Package ‘texmex’

December 31, 2013

Type    Package
Title   Statistical modelling of extreme values
Version 2.1
Date    2013-12-10
Author  Harry Southworth, Janet E. Heffernan
Maintainer Harry Southworth <harry.southworth@gmail.com>

Description Statistical extreme value modelling of threshold excesses, maxima and multivariate extremes. Univariate models for threshold excesses and maxima are the Generalised Pareto, and Generalised Extreme Value model respectively. These models may be fitted by using maximum (optionally penalised-)likelihood, or Bayesian estimation, and both classes of models may be fitted with covariates in any/all model parameters. Model diagnostics support the fitting process. Graphical output for visualising fitted models and return level estimates is provided. For serially dependent sequences, the intervals declustering algorithm of Ferro and Segers is provided, with diagnostic support to aid selection of threshold and declustering horizon. Multivariate modelling is performed via the conditional approach of Heffernan and Tawn, with graphical tools for threshold selection and to diagnose estimation convergence.

License GPL (>= 2)
Depends mvtnorm
Suggests MASS, lattice, knitr
LazyLoad yes
LazyData yes
URL http://code.google.com/p/texmex/
mexDependenceLowLevelFunctions.R mexDependence.R rMaxAR.R mex.R
thinAndBurn.evmSim.R u2gpd.R plot.evmSim.R plot.bootmex.R special.functions.R

NeedsCompilation yes

Repository  CRAN

Date/Publication  2013-12-31 11:51:30

R topics documented:

texmex-package ................................................................. 3
bootmex ................................................................. 5
chi ................................................................. 7
copula ............................................................. 10
dgev ............................................................. 11
dgpd ............................................................. 12
edf ............................................................. 13
evm ............................................................. 13
evmSimSetSeed ............................................................ 14
evmFit ............................................................. 14
evmFit ............................................................. 20
extremalIndex ............................................................. 21
gpdRangeFit ............................................................. 25
liver ............................................................. 27
MCS ............................................................. 28
methods ............................................................ 30
mex ............................................................. 31
mexDependence ....................................................... 36
mexRangeFit ............................................................. 39
migpd ............................................................. 40
migpdSets ............................................................. 40
mrl ............................................................. 42
predict.evmOpt ....................................................... 43
predictWorkers .......................................................... 45
rain, wavesurge and portpirie ............................................... 49
Extreme value modelling

Description

Extreme values modelling, including the conditional multivariate approach of Heffernan and Tawn (2004).

Details

Package: mex
Type: Package
Version: 2.1
Date: 2013-12-10
License: GPL (>=2) | BSD

The package was originally called ‘texmex’ for Threshold EXceedances and Multivariate EXTremes. However, it is no longer the case that only threshold excess models are implemented, so the ‘tex’ bit doesn’t make sense. So, the package is called ‘texmex’ because it used to be called ‘texmex’.

ev: Fit extreme value distributions to data, possibly with covariates. Use maximum likelihood estimation, maximum penalized likelihood estimation, simulate from the posterior distribution or run a parametric bootstrap. Extreme value families include the generalized Pareto distribution (gpd) and generalized extreme value (gev) distribution.
mex: Fit multiple, independent generalized Pareto models to the upper tails of the columns of a data set, and estimate the conditional dependence structure between the columns using the method of Heffernan and Tawn.

bootmex: Bootstrap estimation for parameters in generalized Pareto models and in the dependence structure.
declust: Estimation of extremal index and subsequent declustering of dependent sequences using the intervals estimator of Ferro and Segers.

Author(s)

Harry Southworth, Janet E. Heffernan

Maintainer: Harry Southworth <harry.southworth@gmail.com>
References


Examples

```r
# Analyse the winter data used by Heffernan and Tawn
mymex <- mex(winter, mqu = .7, penalty="none", dqu=.7, which = "NO")
plot(mymex)
# Only do 20 replicates to keep CRAN happy. Do many more in any
# real application
myboot <- bootmex(mymex, R=10)
plot(myboot)
mypred <- predict(myboot, pqu=.95)
summary(mypred, probs = c(.025, .5, .975 ))

# Analyse the liver data included in the package

library(MASS)  # For the rlm function

liver <- liver[liver$ALP.M > 1,]  # Get rid of outlier
liver$dose <- as.numeric(liver$dose)

alt <- resid(rlm(log(ALT.M) ~ log(ALT.B) + ndose, data=liver, method="MM"))
ast <- resid(rlm(log(AST.M) ~ log(AST.B) + ndose, data=liver, method="MM"))
alp <- resid(rlm(log(ALP.M) ~ log(ALP.B) + ndose, data=liver, method="MM"))
tbl <- resid(rlm(log(TBL.M) ~ log(TBL.B) + ndose, data=liver, method="MM"))

r <- data.frame(alt=alt, ast=ast, alp=alp, tbl=tbl)

Amex <- mex(r[liver$dose == "A"], mqu=.7)
Bmex <- mex(r[liver$dose == "B"], mqu=.7)
Cmex <- mex(r[liver$dose == "C"], mqu=.7)
Dmex <- mex(r[liver$dose == "D"], mqu=.7)

par(mfcol=c(3,3))
plot(Amex)

plot(Dmex, col="blue")

## Take a closer look at the marginal behaviour of ALT
# Lines commented out to keep CRAN checks short
#r$ndose <- liver$ndose

#altmod1 <- evm(alt, qu=.7, phi = ~ ndose, xi = ~ ndose, data=r)
#altmod2 <- evm(alt, qu=.7, phi = ~ ndose, data=r)
#altmod3 <- evm(alt, qu=.7, xi = ~ ndose, data=r)
```
#altmod4 <- evm(alt, qu=.7, data=r)

