab = "", ylab = "", col = 4, type = "b", lwd = 2, axes = FALSE,
    pch = "c", cex = 1.2)

par(new=TRUE)
plot(c(223:225), Temp[223:225], xlim = c(190, 240), ylim = c(25, 50),
    xlab = "", ylab = "", col = 4, type = "b", lwd = 2, axes = FALSE,
    pch = "d", cex = 1.2)


par(new=TRUE)
plot(c(214:216), (Temp[214:216]+5), xlim = c(190, 240), ylim = c(25, 50),
    xlab = "", ylab = "", col = 3, type = "b", lwd = 2, axes = FALSE,
    pch = "a", cex = 1.2)

par(new=TRUE)
plot(c(217:219), (Temp[217:219]+5), xlim = c(190, 240), ylim = c(25, 50),
    xlab = "", ylab = "", col = 3, type = "b", lwd = 2, axes = FALSE,
    pch = "b", cex = 1.2)

par(new=TRUE)
plot(c(220:222), (Temp[220:222]+5), xlim = c(190, 240), ylim = c(25, 50),
    xlab = "", ylab = "", col = 3, type = "b", lwd = 2, axes = FALSE,
    pch = "c", cex = 1.2)

par(new=TRUE)
plot(c(223:225), (Temp[223:225]+5), xlim = c(190, 240), ylim = c(25, 50),
    xlab = "", ylab = "", col = 3, type = "b", lwd = 2, axes = FALSE,
    pch = "d", cex = 1.2)

text(200, 50, "HW2", col = 3, font = 2)
text(200, 48, "HWMI=4", col = 3, font = 2)
```

```
text(200, 46, "HWMId = 41.9",col = 3, font = 2)

text(200, 42, "HW1", col = 4, font = 2)
text(200, 40, "HWMI = 3.68", col = 4, font = 2)
text(200, 38, "HWMId=18.6",col = 4, font = 2)


box()
```

`is.fixedfevd` *Stationary Fitted Model Check*

Description

Test if a fitted `fevd` object is stationary or not.

Usage

```
is.fixedfevd(x)

check.constant(x)
```

Arguments

`x`	A list object of class "fevd" as returned by `fevd`. For `check.constant`, this may be a formula or vector.

Details

This function is mostly intended as an internal function, but it may be useful generally.

`check.constant` determines if a formula is given simply by ~ 1. It is used by `is.fixedfevd`.

Value

logical of length one stating whether the fitted model is stationary (TRUE) or not (FALSE).

Author(s)

Eric Gilleland

See Also

`fevd`

Examples

```
z <- revd(100, loc=20, scale=0.5, shape=-0.2)
fit <- fevd(z)
fit

is.fixedfevd(fit)
```

`levd` *Extreme Value Likelihood*

Description

Find the EVD parameter likelihood given data.

Usage

```
levd(x, threshold, location, scale, shape,
    type = c("GEV", "GP", "PP", "Gumbel", "Weibull", "Frechet",
    "Exponential", "Beta", "Pareto"), log = TRUE, negative = TRUE,
    span, npy = 365.25, infval = Inf, weights = 1, blocks = NULL)
```

Arguments

`x`	A numeric vector of data of length n.
`threshold`	number or numeric vector of length n giving the desired threshold, if applicable.
`location`	number or numeric vector of length n giving the location parameter value(s), if applicable.
`scale`	number or numeric vector of length n giving the scale parameter value(s).
`shape`	number or numeric vector of length n giving the shape parameter value(s), if applicable.
`type`	character string naming the particular EVD for which to compute the likelihood.
`log, negative`	logicals; should the negative log-likelihood (default) be returned (both TRUE) or the likelihood (both FALSE)? It is possible to return other possibilities such as the negative likelihood (log = FALSE, negative = TRUE) or the log-likelihood (log = TRUE, negative = FALSE).
`span`	number stating how many periods (usually years) the data cover (applicable only for PP models). Currently not used.
`npy`	number of points per period (period is usually years).
`infval`	Value to return if the likelihood is infinite. If negative is FALSE, the negative of this value will be returned. The default is to return `Inf`, but noe that for optimization routines, this would result in an error and stop the process. Therefore, it can be advantageous to use a very large value instead.
`weights`	numeric of length 1 or n giving weights to be applied in the likelihood calculation (e.g., if some data points are to be weighted more/less heavily than others).

`blocks` An optional list containing information required to evaluate the likelihood of point process models in a computationally-efficient manner by using only the exceedances and not the observations below the threshold(s). See details.

Details

This function is called by a wrapper function within `fevd` and other functions. It is generally an internal function, but may be useful for some users.

The negative log-likelihood for the generalized extreme value (GEV) df, possibly with parameters that are functions of covariates, yi) is given by:

sum(log(scale(yi))) + sum(z^(-1/shape(yi)) + sum(log(z) * (1/shape(yi) + 1)),

where z = (x - location(yi))/scale(yi), x are the data. For the Frechet and Weibull cases, the shape parameter is forced to have the correct sign, so it does not matter if the user chooses positive or negative shape. In the case of shape = 0, defined by continuity (Gumbel case), the negative log-likelihood simplifies to:

sum(log(scale(yi))) + sum(z) + sum(exp(-z)),

where z is as above.

The negative log-likelihood for the GP df is given by:

sum(log(scale(yi))) + sum(log(z) * (1/shape(yi) + 1)),

where z = 1 + shape(yi) * t, where t = (x[x > threshold] - threshold(yi))/scale(yi). Similar to the GEV df, the Beta and Pareto cases are forced to have the correct sign for the shape parameter. In the case of shape = 0, defined by continuity (Exponential case), the negative log-likelihood simplifies to:

sum(log(scale(yi))) + z,

where z is as above in the GP negative log-likelihood.

See Coles (2001) for more details.

Using Blocks to Reduce Computation in PP Fitting:

When `blocks` is supplied, the user should provide only the exceedances and not all of the data values. The list should contain a component called `nBlocks` indicating the number of observations within a block, where blocks are defined in a manner analogous to that used in GEV models. The list should also contain components named `threshold`, `location`, `scale`, `shape`, and `weights` corresponding to the arguments of the same name supplied to `levd`, but with values on a per block basis. If some of the observations within any block are missing (assuming missing at random or missing completely at random), the list should contain a `proportionMissing` component that is a vector with one value per block indicating the proportion of observations missing for the block. Scalar values are allowed when a component is stationary. Warning: to properly analyze nonstationary models, the components must be constant within each block.

Value

A single number giving the likelihood value (or negative log-likelihood or log-likelihood or negative likelihood depending on the value of the log and negative arguments).

Author(s)

Eric Gilleland

References

Coles, S. (2001) *An introduction to statistical modeling of extreme values*, London, U.K.: Springer-Verlag, 208 pp.

See Also

fevd, profliker

Examples

```
data(ftcanmax)

levd(ftcanmax$Prec, location=134.66520, scale=53.28089, shape=0.17363)
```

---

lr.test *Likelihood-Ratio Test*

Description

Conduct the likelihood-ratio test for two nested extreme value distribution models.

Usage

```
lr.test(x, y, alpha = 0.05, df = 1, ...)
```

Arguments

x, y	Each can be either an object of class "fevd" (provided the fit method is MLE or GMLE) or a single numeric giving the negative log-likelihod value for each model. x should be the model with fewer parameters, but if both x and y are "fevd" objects, then the order does not matter (it will be determined from which model has more parameters).
alpha	single numeric between 0 and 1 giving the significance level for the test.
df	single numeric giving the degrees of freedom. If both x and y are "fevd" objects, then the degrees of freedom will be calculated, and this argument ignored. Otherwise, if either or both of x and y are single numerics, then it must be provided or the test may be invalid.
...	Not used.

Details

When it is desired to incorporate covariates into an extreme value analysis, one method is to incorporate them into the parameters of the extreme value distributions themselves in a regression-like manner (cf. Coles, 2001 ch 6; Reiss and Thomas, 2007 ch 15). In order to justify whether or not inclusion of the covariates into the model is significant or not is to apply the likelihood-ratio test (of course, the test is more general than that, cf. Coles (2001) p 35).

The test is only valid for comparing nested models. That is, the parameters of one model must be a subset of the parameters of the second model.

Suppose the base model, m0, is nested within the model m1. Let `x` be the negative log-likelihood for m0 and `y` for m1. Then the likelihood-ratio statistic (or deviance statistic) is given by (Coles, 2001, p 35; Reiss and Thomas, 2007, p 118):

D = -2*(y - x).

Letting c.alpha be the (1 - alpha) quantile of the chi-square distribution with degrees of freedom equal to the difference in the number of model parameters, the null hypothesis that D = 0 is rejected if D > c.alpha (i.e., in favor of model m1).

Value

A list object of class "htest" is returned with components:

`statistic`	The test statistic value (referred to as D above).
`parameter`	numeric vector giving the chi-square critical value (c.alpha described above), the significance leve (alpha) and the degrees of freedom.
`alternative`	character string stating "greater" indicating that the alternative decision is determined if the statistic is greater than c.alpha.
`p.value`	numeric giving the p-value for the test. If the p-value is smaller than alpha, then the decision is to reject the null hypothesis in favor of the model with more parameters.
`method`	character string saying "Likelihood-ratio Test".
`data.name`	character vector of length two giving the names of the datasets used for the test (if "fevd" objects are passed) or the negative log-likelihood values if numbers are passed, or the names of x and y. Although the names may differ, the models should have been fit to the same data set.

Author(s)

Eric Gilleland

References

Coles, S. (2001) *An introduction to statistical modeling of extreme values*, London, U.K.: Springer-Verlag, 208 pp.

Reiss, R.-D. and Thomas, M. (2007) *Statistical Analysis of Extreme Values: with applications to insurance, finance, hydrology and other fields*. Birkhauser, 530pp., 3rd edition.

See Also

fevd, taildep.test

Examples

