Package ‘ismev’

October 13, 2022

Version 1.42

Date 2018-05-08

Title An Introduction to Statistical Modeling of Extreme Values

Author Original S functions written by Janet E. Heffernan with R port and R documentation provided by Alec G. Stephenson.

Maintainer Eric Gilleland <ericg@ucar.edu>

Depends R (>= 2.10.0), mgcv

Description Functions to support the computations carried out in `An Introduction to Statistical Modeling of Extreme Values' by Stuart Coles. The functions may be divided into the following groups; maxima/minima, order statistics, peaks over thresholds and point processes.

License GPL (>= 2)

URL http://www.ral.ucar.edu/~ericg/softextreme.php

NeedsCompilation no

Repository CRAN

Date/Publication 2018-05-10 22:08:29 UTC

R topics documented:

`dowjones` *Daily Closing Prices of The Dow Jones Index*

Description

The `dowjones` data frame has 1304 rows and 2 columns. The second column contains daily closing prices of the Dow Jones Index over the period 1996 to 2000. The first column contains a `POSIXct` object giving the dates of each observation.

Usage

```
data(dowjones)
```

Format

This data frame contains the following columns:

Date A `POSIXct` object containing dates.

Index A numeric vector containing daily closing prices of the Dow Jones Index.

Source

Coles, S. G. (2001) *An Introduction to Statistical Modelling of Extreme Values.* London: Springer.

`engine` *Engine Failure Time Data*

Description

The `engine` data frame has 32 rows and 2 columns. The first column contains the corrosion level, the second column gives the engine failure time.

Usage

```
data(engine)
```

Format

This data frame contains the following columns:

Time A numeric vector of corrosion levels.

Corrosion A numeric vector of failure times.

Source

Unknown.

`euroex` *UK/Euro Exchange Rates*

Description

A numeric vector of daily exchange rates between the Euro and UK sterling.

Usage

```
data(euroex)
```

Format

A vector containing 975 observations.

Source

Unknown.

`exchange` *UK/US and UK/Canada Exchange Rates*

Description

The `exchange` data frame has 975 rows and 2 columns. The columns contain daily exchange rates; UK sterling against the US dollar (first column) and UK sterling against the Canadian dollar (second column). The rownames contain the corresponding dates in a character string with the format `"2000/05/26"`. This can be converted into a `POSIXct` or `POSIXlt` object using `as.POSIXct` or `as.POSIXlt`.

Usage

```
data(exchange)
```

Format

This data frame contains the following columns:

USD.GBP US against UK exchange rate.

CAD.GBP Canada against UK exchange rate.

Source

Coles, S. G. (2001) *An Introduction to Statistical Modelling of Extreme Values.* London: Springer.

`fremantle` *Annual Maximum Sea Levels at Fremantle, Western Australia*

Description

The `fremantle` data frame has 86 rows and 3 columns. The second column gives 86 annual maximimum sea levels recorded at Fremantle, Western Australia, within the period 1897 to 1989. The first column gives the corresponding years. The third column gives annual mean values of the Southern Oscillation Index (SOI), which is a proxy for meteorological volitility.

Usage

```
data(fremantle)
```

Format

This data frame contains the following columns:

Year A numeric vector of years.

SeaLevel A numeric vector of annual sea level maxima.

SOI A numeric vector of annual mean values of the Southern Oscillation Index.

Source

Coles, S. G. (2001) *An Introduction to Statistical Modelling of Extreme Values.* London: Springer.

`gamGPDfitboot`	*Smooth Parameter Estimation and Bootstrapping of Generalized Pareto Distributions with Penalized Maximum Likelihood Estimation*

Description

`gamGPDfit()` fits the parameters of a generalized Pareto distribution (GPD) depending on covariates in a non- or semiparametric way.

`gamGPDboot()` fits and bootstraps the parameters of a GPD distribution depending on covariates in a non- or semiparametric way. Applies the post-blackend bootstrap of Chavez-Demoulin and Davison (2005).

Usage

```
gamGPDfit(x, threshold, nexc=NULL, datvar, xiFrhs, nuFrhs,
          init=gpd.fit(x[, datvar], threshold=threshold, show=FALSE)$mle[2:1],
          niter=32, include.updates=FALSE, epsxi=1e-05, epsnu=1e-05,
          progress=TRUE, verbose=FALSE, ...)
gamGPDboot(x, B, threshold, nexc=NULL, datvar, xiFrhs, nuFrhs,
           init=gpd.fit(x[, datvar], threshold=threshold, show=FALSE)$mle[2:1],
           niter=32, include.updates=FALSE, epsxi=1e-5, epsnu=1e-5,
           boot.progress=TRUE, progress=FALSE, verbose=FALSE, debug=FALSE, ...)
```

Arguments

`x`	data.frame containing the losses (in some component; can be specified with the argument `datvar`; the other components contain the covariates).
`B`	number of bootstrap replications.
`threshold`	threshold of the peaks-over-threshold (POT) method.
`nexc`	number of excesses. This can be used to determine
`datvar`	name of the data column in `x` which contains the the data to be modeled.
`xiFrhs`	right-hand side of the formula for ξ in the `gam()` call for fitting ξ.
`nuFrhs`	right-hand side of the formula for ν in the `gam()` call for fitting ν.
`init`	bivariate vector containing initial values for (ξ, β).
`niter`	maximal number of iterations in the backfitting algorithm.
`include.updates`	`logical` indicating whether updates for xi and nu are returned as well (note: this might lead to objects of large size).
`epsxi`	epsilon for stop criterion for ξ.
`epsnu`	epsilon for stop criterion for ν.