# Prefer model 3, with term for xi on basis of AIC

#balt3 <- evm(alt, qu=.7, xi = ~ ndose, data=r, method="simulate")
#par(mfrow=c(3,3))
#plot(balt3)

# use longer burn-in and also thin the output

#balt3 <- thinAndBurn(balt3,burn=1000,thin=5)
#plot(balt3)

# Get some simulated values for dose D

#DParam <- predict(balt3,type="lp",newdata=data.frame(ndose=4),all=TRUE)[[1]]
#simD <- rgpd(nrow(DParam), sigma=exp(DParam[,"phi"]), xi=DParam[,"xi"], u=quantile(alt, .7))

# These are simulated residuals. Get some baselines and transform all
# to raw scale

#b <- sample(log(liver$ALT.M), size=nrow(balt3$param), replace=TRUE)
#res <- exp(b + simD)

# estimate quantiles on raw scale
#quantile(res, prob=c(.5, .75, .9, .95, .99))

# estimate proportion exceeding 3*upper limit of normal
#mean(res > 36 * 3) # 36 is the upper limit of normal for ALT

---

**bootmex**

*Bootstrap a conditional multivariate extreme values model*

**Description**

Bootstrap a conditional multivariate extreme values model following the method of Heffernan and Tawn, 2004.

**Usage**

bootmex(x, R = 100, nPass=3, trace=10)

## S3 method for class 'bootmex'
plot(x, plots = "gpd", main = "", ...)  
## S3 method for class 'bootmex'
print(x, ...)
Arguments

- **x**: An object of class "mex" as returned by function `mex`.
- **R**: The number of bootstrap runs to perform. Defaults to R=100.
- **nPass**: An integer. Sometimes, particularly with small samples, the estimation process fails with some bootstrap samples. The function checks which runs fail and takes additional bootstrap samples in an attempt to get parameter estimates. By default, it has nPass=3 attempts at this before giving up.
- **trace**: How often to inform the user of progress. Defaults to trace=10.
- **plots**: What type of diagnostic plots to produce. Defaults to "gpd" in which case gpd parameter estimate plots are produced otherwise plots are made for the dependence parameters.
- **main**: Title for plots.
- **...**: Further arguments to be passed to methods.

Details

Details of the bootstrap method are given by Heffernan and Tawn (2004). The procedure is semi-parametric.

Firstly, values of all variables are simulated independently from the parametric Gumbel or Laplace distributions (depending on the choice of margins in the original call to `mex`). The sample size and data dimension match that of the original data set. Then an empirical bootstrap sample is generated from the original data after its transformation to the Gumbel/Laplace scale. Again, sample size and structure match the original data set. The empirical bootstrap samples from each margin are then sorted, and then replaced by their corresponding values from the sorted Gumbel/Laplace samples. This procedure preserves the dependence structure of the empirical bootstrap sample while ensuring the marginal properties of the resulting semi-parametric bootstrap sample are those of the parametric Gumbel/Laplace distribution.

The simulated, ordered Laplace/Gumbel sample is then transformed to the scale of the original data by using the Probability Integral Transform. Values beneath the original thresholds for fitting of the GPD tail models are transformed by using the empirical distribution functions and for values above these thresholds, the fitted GPDs are used. This completes the semi-parametric bootstrap from the data.

Parameter estimation is then carried out as follows: The parameters in the generalized Pareto distributions are estimated by using the bootstrap data, these data are then transformed to the Laplace/Gumbel scale using the original threshold, their empirical distribution function and these estimated GPD parameters. The variables in the dependence structure of these variables are then estimated.

Note that maximum likelihood estimation will often fail for small samples when the generalized Pareto distribution is being fit. Therefore it will often be useful to use penalized likelihood estimation. The function `bootmex` does whatever was done in the call to `mgpd` or `mex` that generated the object with which it is being called.

Also note that sometimes (again, usually with small data sets) all of the simulated Laplace/Gumbel random numbers will be beneath the threshold for the conditioning variable. Such samples are abandoned by `bootmex` and a new sample is generated. This probably introduces some bias into the resulting bootstrap distributions.
The `plot` method produces histograms of bootstrap gpd parameters (the default) or scatterplots of dependence parameters with the point estimates for the original data shown. By design, there is no `coef` method. The bootstrapping is done to account for uncertainty. It is not obvious that adjusting the parameters for the mean bias is the correct thing to do.

**Value**

An object of class `bootmex`. Print and plot functions are available.

**Author(s)**

Harry Southworth

**References**


**See Also**

`migpd`, `mexDependence`, `bootmex`, `predict.mex`.

**Examples**

```r
# Uncomment the following lines to run example - commented out to keep CRAN happy
#mymex <- mex(winter, mqu = .7, dqu = .7, which = "NO")
#myboot <- bootmex(mymex)
#myboot
#plot(myboot, plots="gpd")
#plot(myboot, plots="dependence")
```

---

**chi**

*Measures of extremal dependence*

**Description**

Compute measures of extremal dependence for 2 variables.

**Usage**

```r
chi(data, nq = 100, qlim = NULL, alpha = 0.05, trunc = TRUE)
```

```r
## S3 method for class 'chi'
summary(object, digits=3, ...)
```

```r
## S3 method for class 'chi'
plot(x, show=c("Chi"=TRUE,"ChiBar"=TRUE), lty=1, cility=2, col=1, spcases=TRUE,
cicol=1, xlim=c(0, 1), ylimChi = c(-1, 1), ylimChiBar = c(-1, 1),
```
mainChi = "Chi", mainChiBar = "Chi Bar", xlab = "Quantile",
ylabChi = expression(chi(u)), #"Chi(u)"
ylabChiBar = expression(bar(chi)(u)), #"Chi Bar(u)"
ask, ...)