```
data(PORTw)
fit0 <- fevd(PORTw$TMX1, type="Gumbel")
fit1 <- fevd(PORTw$TMX1)
fit2 <- fevd(TMX1, PORTw, scale.fun=~STDTMAX)
lr.test(fit0, fit1)
lr.test(fit1, fit2)
```

---

make.qcov *Covariate Matrix for Non-Stationary EVD Projections*

Description

Create a matrix for use with pextRemes.

Usage

```
make.qcov(x, vals, nr = 1, ...)

is.qcov(x)
```

Arguments

x	make.qcov: A list object of class "fevd" as output from fevd. is.qcov: Any R object.
vals	Either a named list whose names match the fitted model parameter names, or may be "threshold", a matrix or a numeric vector of length equal to the size of the resulting matrix.
nr	The number of rows desired in the resulting matrix. Only if vals is a vector. If vals argument is not a vector, the code will either fail or the argument will be ignored.
...	optional arguments to matrix (e.g., byrow=TRUE, depending on the order for vals, if a vector). Only used if vals is a vector.

Details

Simply sets up a matrix of parameter coefficients to be used by pextRemes. In particular, all parameters/thresholds that are constant (i.e., do not depend on covariate values) should have columns of all ones. Paramters/threshold that vary in a non-stationary model may have whatever values are of interest.

`is.qcov` performs some very simple tests to determine if an object is a proper `qcov` matrix or not. It is possible to have a matrix that is not a proper `qcov` matrix, but the returned value is TRUE. It is also possible to have a valid `qcov` object that id not appropriate for a particular model. Mostly this is an internal function.

Value

An nr by np + 1 matrix is returned, where np is the number of parameters in the model. The last column is always "threshold" even if the model does not take a threshold (e.g., the GEV df), in which case the last column may be all NA, 0, or some other value depending on the vals argument.

Author(s)

Eric Gilleland

See Also

`pextRemes`, `fevd`, `erlevd`

Examples

```
data(PORTw)
fit <- fevd(TMX1, PORTw, location.fun=~AOindex, units="deg C")
fit

v <- cbind(rep(1,4), c(1, -1, 1, -1), rep(1,4), rep(1,4))
v <- make.qcov(fit, vals=v, nr=4)
v

# cf.
v <- make.qcov(fit, vals=list(mu1=c(1, -1, 1, -1)))
v

# Or
v <- make.qcov(fit, vals=c(rep(1,4), c(1, -1, 1, -1), rep(1,8), rep(0,4)), nr=4)
v
```

---

mrlplot *Mean Residual Life Plot*

Description

An empirical mean residual life plot, including confidence intervals, is produced. The mean residual life plot aids the selection of a threshold for the GPD or point process models.

Usage

```
mrlplot(x, nint = 100, alpha = 0.05, na.action = na.fail, xlab = "Threshold", ...)
```

Arguments

x	numeric vector of data.
nint	Number of thresholds to use.
alpha	number giving the 1 - alpha confidence levels to use.
na.action	function to be called to handle missing values.
xlab	character string giving the abscissa label to use.
...	optional arguments to plot.

Details

The mean excesses are found for each value of a range of thresholds that cover the range of the data (less one at the high end). CIs are also shown based on the normal df for the mean excesses. The goal is to find the lowest threshold such that the graph is linear with increasing thresholds, within uncertainty.

See Coles (2001) sec. 4.3.1 for more information.

Value

A matrix with the mean excess values and their confidence bounds is returned invisibly.

Author(s)

Eric Gilleland

References

Coles, S. (2001). *An introduction to statistical modeling of extreme values*, London, United Kingdom: Springer-Verlag, 208 pp.

See Also

threshrange.plot

Examples

```
data(Fort)
mrlplot(Fort$Prec)
```

`Ozone4H` *Ground-Level Ozone Order Statistics.*

Description

Ground-level ozone order statistics from 1997 at 513 monitoring stations in the eastern United States.

Usage

```
data(Ozone4H)
```

Format

A data frame with 513 observations on the following 5 variables.

station a numeric vector identifying the station (or line) number.

r1 a numeric vector giving the maximum ozone reading (ppb) for 1997.

r2 a numeric vector giving the second-highest ozone reading (ppb) for 1997.

r3 a numeric vector giving the third-highest ozone reading (ppb) for 1997.

r4 a numeric vector giving the fourth-highest ozone reading (ppb) for 1997.

Details

Ground level ozone readings in parts per billion (ppb) are recorded hourly at ozone monitoring stations throughout the country during the "ozone season" (roughly April to October). These data are taken from a dataset giving daily maximum 8-hour average ozone for 5 ozone seasons (including 1997). The new U.S. Environmental Protection Agency (EPA) National Ambient Air Quality Standard (NAAQS) for ground-level ozone is based on a three-year average of fourth-highest daily 8-hour maximum ozone readings.

For more analysis on the original data regarding the U.S. EPA NAAQS for ground-level ozone, see Fuentes (2003), Gilleland and Nychka (2005) and Gilleland et al. (2006). These data are in the form required by the `rlarg.fit` function of Stuart Coles available in the R package **ismev**; see Coles (2001) for more on the r-th largest order statistic model and the function `rlarg.fit`.

Source

Data was originally provided by the U.S. EPA

References

Coles, S. (2001) *An Introduction to Statistical Modeling of Extreme Values*. London, U.K.: Springer-Verlag, 208pp.

Fuentes, M. (2003) Statistical assessment of geographic areas of compliance with air quality. *Journal of Geophysical Research*, **108**, (D24).

Gilleland, E. and Nychka, D. (2005) Statistical Models for Monitoring and Regulating Ground-level Ozone. *Environmetrics*, **16**, 535–546.

Gilleland, E., Nychka, D., and Schneider, U. (2006) Spatial models for the distribution of extremes. In *Applications of Computational Statistics in the Environmental Sciences: Hierarchical Bayes and MCMC Methods*, Edited by J.S. Clark & A. Gelfand. Oxford University Press. 170–183, ISBN 0-19-8569671.

Examples

```
data(Ozone4H)
str(Ozone4H)
plot(Ozone4H)
```

`parcov.fevd` *EVD Parameter Covariance*

Description

Try to calculate the parameter covariance for an extreme value distribution (EVD) fitted using MLE.

Usage

```
parcov.fevd(x)
```

Arguments

`x`	A list object of class "fevd" as returned by `fevd`.

Details

Makes possibly two calls to `optimHess` in an effort to find the parameter covariance matrix for fitted EVDs where MLE is used. The first attempt uses the actual gradient of the negative log-likelihood. If this fails, or the Hessian matrix cannot be inverted, or there are any negative values along the diagonal in the inverted Hessian, then a second attempt is made using finite differences. See Coles (2001) sec. 2.6.4 for more details.

Value

An np by np matrix is returned where np is the number of parameters in the model.

Author(s)

Eric Gilleland

References

Coles, S. (2001) *An introduction to statistical modeling of extreme values*, London, U.K.: Springer-Verlag, 208 pp.

See Also

fevd, summary.fevd, print.fevd

Examples

```
z <- revd(100, loc=20, scale=0.5, shape=-0.2)
fit <- fevd(z)
fit

parcov.fevd(fit)
```

Peak *Salt River Peak Stream Flow*

Description

Peak stream flow data from 1924 through 1999 for the Salt River near Roosevelt, Arizona.

Usage

```
data(Peak)
```

Format

A data frame with 75 observations on the following 2 variables.

Year a numeric vector giving the year.

Flow a numeric vector giving the peak stream flow (cfs).

Winter a numeric vector giving the Winter seasonal mean Darwin pressure (mb–1000).

Spring a numeric vector giving the Spring seasonal mean Darwin pressure (mb–1000).

Summer a numeric vector giving the Summer seasonal mean Darwin pressure (mb–1000).

Fall a numeric vector giving the Fall seasonal mean Darwin pressure (mb–1000) (see Katz et al. (2002) Sec. 5.2.2).

Details

Peak stream flow in cfs (1 cfs=0.028317 m^3/s) data for water years (October through September) from 1924 through 1999 for the Salt River near Roosevelt, Arizona. Data for 1986 are missing. Also includes seasonal mean Darwin pressures (mb–1000).

Several analyses have been performed on streamflow at this location (see, e.g., Anderson and Meerschaert (1998), Dettinger and Diaz (2000); and, for extreme stream flow, Katz et al. (2002) Sec. 5.2.2).

Source

U.S. Geological Survey for Salt River peak flows. NOAA Climate Prediction Center for seasonal mean Darwin pressures.

References

Anderson, P. L. and Meerschaert, M. M. (1998) Modeling river flows with heavy tails. *Water Resour Res*, **34**, (9), 2271–2280.

Dettinger, M. D. and Diaz, H. F. (2000) Global characteristics of stream flow seasonality and variability. *Journal of Hydrometeorology*, **1**, 289–310.

Katz, R. W., Parlange, M. B. and Naveau, P. (2002) Statistics of extremes in hydrology. *Advances in Water Resources*, **25**, 1287–1304.

Examples

```
data(Peak)
str(Peak)
# Fig. 9 of Katz et al. (2002) Sec. 5.2.2.
plot(Peak[,"Year"], Peak[,"Flow"]/1000, type="l", yaxt="n",
    xlab="Water year (Oct-Sept)", ylab="Annual peak flow (thousand cfs)")
axis(2,at=c(0,40,80,120),labels=c("0","40","80","120"))
```

---

pextRemes *Probabilities and Random Draws from Fitted EVDs*

Description

Calculate probabilities from fitted extreme value distributions (EVDs) or draw random samples from them.

Usage

```
pextRemes(x, q, lower.tail = TRUE, ...)

rextRemes(x, n, ...)