`boot.progress`	`logical` indicating whether progress information about `gamGPDboot()` is displayed.
`progress`	`logical` indicating whether progress information about `gamGPDfit()` is displayed. For `gamGPDboot()`, `progress` is only passed to `gamGPDfit()` in the case that `boot.progress==TRUE`.
`verbose`	`logical` indicating whether additional information (in case of undesired behavior) is printed. For `gamGPDboot()`, `progress` is only passed to `gamGPDfit()` if `boot.progress==TRUE`.
`debug`	`logical` indicating whether initial fit (before the bootstrap is initiated) is saved.
`...`	additional arguments passed to `gam()` (which is called internally; see the source code of `gamGPDfitUp()`).

Details

`gamGPDfit()` fits the parameters ξ and β of the generalized Pareto distribution $\mathrm{GPD}(\xi,\beta)$ depending on covariates in a non- or semiparametric way. The distribution function is given by

$$G_{\xi,\beta}(x) = 1 - (1+\xi x/\beta)^{-1/\xi}, \quad x \ge 0,$$

for $\xi > 0$ (which is what we assume) and $\beta > 0$. Note that β is also denoted by σ in this package. Estimation of ξ and β by `gamGPDfit()` is done via penalized maximum likelihood estimation, where the estimators are computed with a backfitting algorithm. In order to guarantee convergence of this algorithm, a reparameterization of β in terms of the parameter ν is done via

$$\beta = \exp(\nu)/(1+\xi).$$

The parameters ξ and ν (and thus β) are allowed to depend on covariates (including time) in a non- or semiparametric way, for example:

$$\xi = \xi(\boldsymbol{x}, t) = \boldsymbol{x}^\top \boldsymbol{\alpha}_\xi + h_\xi(t),$$
$$\nu = \nu(\boldsymbol{x}, t) = \boldsymbol{x}^\top \boldsymbol{\alpha}_\nu + h_\nu(t),$$

where $\boldsymbol{x}$ denotes the vector of covariates, $\boldsymbol{\alpha}_\xi$, $\boldsymbol{\alpha}_\nu$ are parameter vectors and h_ξ, h_ν are regression splines. For more details, see the references and the source code.

`gamGPDboot()` first fits the GPD parameters via `gamGPDfit()`. It then conducts the post-blackend bootstrap of Chavez-Demoulin and Davison (2005). To this end, it computes the residuals, resamples them (`B` times), reconstructs the corresponding excesses, and refits the GPD parameters via `gamGPDfit()` again.

Value

`gamGPDfit()` returns a list with the components

`xi`: estimated parameters ξ;

`beta`: estimated parameters β;

`nu`: estimated parameters ν;

`se.xi`: standard error for ξ ((possibly adjusted) second-order derivative of the reparameterized log-likelihood with respect to ξ) multiplied by -1;

`se.nu`**:** standard error for ν ((possibly adjusted) second-order derivative of the reparameterized log-likelihood with respect to ν) multiplied by -1;

`xi.covar`**:** (unique) covariates for ξ;

`nu.covar`**:** (unique) covariates for ν;

`covar`**:** available covariate combinations used for fitting $\beta(\xi, \nu)$;

`y`**:** vector of excesses (exceedances minus threshold);

`res`**:** residuals;

`MRD`**:** mean relative distances between for all iterations, calculated between old parameters (ξ, ν) (from the last iteration) and new parameters (currently estimated ones);

`logL`**:** log-likelihood at the estimated parameters;

`xiObj`**:** R object of type `gamObject` for estimated ξ (returned by `mgcv::gam()`);

`nuObj`**:** R object of type `gamObject` for estimated ν (returned by `mgcv::gam()`);

`xiUpdates`**:** if `include.updates` is `TRUE`, updates for ξ for each iteration. This is a list of R objects of type `gamObject` which contains `xiObj` as last element;

`nuUpdates`**:** if `include.updates` is `TRUE`, updates for ν for each iteration. This is a list of R objects of type `gamObject` which contains `nuObj` as last element;

`gamGPDboot()` returns a list of length `B+1` where the first component contains the results of the initial fit via `gamGPDfit()` and the other `B` components contain the results for each replication of the post-blackend bootstrap.

Author(s)

Marius Hofert, Valerie Chavez-Demoulin.

References

Chavez-Demoulin, V., and Davison, A. C. (2005), Generalized additive models for sample extremes, *Applied Statistics* **54**(1), 207–222.

Chavez-Demoulin, V., and Hofert, M. (to be submitted), Smooth extremal models fitted by penalized maximum likelihood estimation.