Arguments

data               A matrix containing 2 numeric columns.
nq                 The number of quantiles at which to evaluate the dependence measures.
qlim               The minimum and maximum quantiles at which to do the evaluation.
alpha              The size of the confidence interval to be used. Defaults to alpha = 0.05.
trunc              Logical flag indicating whether the estimates should be truncated at their theoretical bounds. Defaults to trunc = TRUE.
x, object          An object of class chi.
digits             Number of digits for printing.
show               Logical, of length 2, names "Chi" and "ChiBar". Defaults to c("Chi" = TRUE, "ChiBar" = TRUE).
lty, cilty, col, cicol
Line types and colours for the the estimated quantities and their confidence intervals.
xlim, ylim, ylimChi, ylimChiBar
Limits for the axes.
mainChi, mainChiBar
Main titles for the plots.
xlab, ylab, ylabChiBar
Axis labels for the plots.
spcases             Whether or not to plot special cases of perfect (positive and negative) dependence and independence. Defaults to FALSE.
ask                 Whether or not to ask before reusing the graphics device.
...                 Further arguments to be passed to methods.

Details

Computes the functions chi and chi-bar described by Coles, Heffernan and Tawn (1999). The limiting values of these functions as the quantile approaches 1 give an empirical measure of the type and strength of tail dependence exhibited by the data.

A limiting value of ChiBar equal to 1 indicates Asymptotic Dependence, in which case the limiting value of Chi gives a measure of the strength of dependence in this class. A limiting value of ChiBar of less than 1 indicates Asymptotic Independence in which case Chi is irrelevant and the limiting value of ChiBar gives a measure of the strength of dependence.

The plot method shows the ChiBar and Chi functions. In the case of the confidence interval for ChiBar excluding the value 1 for all of the largest quantiles, the plot of the Chi function is shown in grey.
Value
An object of class chi containing the following.

chi Values of chi and their estimated upper and lower confidence limits.
chibar Values of chibar and their estimated upper and lower confidence limits.
quantile The quantiles at which chi and chi-bar were evaluated.
chiulb, chibarulb Upper and lower bounds for chi and chi-bar.

Note
When the data contain ties, the values of chi and chibar are calculated by assigning distinct ranks to tied values using the rank function with argument ties.method = "first". This results in the values of chi and chibar being sensitive to the order in which the tied values appear in the data.

The code is a fairly simple reorganization of code written by Janet E. Heffernan and Alec Stephenson and which appears in the chiplot function in the evd package.

Author(s)
Janet E. Heffernan, Alec Stephenson, Harry Southworth

References

See Also
MCS, rank

Examples

D <- liver[liver$dose == "D",]
chiD <- chi(D[, 5:6])
par(mfrow=c(1,2))
plot(chiD)

A <- liver[liver$dose == "A",]
chiA <- chi(A[, 5:6])
# here the limiting value of chi bar(u) lies away from one so the chi plot is
# not relevant and is plotted in grey
plot(chiA)
copula

Calculate the copula of a matrix of variables

Description

Returns the copula of several random variables.

Usage

copula(x, na.last = NA)
## S3 method for class 'copula'
plot(x, jitter. = FALSE, jitter.factor = 1, ...)

Arguments

x     A matrix or data.frame containing numeric variables.
na.last How to treat missing values. See rank for details.
jitter In the call to plotcopula, if jitter=TRUE, the values are jittered before plotting. Defaults to jitter. = FALSE.
jitter.factor How much jittering to use. Defaults to jitter.factor = 1.
...     Further arguments to be passed to plot method.

Details

The result is obtained by applying edf to each column of x in turn.
Print, plot and summary methods are available for the copula class.

Value

A matrix with the same dimensions as x, each column of which contains the quantiles of each column of x. This object is of class copula.

Author(s)

Harry Southworth

See Also

edf

Examples

D <- liver[liver$dose == "D",]
Dco <- copula(D)
plot(Dco)
Density, cumulative density, quantiles and random number generation for the generalized extreme value distribution

Usage

dgev(x, mu, sigma, xi, log.d = FALSE)
pgev(q, mu, sigma, xi, lower.tail = TRUE, log.p = FALSE)
qgev(p, mu, sigma, xi, lower.tail = TRUE, log.p = FALSE)
rgev(n, mu, sigma, xi)

Arguments

x, q, p  Value, quantile or probability respectively.
n Number of random numbers to simulate.
mu Location parameter.
sigma Scale parameter.
xi Shape parameter.
log.d, log.p Whether or not to work on the log scale.
lower.tail Whether to return the lower tail.

Details

Random number generation is done as a transformation of the Gumbel distribution; Gumbel random variates are generated as the negative logarithm of standard exponentials.

Author(s)

Harry Southworth

Examples

x <- rgev(1000, mu=0, sigma=1, xi=.5)
hist(x)
x <- rgev(1000, mu=0, sigma=exp(rnorm(1000, 1, .25)), xi=rnorm(1000, .5, .2))
hist(x)
plot(pgev(x, mu=0, sigma=1, xi=.5))
Density, cumulative density, quantiles and random number generation for the generalized Pareto distribution

Usage

dgpd(x, sigma, xi, u = 0, log.d = FALSE)
pdgd(q, sigma, xi, u = 0, lower.tail = TRUE, log.p = FALSE)
qgpd(p, sigma, xi, u = 0, lower.tail = TRUE, log.p = FALSE)
rgpd(n, sigma, xi, u = 0)

Arguments

x, q, p  Value, quantile or probability respectively.
n  Number of random numbers to simulate.
sigma  Scale parameter.
xi  Shape parameter.
u  Threshold
log.d, log.p  Whether or not to work on the log scale.
lower.tail  Whether to return the lower tail.

Details

Random number generation is done by transformation of a standard exponential.