## S3 method for class 'fevd'
pextRemes(x, q, lower.tail = TRUE, ..., qcov = NULL)

## S3 method for class 'fevd.bayesian'
pextRemes(x, q, lower.tail = TRUE, ...,
    qcov = NULL, burn.in = 499, FUN = "mean")

## S3 method for class 'fevd.lmoments'
pextRemes(x, q, lower.tail = TRUE, ...)

## S3 method for class 'fevd.mle'
pextRemes(x, q, lower.tail = TRUE, ..., qcov = NULL)

## S3 method for class 'fevd'
rextRemes(x, n, ...)
```

```
## S3 method for class 'fevd.bayesian'
rextRemes(x, n, ..., burn.in = 499, FUN = "mean",
    qcov = NULL)

## S3 method for class 'fevd.lmoments'
rextRemes(x, n, ...)

## S3 method for class 'fevd.mle'
rextRemes(x, n, ..., qcov = NULL)
```

Arguments

x	A list object of class "fevd" as returned by fevd.
q	Vector of quantiles.
n	number of random draws to take from the model.
qcov	numeric matrix with rows the same length as q and columns equal to the number of parameters (+ 1 for the threshold, if a POT model). This gives any covariate values for a nonstationary model. If NULL, and model is non-stationary, only the intercept terms for modeled parameters are used, and if a non-constant threshold, only the first threshold value is used. Not used if model is stationary.
lower.tail	logical; if TRUE (default), probabilities are P[X <= x] otherwise, P[X > x].
burn.in	the burn in period.
FUN	cahracter string naming a function, or a function, to be used to find the parameter estimates from the posterior df. Default is the posterior mean.
...	Not used.

Details

These functions are essentially wrapper functions for the low-level functions pevd and revd. The idea is that they take parameter values from a fitted model in order to calculate probabilities or draw random samples. In the case of non-stationary models, for probabilities, covariate values should be given. If not, the intercept terms (or first threshold value) are used only; and a warning is given. In the case of rextRemes for non-stationary values, n samples of length equal to the length of the data set to which the model was fit are generated and returned as a matrix. In this case, the random draws represent random draws using the current covariate values.

The extreme value distributions (EVD's) are generalized extreme value (GEV) or generalized Pareto (GP). The point process characterization is an equivalent form, but is not handled here; parameters are converted to those of the (approx.) equivalent GP df. The GEV df is given by

Pr(X <= x) = G(x) = exp[-(1 + shape*(x - location)/scale)^(-1/shape)]

for 1 + shape*(x - location) > 0 and scale > 0. It the shape parameter is zero, then the df is defined by continuity and simplies to

G(x) = exp(-exp((x - location)/scale)).

The GEV df is often called a family of df's because it encompasses the three types of EVD's: Gumbel (shape = 0, light tail), Frechet (shape > 0, heavy tail) and the reverse Weibull (shape < 0,

bounded upper tail at location - scale/shape). It was first found by R. von Mises (1936) and also independently noted later by meteorologist A. F. Jenkins (1955). It enjoys theretical support for modeling maxima taken over large blocks of a series of data.

The generalized Pareo df is given by (Pickands, 1975)

Pr(X <= x) = F(x) = 1 - [1 + shape*(x - threshold)/scale]^(-1/shape)

where 1 + shape*(x - threshold)/scale > 0, scale > 0, and x > threshold. If shape = 0, then the GP df is defined by continuity and becomes

F(x) = 1 - exp(-(x - threshold)/scale).

There is an approximate relationship between the GEV and GP df's where the GP df is approximately the tail df for the GEV df. In particular, the scale parameter of the GP is a function of the threshold (denote it scale.u), and is equivalent to scale + shape*(threshold - location) where scale, shape and location are parameters from the "equivalent" GE Vdf. Similar to the GEV df, the shape parameter determines the tail behavior, where shape = 0 gives rise to the exponential df (light tail), shape > 0 the Pareto df (heavy tail) and shape < 0 the Beta df (bounded upper tail at location - scale.u/shape). Theoretical justification supports the use of the GP df family for modeling excesses over a high threshold (i.e., y = x - threshold). It is assumed here that `x`, `q` describe x (not y = x - threshold). Similarly, the random draws are y + threshold.

See Coles (2001) and Reiss and Thomas (2007) for a very accessible text on extreme value analysis and for more theoretical texts, see for example, Beirlant et al. (2004), de Haan and Ferreira (2006), as well as Reiss and Thomas (2007).

Value

A numeric vector of probabilites or random sample is returned. In the case of non-stationary models, a matrix of random samples is returned by `rextRemes`.

Warning

In the case of non-stationary models, the code in its current state is somewhat less than ideal. It requires great care on the part of the user. In particular, the `qcov` argument becomes critical. Parameters that are fixed in the model can be changed if `qcov` is not correctly used. Any parameter that is fixed at a given value (including the intercept terms) should have all ones in their columns. Presently, nothing in the code will force this requirement to be upheld. Using `make.qcov` will help, as it has some checks to ensure constant-valued parameters have all ones in their columns.

Note

It is recommended to use `make.qcov` when creating a `qcov` matrix.

Author(s)

Eric Gilleland

References

Beirlant, J., Goegebeur, Y., Teugels, J. and Segers, J. (2004) *Statistics of Extremes: Theory and Applications*. Chichester, West Sussex, England, UK: Wiley, ISBN 9780471976479, 522pp.

Coles, S. (2001) *An introduction to statistical modeling of extreme values*, London, U.K.: Springer-Verlag, 208 pp.

de Haan, L. and Ferreira, A. (2006) *Extreme Value Theory: An Introduction*. New York, NY, USA: Springer, 288pp.

Jenkinson, A. F. (1955) The frequency distribution of the annual maximum (or minimum) of meteorological elements. *Quart. J. R. Met. Soc.*, **81**, 158–171.

Pickands, J. (1975) Statistical inference using extreme order statistics. *Annals of Statistics*, **3**, 119–131.

Reiss, R.-D. and Thomas, M. (2007) *Statistical Analysis of Extreme Values: with applications to insurance, finance, hydrology and other fields*. Birkhauser, 530pp., 3rd edition.

von Mises, R. (1936) La distribution de la plus grande de n valeurs, *Rev. Math. Union Interbalcanique* **1**, 141–160.

See Also

pevd, revd, fevd, make.qcov

Examples

```
z <- revd(100, loc=20, scale=0.5, shape=-0.2)
fit <- fevd(z)
fit

pextRemes(fit, q=quantile(z, probs=c(0.85, 0.95, 0.99)), lower.tail=FALSE)

z2 <- rextRemes(fit, n=1000)
qqplot(z, z2)

## Not run:
data(PORTw)
fit <- fevd(TMX1, PORTw, units="deg C")
fit

pextRemes(fit, q=c(17, 20, 25, 30), lower.tail=FALSE)
# Note that fit has a bounded upper tail at:
# location - scale/shape ~
# 15.1406132 + (2.9724952/0.2171486) = 28.82937
#
# which is why P[X > 30] = 0.  Note also that 25
# is less than the upper bound, but larger than
# the maximum observed value.

z <- rextRemes(fit, n=50)
qqplot(z, PORTw$TMX1, xlab="Simulated Data Quantiles",
    ylab="Data Quantiles (PORTw TMX1)")

# Not a great fit because data follow a non-stationary
# distribution.
fit <- fevd(TMX1, PORTw, location.fun=~AOindex, units="deg C")
fit
```

```
pextRemes(fit, q=c(17, 20, 25, 30), lower.tail=FALSE)
# Gives a warning because we did not give covariate values.

v <- make.qcov(fit, vals=list(mu1=c(1, -1, 1, -1)))
v
# find probabilities for high positive AOindex vs
# low negative AOindex.  A column for the unnecessary
# threshold is added, but is not used.

pextRemes(fit, q=c(17, 17, 30, 30), qcov=v, lower.tail=FALSE)

z <- rextRemes(fit, n=50)
dim(z)
qqplot(z[,1], PORTw$TMX1, xlab="Simulated Data Quantiles",
    ylab="Data Quantiles (PORTw TMX1)")

qqplot(z[,28], PORTw$TMX1, xlab="Simulated Data Quantiles",
    ylab="Data Quantiles (PORTw TMX1)")
# etc.

##
## GP model with non-constant threshold.
##
fit <- fevd(-MinT ~1, Tphap, threshold=c(-70,-7),
    threshold.fun=~I((Year - 48)/42), type="GP",
    time.units="62/year", verbose=TRUE)
fit

summary(fit$threshold)
v <- make.qcov(fit, vals=c(rep(1,8), c(-77, -73.5, -71.67, -70)), nr=4)
v

# upper bounded df at: u - scale/shape =
c(-77, -73.5, -71.67, -70) + 2.9500992/0.1636367
# -58.97165 -55.47165 -53.64165 -51.97165
summary(-Tphap$MinT)
pextRemes(fit, q=rep(-58, 4), qcov=v, lower.tail=FALSE)



## End(Not run)
```

PORTw *Annual Maximum and Minimum Temperature*

Description

Annual maximum and minimum Winter temperature (degrees centigrade) with a covariate for the North Atlantic Oscillation index from 1927 through 1995. Data is for Winter for Port Jervis, New York (PORTw) and Spring for Sept-Iles, Quebec (SEPTsp).

Usage

```
data(PORTw)
```

Format

A data frame with 68 observations on the following 16 variables.

Year a numeric vector giving the year.

MTMAX a numeric vector giving the mean winter maximum temperatures (degrees centigrade).

MTMIN a numeric vector giving the mean winter minimum temperatures (degrees centigrade).

STDTMAX a numeric vector giving the standard deviations of maximum winter temperatures (degrees centigrade).

STDMIN a numeric vector giving the standard deviations of minimum winter temperatures (degrees centigrade).

TMX1 a numeric vector giving the maximum winter temperature (degrees centigrade).

TMN0 a numeric vector giving the minimum winter temperature (degrees centigrade).

MDTR a numeric vector giving the mean winter diurnal temperature range (degrees centigrade).

AOindex a numeric vector giving the Atlantic Oscillation index (see Thompson and Wallace (1998)).

Details

See Wettstein and Mearns (2002) for a much more detailed explanation of the above variables.

Source

See Wettstein and Mearns (2002).

References

Thompson, D. W. J. and Wallace, J. M. (1998) The Arctic Oscillation signature in the wintertime geopotential height and temperature fields. *Geophys. Res. Lett.*, **25**, 1297–1300.

Wettstein, J. J. and Mearns, L. O. (2002) The influence of the North Atlantic-Arctic Oscillation on mean, variance and extremes of temperature in the northeastern United States and Canada. *Journal of Climate*, **15**, 3586–3600.

Examples

```
data(PORTw)
str(PORTw)
par( mfrow=c(2,1))
plot(PORTw[,"TMX1"], type="l", lwd=2, xlab="", xaxt="n", ylab="Maximum Temperature (C)")
plot(PORTw[,"TMN0"], type="l", lwd=2, xlab="", xaxt="n", ylab="Minimum Temperature (C)")
```

postmode *Posterior Mode from an MCMC Sample*

Description

Calculate the posterior mode from an MCMC sample for "fevd" objects.

Usage

```
postmode(x, burn.in = 499, verbose = FALSE, ...)