Examples

```
### Example 1: fitting capability ###############################################

## generate an example data set
years <- 2003:2012 # years
nyears <- length(years)
n <- 250 # sample size for each (different) xi
u <- 200 # threshold
rGPD <- function(n, xi, beta) ((1-runif(n))^(-xi)-1)*beta/xi # sampling GPD

set.seed(17) # setting seed
xi.true.A <- seq(0.4, 0.8, length=nyears) # true xi for group "A"
## generate losses for group "A"
lossA <- unlist(lapply(1:nyears,
                       function(y) u + rGPD(n, xi=xi.true.A[y], beta=1)))
xi.true.B <- xi.true.A^2 # true xi for group "B"
## generate losses for group "B"
lossB <- unlist(lapply(1:nyears,
                       function(y) u + rGPD(n, xi=xi.true.B[y], beta=1)))
## build data frame
time <- rep(rep(years, each=n), 2) # "2" stands for the two groups
covar <- rep(c("A","B"), each=n*nyears)
value <- c(lossA, lossB)
x <- data.frame(covar=covar, time=time, value=value)

## fit
eps <- 1e-3 # to decrease the run time for this example
fit <- gamGPDfit(x, threshold=u, datvar="value", xiFrhs=~covar+s(time)-1,
                 nuFrhs=~covar+s(time)-1, epsxi=eps, epsnu=eps)
## note: choosing s(..., bs="cr") will fit cubic splines

## grab the fitted values per group and year
xi.fit <- fitted(fit$xiObj)
xi.fit. <- xi.fit[1+(0:(2*nyears-1))*n] # pick fit for each group and year
xi.fit.A <- xi.fit.[1:nyears] # fit for "A" and each year
xi.fit.B <- xi.fit.[(nyears+1):(2*nyears)] # fit for "B" and each year

## plot the fitted values of xi and the true ones we simulated from
par(mfrow=c(1,2))
plot(years, xi.true.A, type="l", ylim=range(xi.true.A, xi.fit.A),
     main="Group A", xlab="Year", ylab=expression(xi))
points(years, xi.fit.A, type="l", col="red")
legend("topleft", inset=0.04, lty=1, col=c("black", "red"),
       legend=c("true", "fitted"), bty="n")
plot(years, xi.true.B, type="l", ylim=range(xi.true.B, xi.fit.B),
     main="Group B", xlab="Year", ylab=expression(xi))
points(years, xi.fit.B, type="l", col="blue")
legend("topleft", inset=0.04, lty=1, col=c("black", "blue"),
       legend=c("true", "fitted"), bty="n")

## Not run:
### Example 2: Comparison of (the more general) gamGPDfit() with gpd.fit() ########

set.seed(17) # setting seed
xi.true.A <- rep(0.4, length=nyears)
xi.true.B <- rep(0.8, length=nyears)
## generate losses for group "A"
lossA <- unlist(lapply(1:nyears,
                       function(y) u + rGPD(n, xi=xi.true.A[y], beta=1)))
## generate losses for group "B"
lossB <- unlist(lapply(1:nyears,
                       function(y) u + rGPD(n, xi=xi.true.B[y], beta=1)))
## build data frame
x <- data.frame(covar=covar, time=time, value=c(lossA, lossB))

## fit with gpd.fit
fit.coles <- gpd.fit(x$value, threshold=u, shl=1, sigl=1, ydat=x)
```

```
xi.fit.coles.A <- fit.coles$mle[3]+1*fit.coles$mle[4]
xi.fit.coles.B <- fit.coles$mle[3]+2*fit.coles$mle[4]

## fit with gamGPDfit()
fit <- gamGPDfit(x, threshold=u, datvar="value", xiFrhs=~covar, nuFrhs=~covar,
                 epsxi=eps, epsnu=eps)
xi.fit <- fitted(fit$xiObj)
xi.fit.A <- as.numeric(xi.fit[1]) # fit for group "A"
xi.fit.B <- as.numeric(xi.fit[nyears*n+1]) # fit for group "B"

## comparison
xi.fit.A-xi.fit.coles.A
xi.fit.B-xi.fit.coles.B

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

`gev.diag` *Diagnostic Plots for GEV Models*

Description

Produces diagnostic plots for GEV models using the output of the function `gev.fit`.

Usage

```
gev.diag(z)
```

Arguments

`z`	An object returned by `gev.fit`.

Value

For stationary models four plots are produced; a probability plot, a quantile plot, a return level plot and a histogram of data with fitted density.

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

See Also

`gev.fit`, `gev.prof`

Examples

```
data(portpirie)
ppfit <- gev.fit(portpirie[,2])
gev.diag(ppfit)
```

gev.fit — *Maximum-likelihood Fitting of the GEV Distribution*

Description

Maximum-likelihood fitting for the generalized extreme value distribution, including generalized linear modelling of each parameter.

Usage

```
gev.fit(xdat, ydat = NULL, mul = NULL, sigl = NULL, shl = NULL,
    mulink = identity, siglink = identity, shlink = identity,
    muinit = NULL, siginit = NULL, shinit = NULL,
    show = TRUE, method = "Nelder-Mead", maxit = 10000, ...)
```

Arguments

`xdat`	A numeric vector of data to be fitted.
`ydat`	A matrix of covariates for generalized linear modelling of the parameters (or `NULL` (the default) for stationary fitting). The number of rows should be the same as the length of `xdat`.
`mul, sigl, shl`	Numeric vectors of integers, giving the columns of `ydat` that contain covariates for generalized linear modelling of the location, scale and shape parameters repectively (or `NULL` (the default) if the corresponding parameter is stationary).
`mulink, siglink, shlink`	Inverse link functions for generalized linear modelling of the location, scale and shape parameters repectively.
`muinit, siginit, shinit`	numeric of length equal to total number of parameters used to model the location, scale or shape parameter(s), resp. See Details section for default (NULL) initial values.
`show`	Logical; if `TRUE` (the default), print details of the fit.
`method`	The optimization method (see `optim` for details).
`maxit`	The maximum number of iterations.
`...`	Other control parameters for the optimization. These are passed to components of the `control` argument of `optim`.

Details

The form of the GEV used is that of Coles (2001) Eq (3.2). Specifically, positive values of the shape parameter imply a heavy tail, and negative values imply a bounded upper tail.

For non-stationary fitting it is recommended that the covariates within the generalized linear models are (at least approximately) centered and scaled (i.e.\ the columns of `ydat` should be approximately centered and scaled).

Let m=mean(xdat) and s=sqrt(6*var(xdat))/pi. Then, initial values assigend when 'muinit' is NULL are m - 0.57722 * s (stationary case). When 'siginit' is NULL, the initial value is taken to be s, and when 'shinit' is NULL, the initial value is taken to be 0.1. When covariates are introduced (non-stationary case), these same initial values are used by default for the constant term, and zeros for all other terms. For example, if a GEV(mu(t)=mu0+mu1*t, sigma, xi) is being fitted, then the initial value for mu0 is m - 0.57722 * s, and 0 for mu1.

Value

A list containing the following components. A subset of these components are printed after the fit. If `show` is `TRUE`, then assuming that successful convergence is indicated, the components `nllh`, `mle` and `se` are always printed.

`nllh`	single numeric giving the negative log-likelihood value.
`mle`	numeric vector giving the MLE's for the location, scale and shape parameters, resp.
`se`	numeric vector giving the standard errors for the MLE's for the location, scale and shape parameters, resp.
`trans`	An logical indicator for a non-stationary fit.
`model`	A list with components `mul`, `sigl` and `shl`.
`link`	A character vector giving inverse link functions.
`conv`	The convergence code, taken from the list returned by `optim`. A zero indicates successful convergence.
`nllh`	The negative logarithm of the likelihood evaluated at the maximum likelihood estimates.
`data`	The data that has been fitted. For non-stationary models, the data is standardized.
`mle`	A vector containing the maximum likelihood estimates.
`cov`	The covariance matrix.
`se`	A vector containing the standard errors.
`vals`	A matrix with three columns containing the maximum likelihood estimates of the location, scale and shape parameters at each data point.

References

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

See Also

`gev.diag`, `optim`, `gev.prof`

Examples