Author(s)

Janet E Heffernan, Paul Metcalfe, Harry Southworth

Examples

x <- rgpd(1000, sigma=1, xi=.5)
hist(x)
x <- rgpd(1000, sigma=exp(rnorm(1000, 1, .25)), xi=rnorm(1000, .5, .2))
hist(x)
plot(pdgd(x, sigma=1, xi=.5))
**edf**

*Compute empirical distribution function*

**Description**

Compute the empirical distribution function

**Usage**

`edf(x, na.last = NA)`

**Arguments**

- `x` A numeric vector
- `na.last` How to treat missing values. See `rank` for details.

**Value**

A vector of quantiles relating to the observations in `x`.

**Author(s)**

Harry Southworth

**See Also**

`copula`

**Examples**

```r
plot(winter$NO, edf(winter$NO))
```

---

**endPoint**

*Calculate upper end point for a fitted extreme value model*

**Description**

Calculate upper end point for fitted extreme value model

**Usage**

`endPoint(y, verbose=TRUE, .unique=TRUE, ...)`

```r
## S3 method for class 'evmOpt'
endPoint(y, verbose=TRUE, .unique=TRUE, ...)
## S3 method for class 'evmSim'
endPoint(y, verbose=TRUE, .unique=TRUE, ...)
```
Arguments

- **y**: Object of class `evmOpt` or `evmSim`, as returned by `evm`.
- **verbose**: Whether to print output.
- **.unique**: Whether or not to use only unique values of `y`.
- ... further arguments to be passed to the `signif` function.

Value

In cases where the fitted shape parameter is negative, the fitted finite upper endpoint of the extreme value model.

Author(s)

Janet E. Heffernan

---

**evm**  
*Extreme value modelling*

---

Description

Likelihood based modelling and inference for extreme value models, possibly with explanatory variables.

Usage

```r
evm(y, data, family, ...)
```

## Default S3 method:
- `evm(y, data, family=gpd, th=-Inf, qu, ...
  ... , penalty = NULL,
  prior = "gaussian", method = "optimize", cov="observed", start = NULL,
  priorParameters = NULL, maxit = 10000, trace = NULL,
  iter = 40500, burn = 500, thin = 4,
  proposal.dist = c("gaussian", "cauchy"),
  jump.cov, jump.const=NULL, R=100, verbose = TRUE)

## S3 method for class 'evmOpt'
- `print(x, digits=max(3,getOption("digits") - 3), ...)`
- `summary(object, nsim=1000, alpha=0.05, ...)`
- `plot(x, main=rep(NULL, 4), xlab=rep(NULL, 4),
  nsim=1000, alpha=0.05, ...)`
- `AIC(object, penalized=FALSE, ..., k=2)`
Arguments

- **y**
  - Either a numeric vector or the name of a variable in `data`.

- **data**
  - A data frame containing `y` and any covariates.

- **family**
  - An object of class `texmexFamily`. Defaults to `family=gpd` and a generalized Pareto distribution is fit to the data. Alternatively, the family could be `gev`, resulting in a generalized extreme value distribution being fit. No other families are currently available in texmex, but users may write their own.

- **th**
  - For threshold excess models (such as when `family=gpd`), the threshold for `y`, exceedances above which will be used to fit the upper tail model. Note that if you have already thresholded your data and want to model all of `y`, you still need to specify `th`.

- **qu**
  - An alternative to `th`, a probability defined such that `quantile(y, qu)` equals `th`.

- **...**
  - In `evm`, formulae for the parameters in the family, e.g. `phi ~ x`. If none are specified, they all default to `~1`.

- **penalty**
  - How to penalize the likelihood. Currently, either "none", "gaussian" or "lasso" are the only allowed values. If `penalty` is "gaussian" or "lasso" then the parameters for the penalization are specified through the `prior` argument. See below. Defaults to `penalty=NULL` and applies maximum likelihood estimation.

- **prior**
  - If `method = "optimize"`, just an alternative way of specifying the penalty, and only one or neither of `penalty` and `prior` should be given. If `method = "simulate"`, `prior` must be "gaussian" because no other prior distributions have been implemented.

- **method**
  - Should be either "optimize" (the default), "simulate" or "bootstrap". The first letter or various abbreviations will do. If "optimize" is used, the (penalized) likelihood is directly optimized using `optim` and point estimates (either ML or MAP estimates) are returned with other information. If "simulate", a Metropolis algorithm is used to simulate from the joint posterior distribution of the parameters. If "bootstrap", a parametric bootstrap is performed.

- **cov**
  - How to compute the covariance matrix of the parameters. Defaults to `cov = "observed"` in which case the observed information matrix is used, if the `info` element of the `texmexFamily` object is present. Note that currently, this is not implemented for `gev`. The only other option is `cov = "numeric"` in which case a numerical approximation of the Hessian is used (see the help for `optim`). In some cases, particularly with small samples, the numerical approximation can be quite different from the closed form (`cov="observed"`) result, and the value derived from the observed information should be preferred. However, in either case,
since the underlying log-likelihood may be far from quadratic for small samples, the resulting estimates of standard errors are liable to approximate poorly the true standard errors. Also see the comments in the Details section, below.

**start**
Starting values for the parameters, to be passed to optim. If not provided, the function will use the start element of the texmexFamily object if it exists.

**priorParameters**
A list with two components. The first should be a vector of means, the second should be a covariance matrix if the penalty/prior is "gaussian" or "quadratic" and a diagonal precision matrix if the penalty/prior is "lasso", "L1" or "Laplace". If method = "simulate" then these represent the parameters in the Gaussian prior distribution. If method = 'optimize' then these represent the parameters in the penalty function. If not supplied: all default prior means are zero; all default prior variances are $10^4$; all covariances are zero.

**maxit**
The number of iterations allowed in optim.

**trace**
Whether or not to print progress to screen. If method = "optimize", the argument is passed into optim – see the help for that function. If method = "simulate", the argument determines at how many steps of the Markov chain the function should tell the user, and in this case it defaults to trace = 10000.