## S3 method for class 'fevd'
postmode(x, burn.in = 499, verbose = FALSE, ...)
```

Arguments

x	An object of class "fevd" where component method = "Bayesian".
burn.in	The furst burn.in samples from the posterior distribution will be removed before calculation.
verbose	logical, should progress information be printed to the screen.
...	Not used.

Details

The log-likelihood and (log) prior is calculated for every sample from the chain, and added together, giving h. The parameters from the sample that yield the maximum value for h are returned. If more than one set of parameters yield a maximum, their average is returned.

Value

A named numeric vector is returned giving the paramter values.

Author(s)

Eric Gilleland

See Also

fevd, findpars

Examples

```
data(ftcanmax)

fit <- fevd(Prec, ftcanmax, method="Bayesian", iter = 1000, verbose=TRUE)

postmode(fit)
```

Potomac *Potomac River Peak Stream Flow Data.*

Description

Potomac River peak stream flow (cfs) data for water years (Oct-Sep) 1895 through 2000 at Point Rocks, Maryland.

Usage

```
data(Potomac)
```

Format

A data frame with 106 observations on the following 2 variables.

Year a numeric vector giving the water year (Oct-Sep).

Flow a numeric vector the peak stream flow (cfs; 1 cfs = 0.028317 cubic meters per second).

Details

Potomac River peak stream flow (cfs) data for water years (Oct-Sep) 1895 through 2000 at Point Rocks, Maryland.

These data (up to 1986) have been analyzed by Smith (1987) and this entire dataset by Katz et al. (2002) Sec. 2.3.2.

Source

U.S. Geological Survey.

References

Katz, R. W., Parlange, M. B. and Naveau, P. (2002) Statistics of extremes in hydrology. *Advances in Water Resources*, **25**, 1287–1304.

Smith, J. A. (1987) Regional flood frequency analysis using extreme order statistics of the annual peak record. *Water Resour Res*, **23**, 1657–1666.

Examples

```
data(Potomac)
str(Potomac)
# Fig. 3 of Katz et al. (2002) Sec. 2.3.2.
plot(Potomac[,"Year"], Potomac[,"Flow"]/1000, yaxt="n", ylim=c(0,500), type="l", lwd=1.5,
    xlab="Water Year (Oct-Sept)", ylab="Annual peak flow (thousand cfs)")
axis(2,at=seq(0,500,100),labels=as.character(seq(0,500,100)))
```

profliker *Profile Likelihood Function*

Description

Find the profile likelihood for a range of values for an extreme value df (EVD).

Usage

```
profliker(object, type = c("return.level", "parameter"), xrange = NULL,
    return.period = 100, which.par = 1, nint = 20, plot = TRUE, gr = NULL,
    method = "BFGS", lower = -Inf, upper = Inf, control = list(), ...)
```

Arguments

object	A list object of class "fevd" as returned by fevd.
type	character string stating whether the parameter of interest is a regular parameter or a return level.
xrange	numeric vector of length two giving the range of values of the parameter over which to calculate the profile likelihood.
return.period	If a return level is of interest, this number gives its associated return period.
which.par	If a parameter is of interest, this number tells for which component of the parameter vector to do the profile likelihood.
nint	The profile likelihood is calculated for a sequence of nint values covering xrange.
plot	logical; should a plot of the likelihood be made? Note that this is controlled by the verbose argument in the ci method function for MLE fevd objects when "proflik" is chosen as the method for finding confidence intervals. It is usually a good idea to plot the profile likelihood to see if the confidence intervals are really found or not.
gr, method, lower, upper, control	optional arguments to optim.
...	optional arguments to plot.

Details

See the help file for ci.fevd.mle for more details on this approach.

Value

A numeric vector is returned invisibly.

Author(s)

Eric Gilleland

See Also

ci.fevd.mle, fevd

Examples

```
z <- revd(100, loc=20, scale=0.5, shape=-0.2)
fit <- fevd(z)
fit

profliker(fit, type="parameter", which.par=3)

profliker(fit, type="parameter",
    xrange=c(-0.35, -0.2), which.par=3)
```

---

qqnorm *Normal qq-plot with 95 Percent Simultaneous Confidence Bands*

Description

Calculates a normal qq-plot for a vector of data along with 95 percent simultaneous confidence bands.

Usage

```
qqnorm(y, pch = 20, xlab = "Standard Normal Quantiles", ylab = "Sample Quantiles",
    make.plot = TRUE, ...)
```

Arguments

y	numeric vector of data.
pch	plot symbol to use.
xlab	Character string giving abscissa label.
ylab	Character string giving ordinate axis label.
make.plot	logical, should the plot be created (TRUE) or not (FALSE)?
...	optional arguments to the plot function.

Details

Confidence intervals are calculated using +/- k, where

k = 0.895 / (sqrt(n) * (1- 0.01 / sqrt(n) + 0.85/n))

Gives a 95 percent asymptotic band based on the Kolmogorov-Smirnov statistic (Doksum and Sievers, 1976).

Value

A data frame object is returned invisibly with components:

x, y	the data and standard normal quantiles, resp.
lower, upper	lower and upper 95 percent confidence bands.

Author(s)

Peter Guttorp, peter "at" stat.washington.edu, modified by Eric Gilleland

References

Doksum, K. A. and G. L. Sievers, 1976. Plotting with confidence: graphical comparisons of two populations. Biometrika, 63 (3), 421–434.

See Also

qnorm, qqplot, shiftplot

Examples

```
z <- rexp(100)
qqnorm( z)

y <- rnorm( 100)
qqnorm( y)
obj <- qqnorm(y, make.plot=FALSE)
str(obj)

data( ftcanmax)
qqnorm( ftcanmax[,"Prec"])
```

---

qqplot *qq-plot Between Two Vectors of Data with 95 Percent Confidence Bands*

Description

QQ-plot between two data vectors with 95 percent confidence bands based on the Kolmogorov-Smirnov statistic (Doksum and Sievers, 1976).

Usage

```
qqplot(x, y, pch = 20, xlab = "x Quantiles", ylab = "y Quantiles", regress = TRUE,
    make.plot = TRUE, ...)

## S3 method for class 'qqplot'
plot(x, ...)
```

```
## S3 method for class 'qqplot'
summary(object, ...)
```

Arguments

x	qqplot: numeric vector of length 'm' giving one data set. plot method function: list object of class "qqplot" returned by qqplot.
object	list object of class "qqplot" returned by qqplot.
y	numeric vector of length 'n' giving the other data set.
pch	Plot character.
xlab	Character string giving the label for the abscissa axis.
ylab	Character string giving the label for the ordinate axis.
regress	logical, should a regression line be fit to the quantiles?
make.plot	logical, should the plot be created (TRUE) or not (FALSE)?
...	Other optional arguments to the plot function. Not used by summary method function.

Details

Plots the sorted (missing-values removed) 'x' values against the sorted, and interpolated (via the approxfun function from package stats), 'y' values. Confidence bands are about the sorted and interpolated 'y' values using +/- K/sqrt(M), where

K = 1.36

and

M = m*n / (m+n).

The plot method function does exactly the same thing as qqplot except that it does not need to do any calculations.

The summary method function merely displays the original call to the function unless a regression line was fit between the quantiles, in which case summary information is displayed for the regression (i.e., the summary method function for lm is run on the "lm" object).

Value

An object of class "qqplot" is invisibly returned by each function (in the case of the method functions, the object entered is simply returned invisibly). This is a list object with components:

call	calling string
names	list object with components x and y giving the object names for the objects passed into x and y, resp.
regression	If regress was TRUE, then this is the fitted regression object as returned by lm. Otherwise, this component is not included.
qdata	data frame with components: x and y giving the quantiles for x and y, resp., and lower and upper giving the lower and upper 95 percent confidence bands, resp.

Author(s)

Peter Guttorp, peter "at" stat.washington.edu

References

Doksum, K.A. and G.L. Sievers, 1976. Plotting with confidence: graphical comparisons of two populations. Biometrika, 63 (3), 421–434.

See Also

approxfun, qqnorm, shiftplot

Examples

```
z <- rnorm(100)
y <- rexp(100)
qqplot( z, y)
qqplot( y, z)

data( ftcanmax)
qqplot( ftcanmax[,"Prec"], z)
obj <- qqplot( ftcanmax[,"Prec"], y, make.plot=FALSE)
plot(obj)
summary(obj)
```

---

return.level *Return Level Estimates*

Description

Return level estimates from fitted fevd model objects.

Usage

```
return.level(x, return.period = c(2, 20, 100), ...)