```
data(portpirie)
gev.fit(portpirie[,2])
```

gev.prof — *Profile Log-likelihoods for Stationary GEV Models*

Description

Produce profile log-likelihoods for shape parameters and m year/block return levels for stationary GEV models using the output of the function `gev.fit`.

Usage

```
gev.prof(z, m, xlow, xup, conf = 0.95, nint = 100)
gev.profxi(z, xlow, xup, conf = 0.95, nint = 100)
```

Arguments

`z`	An object returned by `gev.fit`. The object should represent a stationary model.
`m`	The return level (i.e.\ the profile likelihood is for the value that is exceeded with probability 1/`m`).
`xlow, xup`	The least and greatest value at which to evaluate the profile likelihood.
`conf`	The confidence coefficient of the plotted profile confidence interval.
`nint`	The number of points at which the profile likelihood is evaluated.

Value

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

See Also

`gev.fit`, `gev.diag`

Examples

```
data(portpirie)
ppfit <- gev.fit(portpirie[,2])
## Not run: gev.prof(ppfit, m = 10, 4.1, 5)
## Not run: gev.profxi(ppfit, -0.3, 0.3)
```

glass *Breaking Strengths of Glass Fibres*

Description

A numeric vector containing breaking strengths of 63 glass fibres of length 1.5 centimetres, recorded under experimental conditions.

Usage

```
data(glass)
```

Format

A vector containing 63 observations.

Source

Smith, R. L. and Naylor, J. C. (1987) A comparison of maximum likelihood and Bayesian estimators for the three-parameter Weibull distribution. *Applied Statistics* **36**, 358–396.

References

Coles, S. G. (2001) *An Introduction to Statistical Modelling of Extreme Values.* London: Springer.

gpd.diag *Diagnostic Plots for GPD Models*

Description

Produces diagnostic plots for GPD models using the output of the function `gpd.fit`.

Usage

```
gpd.diag(z)
```

Arguments

`z`	An object returned by `gpd.fit`.

Value

For stationary models four plots are produced; a probability plot, a quantile plot, a return level plot and a histogram of data with fitted density.

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

See Also

gpd.fit, gpd.prof, pp.fit

Examples

```
data(rain)
rnfit <- gpd.fit(rain, 10)
gpd.diag(rnfit)
```

---

gpd.fit *Maximum-likelihood Fitting for the GPD Model*

Description

Maximum-likelihood fitting for the GPD model, including generalized linear modelling of each parameter.

Usage

```
gpd.fit(xdat, threshold, npy = 365, ydat = NULL, sigl = NULL,
    shl = NULL, siglink = identity, shlink = identity, siginit = NULL,
    shinit = NULL, show = TRUE,
    method = "Nelder-Mead", maxit = 10000, ...)
```

Arguments

`xdat`	A numeric vector of data to be fitted.
`threshold`	The threshold; a single number or a numeric vector of the same length as `xdat`.
`npy`	The number of observations per year/block.
`ydat`	A matrix of covariates for generalized linear modelling of the parameters (or `NULL` (the default) for stationary fitting). The number of rows should be the same as the length of `xdat`.
`sigl, shl`	Numeric vectors of integers, giving the columns of `ydat` that contain covariates for generalized linear modelling of the scale and shape parameters repectively (or `NULL` (the default) if the corresponding parameter is stationary).
`siglink, shlink`	Inverse link functions for generalized linear modelling of the scale and shape parameters repectively.
`siginit, shinit`	numeric giving initial value(s) for parameter estimates. If NULL, default is sqrt(6 * var(xdat))/pi and 0.1 for the scale and shape parameters, resp. If using parameter covariates, then these values are used for the constant term, and zeros for all other terms.
`show`	Logical; if `TRUE` (the default), print details of the fit.
`method`	The optimization method (see `optim` for details).

`maxit`	The maximum number of iterations.
`...`	Other control parameters for the optimization. These are passed to components of the `control` argument of `optim`.

Details

For non-stationary fitting it is recommended that the covariates within the generalized linear models are (at least approximately) centered and scaled (i.e.\ the columns of `ydat` should be approximately centered and scaled).

The form of the GP model used follows Coles (2001) Eq (4.7). In particular, the shape parameter is defined so that positive values imply a heavy tail and negative values imply a bounded upper value.

Value

A list containing the following components. A subset of these components are printed after the fit. If `show` is `TRUE`, then assuming that successful convergence is indicated, the components `nexc`, `nllh`, `mle`, `rate` and `se` are always printed.

`nexc`	single numeric giving the number of threshold exceedances.
`nllh`	nsingle umeric giving the negative log-likelihood value.
`mle`	numeric vector giving the MLE's for the scale and shape parameters, resp.
`rate`	single numeric giving the estimated probability of exceeding the threshold.
`se`	numeric vector giving the standard error estiamtes for the scale and shape parameter estimates, resp.
`trans`	An logical indicator for a non-stationary fit.
`model`	A list with components `sigl` and `shl`.
`link`	A character vector giving inverse link functions.
`threshold`	The threshold, or vector of thresholds.
`nexc`	The number of data points above the threshold.
`data`	The data that lie above the threshold. For non-stationary models, the data is standardized.
`conv`	The convergence code, taken from the list returned by `optim`. A zero indicates successful convergence.
`nllh`	The negative logarithm of the likelihood evaluated at the maximum likelihood estimates.
`vals`	A matrix with three columns containing the maximum likelihood estimates of the scale and shape parameters, and the threshold, at each data point.
`mle`	A vector containing the maximum likelihood estimates.
`rate`	The proportion of data points that lie above the threshold.
`cov`	The covariance matrix.
`se`	A vector containing the standard errors.
`n`	The number of data points (i.e.\ the length of `xdat`).
`npy`	The number of observations per year/block.
`xdata`	The data that has been fitted.

References

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

See Also

gpd.diag, optim, gpd.prof, gpd.fitrange, mrl.plot, pp.fit

Examples