**iter**
Number of simulations to generate under method = "simulate". Defaults to 40500.

**burn**
The number of initial steps to be discarded. Defaults to 500.

**thin**
The degree of thinning of the resulting Markov chains. Defaults to 4 (one in every 4 steps is retained).

**proposal.dist**
The proposal distribution to use, either multivariate gaussian or a multivariate Cauchy.

**jump.cov**
Covariance matrix for proposal distribution of Metropolis algorithm. This is scaled by jump.const.

**jump.const, j**
Control parameter for the Metropolis algorithm.

**verbose**
Whether or not to print progress to screen. Defaults to verbose=TRUE.

**x, object**
Object of class evmOpt, evmSim, summary.evmOpt or summary.evmSim returned by evmSim or summary.evmSim. In texmexMetropolis, x is a matrix, the first row of which is used as the starting point of the simulation.

**digits**
Number of digits for printing.

**main**
In plot method for class evmSim, titles for diagnostic plots. Should be a vector of length 4, with values corresponding to the character strings to appear on the titles of the pp, qq, return level, and density estimate plots respectively.

**xlab**
As for main but labels for x-axes rather than titles.

**nsim**
In plot and summary methods for class evmSim. The number of replicates to be simulated to produce the simulated tolerance intervals. Defaults to nsim = 1000.

**alpha**
In plot and summary methods for class evmSim. A 100(1 - alpha)% simulation envelope is produced. Defaults to alpha = 0.05

**penalized**
Whether to compute AIC using the penalized log-likelihood or the true log-likelihood. Defaults to penalized=FALSE and uses the true log-likelihood.
\textbf{k} \hspace{1cm} \text{Constant used in calculation of } \text{AIC}=-2\text{loglik} + k*p, \text{ defaults to } k=2.

\textbf{print.seed} \hspace{1cm} \text{Whether or not to print the seed used in the simulations, or to annotate the plots with it. Defaults to } \text{print.seed}=\text{FALSE}.

\textbf{which.plots} \hspace{1cm} \text{In } \text{plot} \text{ method for class } \text{evmSim}. \text{ Which plots to produce. Option 1 gives kernel density estimates, 2 gives traces of the Markov chains with superimposed cumulative means, 3 gives autocorrelation functions. Defaults to } \text{which.plots}=1:3.

\textbf{density.adjust} \hspace{1cm} \text{In } \text{plot} \text{ method for class } \text{evmSim}. \text{ Passed into } \text{density}. \text{ Controls the amount of smoothing of the kernel density estimate. Defaults to } \text{density.adjust}=2.

\textbf{log.lik, proposals} \hspace{1cm} \text{The log-likelihood for the model, and a matrix of random numbers drawn from the proposal distribution.}

\textbf{map} \hspace{1cm} \text{In } \text{texmexJumpConst}, \text{ a model fit by maximum (penalized) likelihood. The function is not intended to be called by the end user.}

\textbf{R} \hspace{1cm} \text{The number of parametric bootstrap samples to run when } \text{method = "bootstrap" is requested. Defaults to 100.}

\textbf{Details}

In previous versions of \text{texmex}, the main modelling function was \text{gpd} and models using the generalized Pareto distribution (GPD) were used. In this version, the main modelling function is now \text{evm} (extreme value model) and the distribution to be used is specified through the \text{family} argument.

Object \text{gpd} is now an object of class \text{texmexFamily}. Currently, the only other \text{texmexFamily} object available is \text{gev} which results in fitting a generalized extreme value (GEV) distribution to the data.

See Coles (2001) for an introduction to extreme value modelling and the GPD and GEV models.

For the GPD model, we use the following parameterisation of \text{evm}:

\[ P(Y \leq y) = 1 - \left(1 + \frac{\xi y}{\sigma}\right)^{-1/\xi} \]

for \( y \geq 0 \) and \( 1 + \xi y/\sigma \geq 0 \).

For the GEV model, we use:

\[ P(Y \leq y) = \exp \left\{ - \left[ 1 + \xi \left( \frac{y - \mu}{\sigma} \right) \right]^{-1/\xi} \right\} \]

In each case, the scale parameter is sigma (\( \sigma \)) and the shape parameter is xi (\( \xi \)). The GEV distribution also has location parameter mu (\( \mu \)).

Working with the log of the scale parameter improves the stability of computations, makes a quadratic penalty more appropriate and enables the inclusion of covariates in the model for the scale parameter, which must remain positive. We therefore work with \( \phi=\log(\sigma) \). All specification of priors or penalty functions refer to \( \phi \) rather than \( \sigma \). A quadratic penalty can be thought of as a Gaussian prior distribution, whence the terminology of the function.
Parameters of the evm are estimated by using maximum (penalized) likelihood (method = "optimize"), or by simulating from the posterior distribution of the model parameters using a Metropolis algorithm (method = "simulate"). In the latter case, start is used as a starting value for the Metropolis algorithm; in its absence, the maximum penalized likelihood point estimates are computed and used.

A bootstrap approach is also available (method = "bootstrap"). This runs a parametric bootstrap, simulating from the model fit by optimization.

When method = "optimize", the plot function produces diagnostic plots for the fitted model. These differ depending on whether or not there are covariates in the model. If there are no covariates then the diagnostic plots are PP- and QQ-plots, a return level plot (produced by plotrl.evmsim) and a histogram of the data with superimposed density estimate. These are all calculated using the data on the original scale. If there are covariates in the model then the diagnostics consist of PP- and QQ-plots calculated by using the model residuals (which will be standard exponential deviates under the GPD model and standard Gumbel deviates under the GEV model), and plots of residuals versus fitted model parameters.

The PP- and QQ-plots show simulated pointwise tolerance intervals. The region is a 100(1 - alpha)% region based on nsim simulated samples.