## S3 method for class 'fevd'
return.level(x, return.period = c(2, 20, 100), ...)

## S3 method for class 'fevd.bayesian'
return.level(x, return.period = c(2, 20, 100), ..., do.ci = FALSE,
                 burn.in = 499, FUN = "mean", qcov = NULL, qcov.base =
                 NULL)

## S3 method for class 'fevd.lmoments'
return.level(x, return.period = c(2, 20, 100), ...,
    do.ci = FALSE)
```

```
## S3 method for class 'fevd.mle'
return.level(x, return.period = c(2, 20, 100), ...,
    do.ci = FALSE, qcov = NULL, qcov.base = NULL)

## S3 method for class 'ns.fevd.bayesian'
return.level(x, return.period = 100, ...,
    burn.in = 499, FUN = "mean", do.ci = FALSE, verbose = FALSE,
    qcov = NULL, qcov.base = NULL)

## S3 method for class 'ns.fevd.mle'
return.level(x, return.period = c(2, 20, 100), ...,
    alpha = 0.05, method = c("normal"), do.ci = FALSE, verbose = FALSE,
    qcov = NULL, qcov.base = NULL)

## S3 method for class 'return.level'
print(x, ...)
```

Arguments

x	A list object of class "fevd" as returned by fevd. In the case of the print method function, an object returned by return.level.
return.period	numeric vector of desired return periods. For return.level.ns.fevd.mle, this must have length one.
qcov	numeric matrix with rows the same length as q and columns equal to the number of parameters (+ 1 for the threshold, if a POT model). This gives any covariate values for a nonstationary model. If NULL, and model is non-stationary, only the intercept terms for modeled parameters are used, and if a non-constant threshold, only the first threshold value is used. Not used if model is stationary.
qcov.base	numeric matrix analogous to qcov. When provided, the function returns the difference in return levels between the level for the covariates in qcov and the level for covariates in qcov.base.
do.ci	logical; should CIs be returned as well?
burn.in	number giving the burn in value. The first 1:burn.in will not be used in obtaining parmaeter estimates.
FUN	character string naming a function, or a function, to use to find the parameter estimates from the MCMC sample. Default is to take the posterior mean (after burn in).
alpha	The (1 - alpha) * 100 percent confidence level for confidence intervals of return levels in non-stationary models.
method	character string naming which CI method to employ.
verbose	logical, should progress information be printed to the screen?
...	For the stationary case only, any optional arguments to the ci function. Not used by the print method function.

Details

The extreme value distributions (EVD's) are generalized extreme value (GEV) or generalized Pareto (GP). The point process characterization is an equivalent form, but is not handled here. The GEV df is given by

Pr(X <= x) = G(x) = exp[-(1 + shape*(x - location)/scale)^(-1/shape)]

for 1 + shape*(x - location) > 0 and scale > 0. It the shape parameter is zero, then the df is defined by continuity and simplies to

G(x) = exp(-exp((x - location)/scale)).

The GEV df is often called a family of df's because it encompasses the three types of EVD's: Gumbel (shape = 0, light tail), Frechet (shape > 0, heavy tail) and the reverse Weibull (shape < 0, bounded upper tail at location - scale/shape). It was first found by R. von Mises (1936) and also independently noted later by meteorologist A. F. Jenkins (1955). It enjoys theretical support for modeling maxima taken over large blocks of a series of data.

The generalized Pareo df is given by (Pickands, 1975)

Pr(X <= x) = F(x) = 1 - [1 + shape*(x - threshold)/scale]^(-1/shape)

where 1 + shape*(x - threshold)/scale > 0, scale > 0, and x > threshold. If shape = 0, then the GP df is defined by continuity and becomes

F(x) = 1 - exp(-(x - threshold)/scale).

There is an approximate relationship between the GEV and GP df's where the GP df is approximately the tail df for the GEV df. In particular, the scale parameter of the GP is a function of the threshold (denote it scale.u), and is equivalent to scale + shape*(threshold - location) where scale, shape and location are parameters from the "equivalent" GE Vdf. Similar to the GEV df, the shape parameter determines the tail behavior, where shape = 0 gives rise to the exponential df (light tail), shape > 0 the Pareto df (heavy tail) and shape < 0 the Beta df (bounded upper tail at location - scale.u/shape). Theoretical justification supports the use of the GP df family for modeling excesses over a high threshold (i.e., y = x - threshold). It is assumed here that `x`, `q` describe x (not y = x - threshold). Similarly, the random draws are y + threshold.

See Coles (2001) and Reiss and Thomas (2007) for a very accessible text on extreme value analysis and for more theoretical texts, see for example, Beirlant et al. (2004), de Haan and Ferreira (2006), as well as Reiss and Thomas (2007).

Return levels are essentially the same as quantiles. In the case of the GEV family, they are the same. In the case of the GP df, they are very similar, but the exceedance rate is taken into consideration. For non-stationary modeling, effective return levels are calculated for each value of the covariate(s) used in the model fit (see, e.g., Gilleland and Katz, 2011).

`return.level.ns.fevd.mle` allows one to estimate the difference in return levels for a non-stationary model, based on subtracting the return levels for `qcov.base` from those for `qcov`, in which case the outputted values and CIs pertain to differences in return levels.

Value

If do.ci is FALSE, an object of class "return.level" is returned, which is either a numeric vector (stationary models) of length equal to the `return.period` argument giving the return levels, or a matrix of dimension equal to either n by np or q by np where n is the length of the data used to fit the model and np are the number of return periods, and q is the number of rows of qcov, if supplied. The

returned value also includes useful attributes describing how the return levels came to be estimated. In particular, the list of attributes include:

`return.period`	the return periods associated with the estimated return levels.
`data.name`	same as the data.name component of the fevd object.
`fit.call`, `call`	the original call for the fitted object and the call to this function, resp.
`fit.type`	character string naming which type of EVD was fit to the data, and subsequently used to estimate the return levels.
`data.assumption`	character string stating whether the model is stationary or non-stationary.
`period`	character string stating what the units (period.basis from the fevd object) of the period are.
`units`	character string giving the data units, if available.
`qcov`	name of the qcov matrix used to obtain the effective return levels.
`qcov.base`	when provided as input, the name of the qcov.base matrix used to obtain the difference in effective return levels.

If do.ci is TRUE, then an object returned by the appropriate ci function is returned (stationary case only).

Author(s)

Eric Gilleland

References

Beirlant, J., Goegebeur, Y., Teugels, J. and Segers, J. (2004) *Statistics of Extremes: Theory and Applications*. Chichester, West Sussex, England, UK: Wiley, ISBN 9780471976479, 522pp.

Coles, S. (2001) *An introduction to statistical modeling of extreme values*, London, U.K.: Springer-Verlag, 208 pp.

Gilleland, E. and Katz, R. W. (2011). New software to analyze how extremes change over time. *Eos*, 11 January, **92**, (2), 13–14.

de Haan, L. and Ferreira, A. (2006) *Extreme Value Theory: An Introduction*. New York, NY, USA: Springer, 288pp.

Jenkinson, A. F. (1955) The frequency distribution of the annual maximum (or minimum) of meteorological elements. *Quart. J. R. Met. Soc.*, **81**, 158–171.

Pickands, J. (1975) Statistical inference using extreme order statistics. *Annals of Statistics*, **3**, 119–131.

Reiss, R.-D. and Thomas, M. (2007) *Statistical Analysis of Extreme Values: with applications to insurance, finance, hydrology and other fields*. Birkhauser, 530pp., 3rd edition.

von Mises, R. (1936) La distribution de la plus grande de n valeurs, *Rev. Math. Union Interbalcanique* **1**, 141–160.

See Also

`pextRemes`, `fevd`, `rlevd`, `ci.rl.ns.fevd.bayesian`

Examples

```
z <- revd(100, loc=20, scale=0.5, shape=-0.2)
fit <- fevd(z)
fit

return.level(fit)

fitLM <- fevd(z, method="Lmoments")
fitLM
return.level(fitLM)

## Not run:
fitB <- fevd(z, method="Bayesian", verbose=TRUE)
fitB

return.level(fitB)


## End(Not run)
```

`revtrans.evd` *Reverse Transformation*

Description

Reverse transform standardized data to follow a non-standardized extreme value distribution (EVD).

Usage

```
revtrans.evd(z, threshold = NULL, location = NULL, scale, shape = NULL,
    type = c("GEV", "GP", "PP", "Gumbel", "Weibull", "Frechet",
    "Exponential", "Beta", "Pareto"))
```

Arguments

`z`	numeric vector of data of length n following a standardized EVD.
`threshold`	number or numeric vector of length n giving the threshold, if applicable.
`location`	number or numeric vector of length n giving the location parameter(s), if applicable.
`scale`	number or or numeric vector of length n giving the scale parameter(s).
`shape`	number or numeric vector of length n giving the shape parameter(s), if applicable.
`type`	character string naming to what EVD should the data be reverse-transformed. Default is GEV df.

Details

For standardized EVD data (e.g., via `trans`), this function performs the reverse transformation back to the original scale.

Value

numeric vector of length n.

Author(s)

Eric Gilleland

See Also

`trans`, `trans.fevd`, `fevd`

Examples

```
data(PORTw)

fit <- fevd(TMX1, PORTw, location.fun=~AOindex, units="deg C")
fit

z <- trans(fit)

fevd(z)

p <- findpars(fit)

y <- revtrans.evd(z=z, location=p$location, scale=2.6809613,
    shape=-0.1812824)

fevd(y)

qqplot(y, PORTw$TMX1)
```

`rlevd` *Return Levels for Extreme Value Distributions*

Description

Calculate return levels for extreme value distributions (EVDs).

Usage