```
data(rain)
gpd.fit(rain, 10)
```

---

gpd.fitrange *Fitting the GPD Model Over a Range of Thresholds*

Description

Maximum-likelihood fitting for a stationary GPD model, over a range of thresholds. Graphs of parameter estimates which aid the selection of a threshold are produced.

Usage

```
gpd.fitrange(data, umin, umax, nint = 10, show = FALSE, ...)
```

Arguments

`data`	A numeric vector of data to be fitted.
`umin, umax`	The minimum and maximum thresholds at which the model is fitted.
`nint`	The number of fitted models.
`show`	Logical; if `TRUE`, print details of each fit.
`...`	Optional arguments to `gpd.fit`.

Value

Two graphs showing maximum likelihood estimates and confidence intervals of the shape and modified scale parameters over a range of thresholds are produced. A list object is returned invisibly with components: 'threshold' numeric vector of length 'nint' giving the thresholds used, 'mle' an 'nint X 3' matrix giving the maximum likelihood parameter estimates (columns are location, scale and shape respectively), 'se' an 'nint X 3' matrix giving the estimated standard errors for the parameter estimates (columns are location, scale and shape, resp.), 'ci.low', 'ci.up' 'nint X 3' matrices giving the lower and upper 95 intervals, resp. (columns same as for 'mle' and 'se').

See Also

gpd.fit, mrl.plot, pp.fit, pp.fitrange

Examples

```
## Not run: data(rain)
## Not run: gpd.fitrange(rain, 10, 40)
```

`gpd.prof` *Profile Log-likelihoods for Stationary GPD Models*

Description

Produce profile log-likelihoods for shape parameters and m year/block return levels for stationary GPD models using the output of the function `gpd.fit`.

Usage

```
gpd.prof(z, m, xlow, xup, npy = 365, conf = 0.95, nint = 100)
gpd.profxi(z, xlow, xup, conf = 0.95, nint = 100)
```

Arguments

`z`	An object returned by `gpd.fit`. The object should represent a stationary model.
`m`	The return level (i.e.\ the profile likelihood is for the value that is exceeded with probability 1/`m`).
`xlow, xup`	The least and greatest value at which to evaluate the profile likelihood.
`npy`	The number of observations per year.
`conf`	The confidence coefficient of the plotted profile confidence interval.
`nint`	The number of points at which the profile likelihood is evaluated.

Value

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

See Also

`gpd.fit`, `gpd.diag`

Examples

```
data(rain)
rnfit <- gpd.fit(rain, 10)
## Not run: gpd.prof(rnfit, m = 10, 55, 75)
## Not run: gpd.profxi(rnfit, -0.02, 0.15)
```

`gum.diag` *Diagnostic Plots for Gumbel Models*

Description

Produces diagnostic plots for Gumbel models using the output of the function `gum.fit`.

Usage

```
gum.diag(z)
```

Arguments

`z`	An object returned by `gum.fit`.

Value

For stationary models four plots are produced; a probability plot, a quantile plot, a return level plot and a histogram of data with fitted density.

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

See Also

`gev.fit`, `gum.fit`

Examples

```
data(portpirie)
ppfit <- gum.fit(portpirie[,2])
gum.diag(ppfit)
```

`gum.fit` *Maximum-likelihood Fitting of the Gumbel Distribution*

Description

Maximum-likelihood fitting for the gumbel distribution, including generalized linear modelling of each parameter.

Usage

```
gum.fit(xdat, ydat = NULL, mul = NULL, sigl = NULL, mulink = identity,
    siglink = identity, muinit = NULL, siginit = NULL, show = TRUE,
    method = "Nelder-Mead", maxit = 10000, ...)
```

Arguments

`xdat`	A numeric vector of data to be fitted.
`ydat`	A matrix of covariates for generalized linear modelling of the parameters (or `NULL` (the default) for stationary fitting). The number of rows should be the same as the length of `xdat`.
`mul, sigl`	Numeric vectors of integers, giving the columns of `ydat` that contain covariates for generalized linear modelling of the location and scale parameters repectively (or `NULL` (the default) if the corresponding parameter is stationary).
`mulink, siglink`	Inverse link functions for generalized linear modelling of the location and scale parameters repectively.
`muinit, siginit`	numeric giving initial parameter estimates. See Details section for information about default values (NULL).
`show`	Logical; if `TRUE` (the default), print details of the fit.
`method`	The optimization method (see `optim` for details).
`maxit`	The maximum number of iterations.
`...`	Other control parameters for the optimization. These are passed to components of the `control` argument of `optim`.

Details

For non-stationary fitting it is recommended that the covariates within the generalized linear models are (at least approximately) centered and scaled (i.e.\ the columns of `ydat` should be approximately centered and scaled).

Let m=mean(xdat) and s=sqrt(6*var(xdat))/pi. Then, initial values assigend when 'muinit' is NULL are m - 0.57722 * s (stationary case). When 'siginit' is NULL, the initial value is taken to be s, and when 'shinit' is NULL. When covariates are introduced (non-stationary case), these same initial values are used by default for the constant term, and zeros for all other terms. For example, if a Gumbel(mu(t)=mu0+mu1*t, sigma) is being fitted, then the initial value for mu0 is m - 0.57722 * s, and 0 for mu1.

Value

A list containing the following components. A subset of these components are printed after the fit. If `show` is `TRUE`, then assuming that successful convergence is indicated, the components `nllh`, `mle` and `se` are always printed.

`trans`	An logical indicator for a non-stationary fit.
`model`	A list with components `mul` and `sigl`.
`link`	A character vector giving inverse link functions.
`conv`	The convergence code, taken from the list returned by `optim`. A zero indicates successful convergence.
`nllh`	The negative logarithm of the likelihood evaluated at the maximum likelihood estimates.

`data`	The data that has been fitted. For non-stationary models, the data is standardized.
`mle`	A vector containing the maximum likelihood estimates.
`cov`	The covariance matrix.
`se`	A vector containing the standard errors.
`vals`	A matrix with two columns containing the maximum likelihood estimates of the location and scale parameters at each data point.

See Also

`gum.diag`, `optim`, `gev.fit`

Examples