When method = "simulate" the plot function produces diagnostic plots for the Markov chains used to simulate from the posterior distributions for the model parameters. If the chains have converged on the posterior distributions, the trace plots should look like "fat hairy caterpillars" and their cumulative means should converge rapidly. Moreover, the autocorrelation functions should converge quickly to zero.

When method = "bootstrap", the plot function plots the bootstrap distributions of the parameters.

When method = "simulate" the print and summary functions give posterior means and standard deviations. Posterior means are also returned by the coef method. Depending on what you want to do and what the posterior distributions look like (use plot method) you might want to work with quantiles of the posterior distributions instead of relying on standard errors.

When method = "bootstrap", summaries of the bootstrap distribution and the bootstrap estimate of bias are displayed.

**Value**

If method = "optimize", an object of class evmOpt:

- **call** The call to evmSim that produced the object.
- **data** The original data (above and below the threshold for fitting if a distribution for threshold excesses has been used). In detail, data is a list with elements y and D. y is the response variable and D is a list containing the design matrices implied by any formulae used in the call to evm.
- **convergence** Output from optim relating to whether or not the optimizer converged.
- **message** A message telling the user whether or not convergence was achieved.
- **threshold** The threshold of the data above which the evmSim model was fit.
- **penalty** The type of penalty function used, if any.
coefficients  The parameter estimates as computed under maximum likelihood or maximum penalized likelihood.

rate  The proportion of observations above the threshold. If the model is not a threshold exceedance model (e.g. the GEV model), the rate will be 1.

priorParameters  See above.

residuals  Residuals computed using the residual function in the `texmexFamily` object, if any.

ploglik  The value of the optimized penalized log-likelihood.

loglik  The value of the optimized (unpenalized) log-likelihood. If `penalty='none'` is used, this will be identical to `ploglik`, above.

cov  The estimated covariance of the parameters in the model.

se  The estimated standard errors of the parameters in the model.

xlevels  A named list containing a named list for each design matrix (main parameter) in the model. Each list contains an element named after each factor in the linear predictor for the respective design matrix. These are used by the `predict` method to ensure all factor levels are known, even if they don't appear in `newdata`.

If `method = "simulate"`, an object of class `evmSim`:

call  The call to `evmSim` that produced the object.

threshold  The threshold above which the model was fit.

map  The point estimates found by maximum penalized likelihood and which were used as the starting point for the Markov chain. This is of class `evmOpt` and methods for this class (such as `resid` and `plot`) may be useful.

burn  The number of steps of the Markov chain that are to be treated as the burn-in and not used in inferences.

thin  The degree of thinning used.

chains  The entire Markov chain generated by the Metropolis algorithm.

y  The response data above the threshold for fitting.

seed  The seed used by the random number generator.

param  The remainder of the chain after deleting the burn-in and applying any thinning.

If `method = "bootstrap"`, an object of class `evmBoot`:

call  The call to `evmBoot` that produced the object.

replicates  The parameter estimates from the bootstrap fits.

map  The fit by by maximum penalized likelihood to the orginal data. This is of class `evmOpt` and methods for this class (such as `resid` and `plot`) may be useful.

There are summary, plot, print and coefficients methods available for these classes.
Note

For both GPD and GEV models, when there are estimated values of $\xi \leq -0.5$, the regularity conditions of the likelihood break down and inference based on approximate standard errors cannot be performed. In this case, the most fruitful approach to inference appears to be by the bootstrap. It might be possible to simulate from the posterior, but finding a good proposal distribution might be difficult and you should take care to get an acceptance rate that is reasonably high (around 40% when there are no covariates, lower otherwise).

Author(s)

Janet E. Heffernan, Harry Southworth. Some of the internal code is based on the gpd.fit function in the ismev package and is due to Stuart Coles.

References


See Also

rl.evmOpt, predict.evmOpt, evm.declustered.

Examples

```r
mod <- evm(rain, th=30)
mod
par(mfrow=c(2, 2))
plot(mod)

# mod <- evm(rain, th=30, method="sim")
# par(mfrow=c(3, 2))
# plot(mod)

mod <- evm(SeaLevel, data=portpirie, family=gev)
mod
plot(mod)

# mod <- evm(SeaLevel, data=portpirie, family=gev, method="sim")
# par(mfrow=c(3, 3))
# plot(mod)
```

---

evmSimSetSeed

Set the seed from a fitted evmSim object.

Description

Set the seed from a fitted evmSim object to ensure reproducibility of output.
Usage

`evmSimSetSeed(x)`

Arguments

`x` An object of class `evmSim`, as returned by `evm` using `method = "simulate"`.

Details

Sets the seed to the value used to fit the model.

Author(s)

Harry Southworth

See Also

`evm`

Examples

```r
# data <- rnorm(1000)
# mod <- evm(data, qu=.7, method="simulate")
# evmSimSetSeed(mod)
# mod1 <- evm(data, qu=.7, method="simulate") # this produces the same MCMC output as mod
```

---

`extremalIndex` Extremal index estimation and automatic declustering

Description

Given a threshold which defines excesses above that threshold, estimate the extremal index of a dependent sequence by using the method of Ferro and Segers, 2003. The extremal index estimate can then be used to carry out automatic declustering of the sequence to identify independent clusters and estimate the GPD for cluster maxima. Graphical diagnostics of model fit are available.

Usage

```r
extremalIndex(y, data = NULL, threshold)
```

```r
extremalIndexRangeFit(y, data = NULL, umin = quantile(y,.5),
                      umax = quantile(y, 0.95), nint = 10, nboot = 100, alpha = .05,
                      xlab = "Threshold", addNexcesses = TRUE, estGPD=TRUE, verbose = TRUE,
                      trace = 10, ...)
```

```r
bootExtremalIndex(x)
```

```r
declust(y, r=NULL, data = NULL, ...)
```
## extremalIndex

```r
## S3 method for class 'extremalIndex'
declust(y, r=NULL,...)
```

```r
## S3 method for class 'declustered'
plot(x, ylab = "Data",...)
```

```r
## S3 method for class 'declustered'
evm(y, data=NULL, family=gpd, ...)
```

### Arguments

- **y**  
  Argument to function extremalIndex: either a numeric vector or the name of a variable in `data`.