```
rlevd(period, loc = 0, scale = 1, shape = 0, threshold = 0,
    type = c("GEV", "GP", "PP", "Gumbel", "Frechet", "Weibull",
    "Exponential", "Beta", "Pareto"),
    npy = 365.25, rate = 0.01)
```

Arguments

period	numeric vector giving the desired return periods.
loc, scale, shape	single numbers giving the parameter values.
threshold	number giving the threshold, if applicable.
type	character string naming which EVD to calculate return levels from. If type is "PP", then it is converted to "GEV".
npy	number stating how many values per year.
rate	The rate of exceedance.

Details

The extreme value distributions (EVD's) are generalized extreme value (GEV) or generalized Pareto (GP). The point process characterization is an equivalent form, but is not handled here. The GEV df is given by

Pr(X <= x) = G(x) = exp[-(1 + shape*(x - location)/scale)^(-1/shape)]

for 1 + shape*(x - location) > 0 and scale > 0. It the shape parameter is zero, then the df is defined by continuity and simplies to

G(x) = exp(-exp((x - location)/scale)).

The GEV df is often called a family of df's because it encompasses the three types of EVD's: Gumbel (shape = 0, light tail), Frechet (shape > 0, heavy tail) and the reverse Weibull (shape < 0, bounded upper tail at location - scale/shape). It was first found by R. von Mises (1936) and also independently noted later by meteorologist A. F. Jenkins (1955). It enjoys theretical support for modeling maxima taken over large blocks of a series of data.

The generalized Pareo df is given by (Pickands, 1975)

Pr(X <= x) = F(x) = 1 - [1 + shape*(x - threshold)/scale]^(-1/shape)

where 1 + shape*(x - threshold)/scale > 0, scale > 0, and x > threshold. If shape = 0, then the GP df is defined by continuity and becomes

F(x) = 1 - exp(-(x - threshold)/scale).

There is an approximate relationship between the GEV and GP df's where the GP df is approximately the tail df for the GEV df. In particular, the scale parameter of the GP is a function of the threshold (denote it scale.u), and is equivalent to scale + shape*(threshold - location) where scale, shape and location are parameters from the "equivalent" GE Vdf. Similar to the GEV df, the shape parameter determines the tail behavior, where shape = 0 gives rise to the exponential df (light tail), shape > 0 the Pareto df (heavy tail) and shape < 0 the Beta df (bounded upper tail at location - scale.u/shape). Theoretical justification supports the use of the GP df family for modeling excesses

over a high threshold (i.e., y = x - threshold). It is assumed here that x, q describe x (not y = x - threshold). Similarly, the random draws are y + threshold.

See Coles (2001) and Reiss and Thomas (2007) for a very accessible text on extreme value analysis and for more theoretical texts, see for example, Beirlant et al. (2004), de Haan and Ferreira (2006), as well as Reiss and Thomas (2007).

Return levels are essentially the same as quantiles. In the case of the GEV family, they are the same. In the case of the GP df, they are very similar, but the exceedance rate is taken into consideration.

Value

named numeric vector of same length as `period` giving the calculated return levels for each return period.

Note

Currently, this function does not handle the PP type. Return levels for this case can be handled in several different ways. For example, they could be calculated from the equivalent GEV df or equivalent GP df. In any case, one needs first to determine how to handle the frequency component.

Author(s)

Eric Gilleland

References

Beirlant, J., Goegebeur, Y., Teugels, J. and Segers, J. (2004) *Statistics of Extremes: Theory and Applications*. Chichester, West Sussex, England, UK: Wiley, ISBN 9780471976479, 522pp.

Coles, S. (2001) *An introduction to statistical modeling of extreme values*, London, U.K.: Springer-Verlag, 208 pp.

de Haan, L. and Ferreira, A. (2006) *Extreme Value Theory: An Introduction*. New York, NY, USA: Springer, 288pp.

Jenkinson, A. F. (1955) The frequency distribution of the annual maximum (or minimum) of meteorological elements. *Quart. J. R. Met. Soc.*, **81**, 158–171.

Pickands, J. (1975) Statistical inference using extreme order statistics. *Annals of Statistics*, **3**, 119–131.

Reiss, R.-D. and Thomas, M. (2007) *Statistical Analysis of Extreme Values: with applications to insurance, finance, hydrology and other fields*. Birkhauser, 530pp., 3rd edition.

von Mises, R. (1936) La distribution de la plus grande de n valeurs, *Rev. Math. Union Interbalcanique* **1**, 141–160.

See Also

`devd`, `return.level`

Examples

```
rlevd(c(2, 20, 100), loc=10, scale=2, shape=0.5)

rlevd(c(2, 20, 100), scale=2, shape=0.5, type="GP")
```

Rsum *Hurricane Frequency Dataset.*

Description

This dataset gives the number of hurricanes per year (from 1925 to 1995) as well as the ENSO state and total monetary damage.

Usage

```
data(Rsum)
```

Format

A data frame with 71 observations on the following 4 variables.

Year a numeric vector giving the year.

EN a numeric vector giving the ENSO state (-1 for La Ninha, 1 for El Ninho and 0 otherwise).

Ct a numeric vector giving the number of hurricanes for the corresponding year.

TDam a numeric vector giving the total monetary damage (millions of U.S. dollars).

Details

More information on these data can be found in Pielke and Landsea (1998) or Katz (2002).

References

Katz, R. W. (2002) Stochastic modeling of hurricane damage. *Journal of Applied Meteorology*, **41**, 754–762.

Pielke, R. A. and Landsea, C. W. (1998) Normalized hurricane damages in the United States: 1925-95. *Weather and Forecasting*, **13**, (3), 621–631.

Examples

```
data(Rsum)
str(Rsum)
plot(Rsum)

# Reproduce Fig. 1 of Katz (2002).
plot( Rsum[,"Year"], Rsum[,"TDam"]/1000, type="h", xlab="Year",
ylab="Total damage (billion U.S. dollars)",
```

```
ylim=c(0,80), lwd=2)

# Reproduce Fig. 2 of Katz (2002).
plot(Rsum[,"Year"],Rsum[,"Ct"],type="h", xlab="Year",
    ylab="Number of Hurricanes", ylim=c(0,5), lwd=2)
```

SantaAna *Santa Ana Winds Data*

Description

Meteorological data pertaining to Santa Ana winds.

Usage

```
data("SantaAna")
```

Format

The format is: chr "SantaAna"

Examples

```
# data(SantaAna)
## maybe str(SantaAna) ; plot(SantaAna) ...
```

---

shiftplot *Shift Plot Between Two Sets of Data*

Description

A shift plot is a plot of the quantiles of a data set y minus those of another data set x against those of x. Includes 95 percent simultaneous confidence bands.

Usage

```
shiftplot(x, y, pch = 20, xlab = "x Quantiles", ylab = "y Quantiles", main = NULL, ...)
```

Arguments

x	numeric vector of length m.
y	numeric vector of length n.
pch	Plotting character.
xlab	Character string giving abscissa axis label.
ylab	Character string giving ordinate axis label.
main	Character string giving plot title.
...	Other optional arguments to plot function.

Details

The shift plot is a graph of y_q - x_q vs. x_q, where y_q and x_q denote the quantiles of x and y, resp. 95 percent simultaneous confidence bands are calculated per Doksum and Sievers (1976). The primary usage of this plot is where x is a control group and y is an experimental method; or something similar. For example, x might represent observations, and y might represent climate model output; or some such.

Value

No value is returned, but a plot is created.

Author(s)

Peter Guttorp

References

Doksum, K. A. and Sievers, G. L. (1976) Plotting with confidence: graphical comparisons of two populations. *Biometrika*, **63**, (3), 421–434.

See Also

qqplot, qqnorm, approxfun

Examples

```
z <- rnorm( 100)
y <- rexp(30)
shiftplot( z, y)

data( ftcanmax)
shiftplot( y, ftcanmax[,"Prec"])
```

---

strip *Strip Fitted EVD Object of Everything but the Parameter Estimates*

Description

Take any fevd object, regardless of estimation method, and return only a vector of the estimated parameters.

Usage

```
strip(x, use.names = TRUE, ...)
## S3 method for class 'fevd'
strip(x, use.names = TRUE, ...)
```

Arguments

x	An object of class "fevd".
use.names	logical stating whether or not to keep the names attribute
...	For the Bayesian method, if an alternative function to taking the mean or posterior mode of the MCMC samples is used, then optional arguments may be passed. Otherwise, not used.

Details

This function is very similar to `distill`, but returns less information.

Value

numeric vector with the parameter estimates.

Author(s)

Eric Gilleland

See Also

`distill.fevd`

Examples

```
z <- revd(100, loc=20, scale=0.5, shape=-0.2)
fit <- fevd(z)
fit

strip( fit )
strip( fit, use.names = FALSE )

# Compare with ...
distill( fit )
distill( fit, cov = FALSE )

## Not run:
data( "Fort" )
fit <- fevd(Prec, Fort, threshold=0.395,
    scale.fun=~sin(2 * pi * (year - 1900)/365.25) +
        cos(2 * pi * (year - 1900)/365.25),
    type="PP", method="Bayesian", iter=1999, use.phi=TRUE, verbose=TRUE)

fit

strip( fit )
strip( fit, burn.in = 700 )
strip( fit, FUN = "postmode" )


## End(Not run)
```

`taildep` *Tail Dependence*

Description

Function to calculate the estimated tail dependence parameters chi and chibar.

Usage