```
data(portpirie)
gum.fit(portpirie[,2])
```

`ismev` *ismev – an Introduction to Statistical Modeling of Extreme Values*

Description

ismev includes functions to support the computations carried out in Coles (2001). The functions may be divided into the following groups; maxima/minima, order statistics, peaks over thresholds and point processes. **ismev** is an R port of the S-Plus extreme value statistical routines believed to be originally written by Janet E. Heffernan.

Primary functions include:

`gev.fit`, `gev.diag`, `gpd.fit`, `gpd.diag`, `pp.fit` and `pp.diag`.

Original R port was carried out by Alec G. Stephenson, and the package is currently being maintained by Eric Gilleland.

Datasets from Coles (2001) included are:

dowjones euroex fremantle portpirie venice wind engine exchange glass rain wavesurge wooster

References

Coles, Stuart (2001) *An Introduction to Statistical Modeling of Extreme Values*, London, UK: Springer, ISBN: 1852334592, 208 pp.

`mrl.plot` *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

```
mrl.plot(data, umin = min(data), umax = max(data) - 0.1,
    conf = 0.95, nint = 100)
```

Arguments

`data`	A numeric vector of data to be fitted.
`umin, umax`	The minimum and maximum thresholds at which the mean residual life function is calculated.
`conf`	The confidence coefficient for the confidence intervals depicted in the plot.
`nint`	The number of points at which the mean residual life function is calculated.

See Also

`gpd.fit`, `gpd.fitrange`, `pp.fit`

Examples

```
data(rain)
mrl.plot(rain)
```

`portpirie` *Annual Maximum Sea Levels at Port Pirie, South Australia*

Description

The `portpirie` data frame has 65 rows and 2 columns. The second column gives annual maximimum sea levels recorded at Port Pirie, South Australia, from 1923 to 1987. The first column gives the corresponding years.

Usage

```
data(portpirie)
```

Format

This data frame contains the following columns:

Year A numeric vector of years.

SeaLevel A numeric vector of annual sea level maxima.

Source

Coles, S. G. (2001) *An Introduction to Statistical Modelling of Extreme Values.* London: Springer.

`pp.diag` *Diagnostic Plots for Point Process Models*

Description

Produces diagnostic plots for point process models using the output of the function `pp.fit`.

Usage

```
pp.diag(z)
```

Arguments

`z`	An object returned by `pp.fit`.

Value

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

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

See Also

`pp.fit`, `gpd.fit`

Examples

```
data(rain)
rnfit <- pp.fit(rain, 10)
pp.diag(rnfit)
```

pp.fit *Maximum-likelihood Fitting for the Point Process Model*

Description

Maximum-likelihood fitting for the point process model, including generalized linear modelling of each parameter.

Usage