- **data**  
  A data frame containing `y` and any covariates. In `evm.declustered`, it should be `NULL` and is included to match the arguments of generic `evm`.

- **family**  
  The type of extreme value model. The user should not change this from its default in `evm.declustered`.

- **threshold**  
  The threshold for `y`, exceedances above which will be used to estimate the extremal index and carry out automatic declustering.

- **x**  
  Objects passed to methods.

- **r**  
  Positive integer: run length to be used under "runs" declustering. If specified then so-called "runs" declustering will be carried out, otherwise defaults to `NULL` in which case the automatic "intervals" declustering method of Ferro and Segers is used.

- **umin**  
  The minimum threshold above which to esimate the parameters.

- **umax**  
  The maximum threshold above which to esimate the parameters.

- **nint**  
  The number of thresholds at which to perform the estimation.

- **nboot**  
  Number of bootstrap samples to simulate at each threshold for estimation.

- **alpha**  
  100(1 - alpha)% confidence intervals will be plotted with the point estimates. Defaults to `alpha = 0.05`.

- **xlab**  
  Label for the x-axis.

- **ylab**  
  Label for the y-axis.

- **addNexcesses**  
  Whether to annotate the top axis of plots with the number of excesses above the corresponding threshold. Defaults to `TRUE`.

- **estGPD**  
  Whether to estimate GPD parameters at each choice of threshold – defaults to `TRUE` in which case the GPD parameters are estimated.

- **verbose**  
  Whether to report on progress in RangeFit calculations. Defaults to `TRUE`.

- **trace**  
  How frequently to report bootstrap progress in RangeFit calculations. Defaults to 10.

- **...**  
  Further arguments to methods.
Details

The function `extremalIndex` estimates the extremal index of a dependent series of observations above a given threshold `threshold`, returning an object of class "extremalIndex". Plot and print methods are available for this class. A graphical diagnostic akin to Figure 1 in Ferro and Segers (2003) is produced by the `plot` method for this class. This plot is used to test the model assumption underpinning the estimation, with good fit being indicated by inter-exceedance times which correspond to inter-cluster times lying close to the diagonal line indicated.

In addition to good model fit, an appropriate choice of threshold is one above which the estimated extremal index is stable over further, higher thresholds (up to estimation uncertainty). This can be assessed by using the function `extremalIndexRangeFit`, which examines a range of threshold values. At each threshold, the extremal index is estimated; that estimate is used to decluster the series and the parameters of the GPD are optionally estimated for the resulting declustered series. Uncertainty in the estimation of the extremal index and GPD parameters is assessed by using a bootstrap scheme which accounts for uncertainty in the extremal index estimation, and the corresponding uncertainty in the declustering of the series.

The function `declust` returns an object of class "declustered", identifying independent clusters in the original series. Print, plot and show methods are available for this class. The GPD model can be fitted to objects of this class, including the use of covariates in the linear predictors for the parameters of the GPD. See examples below.

Value

The function `extremalIndex` returns a list of class "extremalIndex":

- `EIintervals`: Estimate of the extremal index by using the intervals estimator of Ferro and Segers.
- `threshold`: threshold for declustering and estimation
- `totaln`: length of original data series
- `nExceed`: number of exceedances of threshold in original series.
- `thExceedanceProb`: probability of threshold exceedance in original series.
- `call`: the original function call
- `interExceedTimes`: times between threshold exceedances
- `thExceedances`: observation from the original series which are above threshold
- `exceedanceTimes`: times of occurrence of threshold exceedances
- `y`: original dependent series
- `data`: data frame or NULL

The function `declust` returns a list of type "declustered":

- `clusters`: integer labels assigning threshold exceedances to clusters
- `sizes`: number of exceedances in each cluster
clusterMaxima  vector made up of the largest observation from each distinct cluster. In the case of ties, the first value is taken.

isClusterMax  logical; length equal to number of threshold exceedances, value is TRUE for threshold exceedances which correspond to cluster maxima

y  see entry for object of class "extremalIndex" above

data  see entry for object of class "extremalIndex" above

threshold  see entry for object of class "extremalIndex" above

EIintervals  see entry for object of class "extremalIndex" above

call  see entry for object of class "extremalIndex" above

InterExceedTimes  times between threshold exceedances, length is one less than the number of threshold exceedances

InterCluster  logical: indicates inter exceedance times larger than r the run length used for declustering

thExceedances  see entry for object of class "extremalIndex" above

exceedanceTimes  see entry for object of class "extremalIndex" above

r  run length used for declustering

nClusters  Number of independent clusters identified

method  Method used for declustering (either "intervals" or "runs")

The function bootExtremalIndex return a single vector corresponding to a bootstrap sample from the original series: observations are censored at threshold so that values below this threshold are indicated by the value -1.

The method evm for class "declustered" returns an object of type "evmOpt" or "evmSim" depending on the precise function call - see documentation for evm.