```
taildep(x, y, u, type = c("all", "chi", "chibar"), na.rm = FALSE)
```

Arguments

`x, y`	numeric vectors of same length. `x` may be a two-column matrix or data frame, in which case each column is assumed to be the two vectors of interest (both dependence estimates are symmetric so that it does not matter which is which).
`u`	single numeric between 0 and 1 (non-inclusive) giving the probability threshold overwhich to compute the dependence measures (should be close to 1, but low enough to include enough data.
`type`	character string determining which dependence parameter to estimate (chi or chibar). Default estimates both.
`na.rm`	logical, should missing values be removed?

Details

The tail dependence parameters are those described in, e.g., Reiss and Thomas (2007) Eq (2.60) for "chi" and Eq (13.25) "chibar", and estimated by Eq (2.62) and Eq (13.28), resp. See also, Sibuya (1960) and Coles (2001) sec. 8.4, as well as other texts on EVT such as Beirlant et al. (2004) sec. 9.4.1 and 10.3.4 and de Haan and Ferreira (2006).

Specifically, for two series X and Y with associated df's F and G, chi, a function of u, is defined as

chi(u) = Pr[Y > G^(-1)(u) | X > F^(-1)(u)] = Pr[V > u | U > u],

where (U,V) = (F(X),G(Y))–i.e., the copula. Define chi = limit as u goes to 1 of chi(u).

The coefficient of tail dependence, chibar(u) was introduced by Coles et al. (1999), and is given by

chibar(u) = 2*log(Pr[U > u])/log(Pr[U > u, V > u]) - 1.

Define chibar = limit as u goes to 1 of chibar(u).

The associated estimators for the tail dependence parameters employed by these functions are based on the above two coefficients of tail dependence, and are given by Reiss and Thomas (2007) Eq (2.62) and (13.25) as

chi.hat(x, y; u) = sum(x_i > sort(x)[floor(n*u)] and y_i > sort(y)[floor(n*u)])/(n*(1-u)) [based on chi]

and

chibar.hat(x, y; u) = 2*log(1 - u)/log(mean(x_i > sort(x)[floor(n*u)] and y_i > sort(y)[floor(n*u)])) - 1.

Some properties of the above dependence coefficients, chi(u), chi, and chibar(u) and chibar, are that 0 <= chi(u), chi <= 1, where if X and Y are stochastically independent, then chi(u) = 1 - u, and chibar = 0. If X = Y (perfectly dependent), then chi(u) = chi = 1. For chibar(u) and chibar, we have that -1 <= chibar(u), chibar <= 1. If U = V, then chibar = 1. If chi = 0, then chibar < 1 (tail independence with chibar determining the degree of dependence).

Value

numeric vector of length 1 or 2 depending on the type argument giving the estimated tail dependence parameters.

Author(s)

Eric Gilleland

References

Beirlant, J., Goegebeur, Y., Teugels, J. and Segers, J. (2004) *Statistics of Extremes: Theory and Applications*. Chichester, West Sussex, England, UK: Wiley, ISBN 9780471976479, 522pp.

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

Coles, S., Heffernan, J. E., and Tawn, J. A. (1999) Dependence measures for extreme value analyses. *Extremes*, **2**, 339–365.

de Haan, L. and Ferreira, A. (2006) *Extreme Value Theory: An Introduction*. New York, NY, USA: Springer, 288pp.

Reiss, R.-D. and Thomas, M. (2007) *Statistical Analysis of Extreme Values: with applications to insurance, finance, hydrology and other fields*. Birkhauser, 530pp., 3rd edition.

Sibuya, M. (1960) Bivariate extreme statistics. *Ann. Inst. Math. Statist.*, **11**, 195–210.

See Also

atdf, taildep.test

Examples

```
##
## Example where a r.v. is completely dependent in
## terms of the variables, but completely tail
## independent (see Reiss and Thomas p. 75).
z <- runif(100, -1, 0)
w <- -1*(1 + z)
taildep(z,w,u=0.8)

## Not run:
data(FCwx)
taildep(FCwx$MxT, FCwx$MnT, 0.8)
taildep(FCwx$MxT, FCwx$Prec, 0.8)

## End(Not run)
```

`taildep.test` *Tail Dependence Test*

Description

Testing tail dependence against tail independence.

Usage

```
taildep.test(x, y, cthresh = -0.5, trans = "relative.rank", na.action = na.fail, ...)

relative.rank(x, div = "n", ...)
```

Arguments

`x, y`	numeric vectors of same length. For `taildep.test`, `x` may be a two-column matrix or data frame, in which case each column is assumed to be the two vectors of interest.
`cthresh`	single numeric between -1 and 0 (non-inclusive) over which the transformed and shifted x + y variable is tested (see Details).
`trans`	character string naming a function to transform the x and y variables so that they are in the lower left quadrant (see Details). If variables are already transformed as such (or it is not necessary), then use "identity".
`div`	character one of "n" or "n+1" stating whether to divide the ranks by n or n + 1 so that the reslting transformations are in [0,1] or (0,1), resp.
`na.action`	function to be called to handle missing values.
`...`	optional arguments to the `trans` function. In the case of `relative.rank` these are optional arguments to the function `rank`.

Details

This is the tail dependence test described in Reiss and Thomas (2007) section 13.3. It is, unusually, a test whose null hypothesis is that the two random variables, X and Y, are dependent. So, for example, if a significance level alpha = 0.01 test is desired, then the null huypothesis (dependence) is rejected for values of the statistic with p-values less than 0.01.

To do the test, the variables must first be transformed to the left lower quadrant. Following Reiss and Thomas (2007), the default is to transform the data by means of the sample distribution functions (df's), u = Fhat_n(x) and v = Fhat_n(y) (i.e., using the function `relative.rank`). This yields random variables between 0 and 1, and subsequently they are shifted to be between -1 and 0 (this is done by `taildep.test` so should not be done by the `trans` function).

Ultimately, the test statistic is given by

-(sum(log(c.tilde) + m)/sqrt(m)),

where c.tilde = (u + v)*1(u+v > c)/c, for c a threshold (i.e., `cthresh`). The statistic is assumed to be N(0,1), and the p-value is calculated accordingly.

The test is somewhat sensitive to the choice of threshold, `cthresh`, and it is probably a good idea to try several values (approaching zero from the left). Ideally, the threshold should yield about 10 - 15 percent excesses.

Value

A list object of class "htest" is returned with components:

`call`	the calling string
`data.name`	character vector giving the names of the data sets employed (if x is a matrix, then the second component will be " ".
`method`	character string, which will always be "Reiss-Thomas (13.35)".
`transformation`	same as trans argument.
`parameter`	named vector giving the value of the threshold and any arguments passed to the trans function (perhaps this is not a good idea, and may be changed eventually).
`c.full`	value of the vector u + v after having been transformed and shifted to be between -1 and 0. This is so that the user can adjust the threshold so that 10 - 15 percent of the values exceed it.
`statistic`	numeric giving the value of the test statistic.
`alternative`	character string stating "greater".
`p.value`	numeric between 0 and 1 giving the p-value for the test.

Author(s)

Eric Gilleland

References

Reiss, R.-D. and Thomas, M. (2007) *Statistical Analysis of Extreme Values: with applications to insurance, finance, hydrology and other fields*. Birkhauser, 530pp., 3rd edition.

See Also

`taildep`, `atdf`, `lr.test`

Examples

```
x <- arima.sim(n = 63, list(ar = c(0.8897, -0.4858), ma = c(-0.2279, 0.2488)),
               sd = sqrt(0.1796))

y <- x + rnorm(63)

taildep.test(x, y)

# Recall that null hypothesis is tail dependence!

## Not run:
data(PORTw)
```

```
taildep.test(PORTw$TMX1, PORTw$TMN0, cthresh=-0.3)

data(FCwx)
taildep.test(FCwx$MxT, FCwx$Prec, cthresh=-0.4)

# Run the example (13.3.6) in Reiss and Thomas (2007)
# using the 'wavesurge' dataset from package 'ismev'.
data(wavesurge)
cth <- seq(-0.46,-0.35,0.01)
tab13.1 <- matrix(NA, 2, 12)
colnames(tab13.1) <- as.character(cth)
for(i in 1:12) {
    tmp <- taildep.test(wavesurge, cthresh=cth[i], ties.method="max")
    tab13.1[1,i] <- tmp$parameter["m"]
    tab13.1[2,i] <- tmp$p.value
} # end of for 'i' loop.

rownames(tab13.1) <- c("m", "p-value")
tab13.1

## End(Not run)
```

`threshrange.plot` *Threshold Selection Through Fitting Models to a Range of Thresholds*

Description

Find an appropriate threshold for GP or PP models by fitting them to a sequence of thresholds in order to find the lowest threshold that yields roughly the same parameter estiamtes as any higher threshold.

Usage

```
threshrange.plot(x, r, type = c("GP", "PP", "Exponential"), nint = 10, alpha = 0.05,
    na.action = na.fail, set.panels = TRUE, verbose = FALSE, ...)
```

Arguments

`x`	numeric vector of data.
`r`	numeric vector of length two giving the range of thresholds.
`type`	character string stating which model to fit.
`nint`	number of thresholds to use.
`alpha`	number between zero and one stating which 1 - `alpha` confidence level to use for the confidence limits.
`na.action`	function to be called to handle missing values.
`set.panels`	logical; should the function set the panels on the device (TRUE) or not (FALSE).
`verbose`	logical; should progress information be printed to the screen?
`...`	optional arguments to `fevd`.

Details

Several GP or PP (or exponential) models are fit to `x` according to a range of `nint` thresholds given by `r`. The estimated parameters are plotted against these thresholds along with their associated (1 - `alpha`) * 100 percent CIs.

Choice of threshold is a compromise between low variance (lower thresholds yield more data with which to fit the models) and bias (higher thresholds yield estimates that are less biased because model assumptions require very high thresholds, and it can happen that lower data values may be more abundant causing the model to be biased toward the wrong values) in the parameter estimates. An appropriate threshold should yield the same parameter estimates (within uncertainty) as would be fit for any model fit to higher thresholds. Therefore, the idea is to find the lowest possible threshold whereby a higher threshold would give the same results within uncertainty bounds.

See Coles (2001) sec. 4.3.4 and 4.4 for more information.

Note that the default uses maximum likelihood estimation. While it is possible to use other methods, it is not recommended because of efficiency problems.

Value

A matrix of parameter values and CI bounds for each threshold value is returned invisibly. A plot is created.

Author(s)

Eric Gilleland

References

Coles, S. (2001). *An introduction to statistical modeling of extreme values*, London, United Kingdom: Springer-Verlag, 208 pp.

See Also

`fevd`, `mrlplot`

Examples

```
data(Fort)
threshrange.plot(Fort$Prec, r = c(1, 2.25), nint=5)

## Not run:
threshrange.plot(Fort$Prec, r=c(0.01,1), nint=30, verbose=TRUE)

threshrange.plot(Fort$Prec, r=c(0.2,0.8), type="PP", nint=15,
    verbose=TRUE)

threshrange.plot(Fort$Prec, r=c(0.2,0.8), type="PP", nint=15,
    optim.args=list(method="Nelder-Mead"), verbose=TRUE)