```
pp.fit(xdat, threshold, npy = 365, ydat = NULL, mul = NULL, sigl =
    NULL, shl = NULL, mulink = identity, siglink = identity, shlink =
    identity, muinit = NULL, siginit = NULL, shinit = NULL, show = TRUE,
    method = "Nelder-Mead", maxit = 10000, ...)
```

Arguments

`xdat`	A numeric vector of data to be fitted.
`threshold`	The threshold; a single number or a numeric vector of the same length as `xdat`.
`npy`	The number of observations per year/block.
`ydat`	A matrix of covariates for generalized linear modelling of the parameters (or `NULL` (the default) for stationary fitting). The number of rows should be the same as the length of `xdat`.
`mul, sigl, shl`	Numeric vectors of integers, giving the columns of `ydat` that contain covariates for generalized linear modelling of the location, scale and shape parameters repectively (or `NULL` (the default) if the corresponding parameter is stationary).
`mulink, siglink, shlink`	Inverse link functions for generalized linear modelling of the location, scale and shape parameters repectively.
`muinit, siginit, shinit`	numeric giving initial parameter estimates. See Details section for information on default (NULL) initial values.
`show`	Logical; if `TRUE` (the default), print details of the fit.
`method`	The optimization method (see `optim` for details).
`maxit`	The maximum number of iterations.
`...`	Other control parameters for the optimization. These are passed to components of the `control` argument of `optim`.

Details

For non-stationary fitting it is recommended that the covariates within the generalized linear models are (at least approximately) centered and scaled (i.e.\ the columns of `ydat` should be approximately centered and scaled). Otherwise, the numerics may become unstable.

As of version 1.32, a more accurate estimate of the exceedance rate, in the face of covariates, is used (at the expense of computational efficiency). In particular, when including covariates, parameter estimates may differ from those in Coles (2001).

Let m=mean(xdat) and s=sqrt(6*var(xdat))/pi. Then, initial values assigend when 'muinit' is NULL are m - 0.57722 * s (stationary case). When 'siginit' is NULL, the initial value is taken to be s, and when 'shinit' is NULL, the initial value is taken to be 0.1. When covariates are introduced (non-stationary case), these same initial values are used by default for the constant term, and zeros for all other terms. For example, if a GEV(mu(t)=mu0+mu1*t, sigma, xi) is being fitted, then the initial value for mu0 is m - 0.57722 * s, and 0 for mu1.

Value

A list containing the following components. A subset of these components are printed after the fit. If `show` is `TRUE`, then assuming that successful convergence is indicated, the components `nexc`, `nllh`, `mle` and `se` are always printed.

`trans`	An logical indicator for a non-stationary fit.
`model`	A list with components `mul`, `sigl` and `shl`.
`link`	A character vector giving inverse link functions.
`threshold`	The threshold, or vector of thresholds.
`npy`	The number of observations per year/block.
`nexc`	The number of data points above the threshold.
`data`	The data that lie above the threshold. For non-stationary models, the data is standardized.
`conv`	The convergence code, taken from the list returned by `optim`. A zero indicates successful convergence.
`nllh`	The negative logarithm of the likelihood evaluated at the maximum likelihood estimates.
`vals`	A matrix with four columns containing the maximum likelihood estimates of the location, scale and shape parameters, and the threshold, at each data point.
`gpd`	A matrix with three rows containing the maximum likelihood estimates of corresponding GPD location, scale and shape parameters at each data point.
`mle`	A vector containing the maximum likelihood estimates.
`cov`	The covariance matrix.
`se`	A vector containing the standard errors.

Warning

Different optimization methods may result in wildly different parameter estimates.

Note

This code is adapted by Eric Gilleland from code originally written for S-Plus by Stuart Coles, and ported to R by Alec Stephenson. See details section above.

References

Beirlant J, Goegebeur Y, Segers J and Teugels J. (2004). Statistics of Extremes, Wiley, Chichester, England.

Coles, Stuart (2001). An Introduction to Statistical Modeling of Extreme Values. Springer-Verlag, London.

See Also

`pp.diag`, `optim`, `pp.fitrange`, `mrl.plot`, `gpd.fit`

Examples

```
data(rain)
pp.fit(rain, 10)
```

`pp.fitrange` *Fitting the Point Process Model Over a Range of Thresholds*

Description

Maximum-likelihood fitting for a stationary point process model, over a range of thresholds. Graphs of parameter estimates which aid the selection of a threshold are produced.

Usage

```
pp.fitrange(data, umin, umax, npy = 365, nint = 10, show = FALSE, ...)
```

Arguments

`data`	A numeric vector of data to be fitted.
`umin, umax`	The minimum and maximum thresholds at which the model is fitted.
`npy`	The number of observations per year/block.
`nint`	The number of fitted models.
`show`	Logical; if `TRUE`, print details of each fit.
`...`	Optional arguments to `pp.fit`.

Value

Three graphs showing maximum likelihood estimates and confidence intervals of the location, scale and shape parameters over a range of thresholds are produced. A list object is returned invisibly with components: 'threshold' numeric vector of length 'nint' giving the thresholds used, 'mle' an 'nint X 3' matrix giving the maximum likelihood parameter estimates (columns are location, scale and shape respectively), 'se' an 'nint X 3' matrix giving the estimated standard errors for the parameter estimates (columns are location, scale and shape, resp.), 'ci.low', 'ci.up' 'nint X 3' matrices giving the lower and upper 95 intervals, resp. (columns same as for 'mle' and 'se').

See Also

pp.fit, mrl.plot, gpd.fit, gpd.fitrange

Examples

```
## Not run: data(rain)
## Not run: pp.fitrange(rain, 10, 40)
```

---

rain — *Daily Rainfall Accumulations in South-West England*

Description

A numeric vector containing daily rainfall accumulations at a location in south-west England over the period 1914 to 1962.

Usage

```
data(rain)
```

Format

A vector containing 17531 observations.

Source

Coles, S. G. and Tawn, J. A. (1996) Modelling extremes of the areal rainfall process. *Journal of the Royal Statistical Society, B* **53**, 329–347.

References

Coles, S. G. (2001) *An Introduction to Statistical Modelling of Extreme Values.* London: Springer.

---

rlarg.diag — *Diagnostic Plots for Order Statistics Models*

Description

Produces diagnostic plots for order statistics models using the output of the function `rlarg.fit`.

Usage

```
rlarg.diag(z, n = z$r)
```

Arguments

`z`	An object returned by `rlarg.fit`.
`n`	Probability and quantile plots are produced for the largest `n` order statistics.

Value

For stationary models four plots are initially produced; a probability plot, a quantile plot, a return level plot and a histogram of data with fitted density. Then probability and quantile plots are produced for the largest `n` order statistics.

For non-stationary models residual probability plots and residual quantile plots are produced for the largest `n` order statistics.

See Also

`rlarg.fit`

Examples

```
## Not run: data(venice)
## Not run: venfit <- rlarg.fit(venice[,-1])
## Not run: rlarg.diag(venfit)
```

`rlarg.fit` *Maximum-likelihood Fitting of Order Statistics Model*

Description

Maximum-likelihood fitting for the order statistic model, including generalized linear modelling of each parameter.

Usage