## End(Not run)
```

Tphap *Daily Maximum and Minimum Temperature in Phoenix, Arizona.*

Description

Daily maximum and minimum temperature (degrees Fahrenheit) for July through August 1948 through 1990 at Sky Harbor airport in Phoenix, Arizona.

Usage

```
data(Tphap)
```

Format

A data frame with 43 observations on the following 3 variables.

Year a numeric vector giving the number of years since 1900.

Month a numeric vector giving the month.

Day a numeric vector giving the day of the month.

MaxT a numeric vector giving the daily maximum temperatures in degrees Fahrenheit.

MinT a numeric vector giving the daily minimum temperatures in degrees Fahrenheit.

Details

Data are daily maximum and minimum temperature for the summer months of July through August from 1948 through 1990.

Source

U.S. National Weather Service Forecast office at the Phoenix Sky Harbor Airport.

References

Balling, R. C., Jr., Skindlov, J. A. and Phillips, D. H. (1990) The impact of increasing summer mean temperatures on extreme maximum and minimum temperatures in Phoenix, Arizona. *Journal of Climate*, **3**, 1491–1494.

Tarleton, L. F. and Katz, R. W. (1995) Statistical explanation for trends in extreme summer temperatures at Phoenix, A.Z. *Journal of Climate*, **8**, (6), 1704–1708.

Examples

```
data(Tphap)
str(Tphap)

par( mfrow=c(2,1))
hist( Tphap[,"MaxT"], main="", xlab="Max Temp", xlim=c(60,120), freq=FALSE, breaks="FD", col="red")
hist( Tphap[,"MinT"], main="", xlab="Min Temp", xlim=c(60,120), freq=FALSE, breaks="FD", col="blue")
```

trans *Transform Data*

Description

Method function to transform a data set. In the case of fevd objects, the transformation is to a standardized Gumbel or exponential scale.

Usage

```
trans(object, ...)

## S3 method for class 'fevd'
trans(object, ..., burn.in = 499, return.all = FALSE)
```

Arguments

object	An R object with a trans method. In the case of "fevd" objects, output from fevd.
burn.in	number giving the burn in value. The first 1:burn.in will not be used in obtaining parmaeter estiamtes.
return.all	logical, only for POT models, but primarily for use with the Point Process model. Should only the threshold exceedances be returned?
...	Not used.

Details

Many important situations occur in extreme value analysis (EVA) where it is useful or necessary to transform data to a standardized scale. For example, when investigating multivariate or conditional EVA much of the theory revolves around first transfroming the data to a unit scale. Further, for non-stationary models, it can be useful to transform the data to a df that does not depend on the covariates.

The present function transforms data taken from "fevd" class objects and transforms them to either a standard Gumbel (GEV, Gumbel case) or standard exponential (GP, PP, exponential case) df. In the first case, if the data are Gumbel distributed (really, if a gumbel fit was performed) the transformation is:

z = (x - location(yi))/scale(yi),

where yi represent possible covariate terms and z is distributed according to a Gumbel(0, 1) df. If the data are GEV distributed, then the transformation is:

z = - log(1 + (shape(yi)/scale(yi) * (x - location(yi)))^(-1/shape(yi))),

and again z is distributed Gumbel(0, 1).

In the case of exponentially distributed data, the transformation is:

z = (x - threshold(yi))/scale(yi)

and z is distributed according to an exponential(1) df.

For GP distributed data, the transformation is:

z = -log((1 + (shape(yi)/scale(yi) * (x - threshold(yi))))^(-1/shape(yi))

where again z follows an exponential(1) df.

For PP models, the transformation is:

z = (1 + shape(yi)/scale(yi) * (x - threshold(yi)))^(-1/shape(yi))

and z is distributed exponential(1).

See Coles (2001) sec. 2.3.2 for more details.

Value

numeric vector of transformed data.

Author(s)

Eric Gilleland

References

Coles, S. (2001) *An introduction to statistical modeling of extreme values*, London, U.K.: Springer-Verlag, 208 pp.

See Also

revtrans.evd, fevd

Examples

```
data(PORTw)

fit <- fevd(TMX1, PORTw, location.fun=~AOindex, units="deg C")
fit

z <- trans(fit)

fevd(z)
```

---

xbooter *Additional Bootstrap Functions for Univariate EVA*

Description

Additonal bootstrap capabilities for extreme-value analysis for fevd objects.

Usage

```
xbooter(x, B, rsize, block.length = 1,
return.period = c(10, 20, 50, 100, 200, 500),
qcov = NULL, qcov.base = NULL, shuffle = NULL,
replace = TRUE, verbose = FALSE, ...)
```

Arguments

x	list object of class "fevd"
B, rsize, block.length, shuffle, replace	See the help file for booter from the distillery package.
return.period	numeric value for the desired return period for which CIs are desired.
qcov	numeric matrix with rows the same length as q and columns equal to the number of parameters (+ 1 for the threshold, if a POT model). This gives any covariate values for a nonstationary model. If NULL, and model is non-stationary, only the intercept terms for modeled parameters are used, and if a non-constant threshold, only the first threshold value is used. Not used if model is stationary.
qcov.base	numeric matrix analogous to qcov. When provided, the function returns the difference in return levels between the level for the covariates in qcov and the level for covariates in qcov.base.
verbose	logical if TRUE progress information is printed to the screen.
...	Additonal optional arguments to the booter function.

Details

The ci method function will perform parametric bootstrapping for "fevd" objects, but this function is a wrapper to booter, which allows for greater flexibility with "fevd" objects. Gives CIs for the EVD parameters and return levels.

Value

Object of class "booted" is returned. See the help file for booter for more information.

Author(s)

Eric Gilleland

References

Gilleland, E. (2020) Bootstrap methods for statistical inference. Part I: Comparative forecast verification for continuous variables. *Journal of Atmospheric and Oceanic Technology*, **37** (11), 2117 - 2134, doi: 10.1175/JTECH-D-20-0069.1.

Gilleland, E. (2020) Bootstrap methods for statistical inference. Part II: Extreme-value analysis. *Journal of Atmospheric and Oceanic Technology*, **37** (11), 2135 - 2144, doi: 10.1175/JTECH-D-20-0070.1.

See Also

fevd, distillery::booter, xtibber, ci.fevd

Examples

```
set.seed( 409 )
z <- apply( matrix( rnorm( 100 * 1000 ), 1000, 100 ), 2, max )
fit <- fevd( z )

# In order to keep the code fast for CRAN compiling,
# a low value for B is used here, but should use a larger
# value in general.
bfit <- xbooter( fit, B = 50, verbose = TRUE )
ci( bfit, type = "perc" )
```

---

xtibber *Test-Inversion Bootstrap for Extreme-Value Analysis*

Description

Test-inversion bootstrap (TIB) for fevd class objects.

Usage

```
xtibber(x, type = c("return.level", "parameter"), which.one,
tib.method = c("interp", "rm"), nuisance = "shape", B,
test.pars, rsize, block.length = 1, shuffle = NULL,
replace = TRUE, alpha = 0.05, qcov = NULL,
qcov.base = NULL, stud = FALSE, step.size, tol = 1e-04,
max.iter = 1000, keep.iters = TRUE, verbose = FALSE, ...)
```

Arguments

x	List object of class "fevd".
type	character string stating whether to calculate TIB intervals for a return level or a parameter as this funciton will only calculate an interval for a single parameter/return level at a time.
which.one	number or character stating which return level or which parameter to find CIs for.
tib.method	character stating whether to estimate the TIB interval by interpolating from a series of pre-determined values of the nuisance parameter or to use the Robbins-Monroe (RM) method. See the help file for tibber from the distillery package for more information.
nuisance	character naming the nuisance parameter.
B, rsize, block.length, shuffle, replace	See the help file for booter from the distillery package for more information on these arguments.

test.pars	numeric vector giving the sequence of nuisance parameter values for the interpolation method, or a numeric vector of length two giving the starting values for the RM method.
alpha	numeric between zero and one giving the desired confidence level.
qcov	numeric matrix with rows the same length as q and columns equal to the number of parameters (+ 1 for the threshold, if a POT model). This gives any covariate values for a nonstationary model. If NULL, and model is non-stationary, only the intercept terms for modeled parameters are used, and if a non-constant threshold, only the first threshold value is used. Not used if model is stationary.
qcov.base	numeric matrix analogous to qcov. When provided, the function returns the difference in return levels between the level for the covariates in qcov and the level for covariates in qcov.base.
stud	logical if TRUE will calculate Studentized intervals (generally not profitable with the TIB method).
step.size	Used with the RM method only. Numeric giving the size of increments to use in the root-finding algorithm.
tol	Used with the RM method only. Numeric stating how close to the desired level of confidence is satisfactory.
max.iter	numeric giving the maximum number of iterations for the root-finding algorithm before giving up.
keep.iters	logical, should all of the values in the root-finding search be kept? Needed if a plot will be made.
verbose	logical, if TRUE will print progress information to the screen.
...	optional arguments to nlminb.

Details

This function provides a wrapper to the tibber function from distillery for "fevd" objects.

Value

See the help file for tibber for more information on the value

Author(s)

Eric Gilleland

References

Gilleland, E. (2020) Bootstrap methods for statistical inference. Part I: Comparative forecast verification for continuous variables. *Journal of Atmospheric and Oceanic Technology*, **37** (11), 2117 - 2134, doi: 10.1175/JTECH-D-20-0069.1.

Gilleland, E. (2020) Bootstrap methods for statistical inference. Part II: Extreme-value analysis. *Journal of Atmospheric and Oceanic Technology*, **37** (11), 2135 - 2144, doi: 10.1175/JTECH-D-20-0070.1.

See Also

fevd, distillery::tibber, distillery::booter

Examples

```
## Not run:
data("ftcanmax")
fit <- fevd( Prec, data = ftcanmax )

tbfit <- xtibber( fit, which.one = 100, B = 500,
 test.pars = seq(-0.01,0.2,,100), verbose = TRUE )

tbfit

plot( tbfit )


## End(Not run)
```

Index