```
rlarg.fit(xdat, r = dim(xdat)[2], ydat = NULL, mul = NULL, sigl = NULL,
  shl = NULL, mulink = identity, siglink = identity, shlink = identity,
  muinit = NULL, siginit = NULL, shinit = NULL, show = TRUE,
  method = "Nelder-Mead", maxit = 10000, ...)
```

Arguments

`xdat`	A numeric matrix of data to be fitted. Each row should be a vector of decreasing order, containing the largest order statistics for each year (or time period). The first column therefore contains annual (or period) maxima. Only the first `r` columns are used for the fitted model. By default, all columns are used. If one year (or time period) contains fewer order statistics than another, missing values can be appended to the end of the corresponding row.
`r`	The largest `r` order statistics are used for the fitted model.

`ydat`	A matrix of covariates for generalized linear modelling of the parameters (or `NULL` (the default) for stationary fitting). The number of rows should be the same as the number of rows of `xdat`.
`mul, sigl, shl`	Numeric vectors of integers, giving the columns of `ydat` that contain covariates for generalized linear modelling of the location, scale and shape parameters repectively (or `NULL` (the default) if the corresponding parameter is stationary).
`mulink, siglink, shlink`	Inverse link functions for generalized linear modelling of the location, scale and shape parameters repectively.
`muinit, siginit, shinit`	numeric of length equal to total number of parameters used to model the location, scale or shape parameter(s), resp. See Details section for default (NULL) initial values.
`show`	Logical; if `TRUE` (the default), print details of the fit.
`method`	The optimization method (see `optim` for details).
`maxit`	The maximum number of iterations.
`...`	Other control parameters for the optimization. These are passed to components of the `control` argument of `optim`.

Details

For non-stationary fitting it is recommended that the covariates within the generalized linear models are (at least approximately) centered and scaled (i.e.\ the columns of `ydat` should be approximately centered and scaled).

Let m=mean(xdat) and s=sqrt(6*var(xdat))/pi. Then, initial values assigend when 'muinit' is NULL are m - 0.57722 * s (stationary case). When 'siginit' is NULL, the initial value is taken to be s, and when 'shinit' is NULL, the initial value is taken to be 0.1. When covariates are introduced (non-stationary case), these same initial values are used by default for the constant term, and zeros for all other terms. For example, if a GEV(mu(t)=mu0+mu1*t, sigma, xi) is being fitted, then the initial value for mu0 is m - 0.57722 * s, and 0 for mu1.

Value

A list containing the following components. A subset of these components are printed after the fit. If `show` is `TRUE`, then assuming that successful convergence is indicated, the components `nllh`, `mle` and `se` are always printed.

`trans`	An logical indicator for a non-stationary fit.
`model`	A list with components `mul`, `sigl` and `shl`.
`link`	A character vector giving inverse link functions.
`conv`	The convergence code, taken from the list returned by `optim`. A zero indicates successful convergence.
`nllh`	The negative logarithm of the likelihood evaluated at the maximum likelihood estimates.
`data`	The data that has been fitted. For non-stationary models, the data is standardized.

`mle`	A vector containing the maximum likelihood estimates.
`cov`	The covariance matrix.
`se`	A vector containing the standard errors.
`vals`	A matrix with three columns containing the maximum likelihood estimates of the location, scale and shape parameters at each data point.
`r`	The number of order statistics used.

See Also

`rlarg.diag`, `optim`

Examples

```
## Not run: data(venice)
## Not run: rlarg.fit(venice[,-1])
```

`venice` *Venice Sea Levels*

Description

The `venice` data frame has 51 rows and 11 columns. The final ten columns contain the 10 largest sea levels observed within the year given by the first column. The ten largest sea levels are given for every year in the period 1931 to 1981, excluding 1935 in which only the six largest measurements are available.

Usage

```
data(venice)
```

Format

This data frame contains the following columns:

Year A numeric vector of years.

r1 Annual sea level maxima.

r2 The second largest sea level.

r3 The third largest sea level.

r4 The forth largest sea level.

r5 The fifth largest sea level.

r6 The sixth largest sea level.

r7 The seventh largest sea level.

r8 The eigth largest sea level.

r9 The ninth largest sea level.

r10 The tenth largest sea level.

Source

Smith, R. L. (1986) Extreme value theory based on the *r* largest annual events. *Journal of Hydrology* **86**, 27–43.

References

Coles, S. G. (2001) *An Introduction to Statistical Modelling of Extreme Values.* London: Springer.

`wavesurge` *Wave and Surge Heights in South-West England*

Description

The `wavesurge` data frame has 2894 rows and 2 columns. The columns contain wave and surge heights (in metres) at a single location off south-west England.

Usage

```
data(wavesurge)
```

Format

This data frame contains the following columns:

wave A numeric vector of wave heights.

surge A numeric vector of surge heights.

Source

Coles, S. G. (2001) *An Introduction to Statistical Modelling of Extreme Values.* London: Springer.

`wind` *Annual Maximum Wind Speeds at Albany and Hartford*

Description

The `wind` data frame has 40 rows and 3 columns. The second and third columns contain annual maximum wind speeds at Albany, New York and Hartford, Connecticut respectively, over the period 1944 to 1983. The first column gives the corresponding years.

Usage

```
data(wind)
```

Format

This data frame contains the following columns:

Year A numeric vector of years.

Hartford Annual maximum wind speeds at Hartford.

Albany Annual maximum wind speeds at Albany.

Source

Coles, S. G. (2001) *An Introduction to Statistical Modelling of Extreme Values.* London: Springer.

`wooster` *Minimum Temperatures at Wooster, Ohio*

Description

A numeric vector containing daily minimum temperatures, in degrees Fahrenheit, at Wooster, Ohio, over the period 1983 to 1988.

Usage

```
data(wooster)
```

Format

A vector containing 1826 observations.

Source

Coles, S. G., Tawn, J. A. and Smith, R. L. (1994) A seasonal Markov model for extremely low temperatures. *Environmetrics* **5**, 221–239.

References

Coles, S. G. (2001) *An Introduction to Statistical Modelling of Extreme Values.* London: Springer.

Smith, R. L., Tawn, J. A. and Coles, S. G. (1997) Markov chain models for threshold exceedences. *Biometrica* **84**, 249–268.

Index