Author(s)

Janet E. Heffernan

References


See Also

evm
gpdRangeFit

Examples

par(mfrow=c(2,2));
extremalIndexRangeFit(summer$O3,nboot=10)
ei <- extremalIndex(summer$O3,threshold=45)
plot(ei)
d <- declust(ei)
plot(d)
evm(d)

## fitting with covariates:

so2 <- extremalIndex(SO2,data=winter,threshold=15)
plot(so2)
so2 <- extremalIndex(SO2,data=winter,threshold=20)
plot(so2) ## fits better

so2.d <- declust(so2)
par(mfrow=c(1,1)); plot(so2.d)
so2.d.gpd <- evm(so2.d) # AIC 661.1

evm(so2.d,phi=-NO)
evm(so2.d,phi=-NO2)
evm(so2.d,phi=-O3) # better AIC 651.9
evm(so2.d,phi=-PM10)

so2.d.gpd.o3 <- evm(so2.d,phi=-O3)

par(mfrow=c(2,2)); plot(so2.d.gpd.o3)

---

gpdRangeFit Estimate generalized Pareto distribution parameters over a range of values

Description

Estimate generalized Pareto distribution parameters over a range of values, using maximum (penalized) likelihood.

Usage

gpdRangeFit(data, umin=quantile(data, 0.05), umax=quantile(data, 0.95), nint = 10,
penalty = "gaussian", priorParameters = NULL, alpha=0.05)

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

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

## S3 method for class 'gpdRangeFit'
plot(x, xlab = "Threshold", ylab = NULL, main = NULL, addNexcesses=TRUE, ...)
Arguments

- **data**: The data vector to be modelled.
- **umin**: The minimum threshold above which to estimate the parameters.
- **umax**: The maximum threshold above which to estimate the parameters.
- **nint**: The number of thresholds at which to perform the estimation.
- **penalty**: The type of penalty to be used in the maximum penalized likelihood estimation. Should be either "gaussian" or "none". Defaults to "gaussian".
- **priorParameters**: Parameters to be used for the penalty function. See the help for `evm` for more information.
- **alpha**: $100(1 - \alpha)\%$ confidence intervals will be plotted with the point estimates. Defaults to $\alpha = 0.05$.
- **x, object**: Arguments to `print` and `summary` functions.
- **xlab**: Label for the x-axis.
- **ylab**: Label for the y-axis.
- **main**: The main title.
- **addNexcesses**: Annotate top axis with numbers of threshold excesses arising with the corresponding values of threshold on the bottom axis.
- **...**: Arguments to `plot`

Details

This is Stuart Coles’ `gpd.fitrange`, as it appears in the `ismev` package, refactored into a function that does the computations, and method functions. The function uses `evm` internally and uses the default options for that function.

Note this function does not extend to assessing model fit when there are covariates included in the model.

Author(s)

Stuart Coles, Janet E Heffernan, Harry Southworth

See Also

- `evm`

Examples

```r
par(mfrow=c(1,2))
plot(gpdRangeFit(rain))
```
Liver related laboratory data

Description
Liver related laboratory data from a randomized, blind, parallel group clinical trial with 4 doses of a drug.

Usage
data(liver)

Format
A data frame with 606 observations on the following 9 variables.

- **alpNb**: Alkaline phosphatase at baseline. A numeric vector.
- **altNb**: Alanine aminotransferase at baseline. A numeric vector.
- **astNb**: Aspartate aminotransferase at baseline. A numeric vector.
- **tblNb**: Total bilirubin at baseline. A numeric vector.
- **alpNm**: Alkaline phosphatase after treatment. A numeric vector.
- **altNm**: Alanine aminotransferase after treatment. A numeric vector.
- **astNm**: Aspartate aminotransferase after treatment. A numeric vector.
- **tblNm**: Total bilirubin after treatment. A numeric vector.
- **dose**: The treatment group (i.e. dose group). A factor with levels a b c d

Details
Dose A is the lowest dose, dose, B the next, C the next, and D the highest dose. The baseline values were taken prior to any treatment being received, and the clinical trial had a single post-baseline visit.

Source
AstraZeneca data on file.
Description

Compute multivariate conditional Spearman’s rho over a range of quantiles.

Usage

```r
MCS(X, p = seq(0.1, 0.9, by = 0.1))
bootMCS(X, p = seq(0.1, 0.9, by = 0.1), R = 100, trace = 10)
```

```r
## S3 method for class 'MCS'
plot(x, xlab="p", ylab="MCS", ...)

## S3 method for class 'bootMCS'
summary(object, alpha = 0.05, ...)  
## S3 method for class 'bootMCS'
plot(x, xlab="p", ylab="MCS", alpha = 0.05, ylim, ...)
```

Arguments

- **X**: A matrix of numeric variables.
- **p**: The quantiles at which to evaluate.
- **R**: The number of bootstrap samples to run. Defaults to \( R = 100 \).
- **trace**: How often to inform the user of progress. Defaults to \( \text{trace} = 10 \).
- **x**, **object**: An object of class MCS or bootMCS.
- **xlab**, **ylab**: Axis labels.
- **alpha**: A 100(1 - alpha)% pointwise confidence interval will be produced. Defaults to \( \alpha = 0.05 \).
- **ylim**: Plotting limits for bootstrap plot.
- **...**: Optional arguments to be passed into methods.

Details

The method is described in detail by Schmid and Schmidt (2007). The main code was written by Yiannis Papastathopoulou, wrappers written by Harry Southworth.

When the result of a call to bootMCS is plotted, simple quantile bootstrap confidence intervals are displayed.
Value

MCS returns an object of class MCS. There are plot and summary methods available for this class.

- **MCS**: The estimated correlations.
- **p**: The quantiles at which the correlations were evaluated at
- **call**: The function call used.

bootMCS returns an object of class bootMCS. There are plot and summary methods available for this class.

- **replicates**: Bootstrap replicates.
- **p**: The quantiles at which the correlations were evaluated at
- **R**: Number of bootstrap samples.
- **call**: The function call used.

Author(s)

Yiannis Papastathopoulos, Harry Southworth

References


See Also

- chi

Examples

```r
D <- liver[liver$dose == "D",]
plot(D)
# Following lines commented out to keep CRAN happy
#Dmcs <- bootMCS(D[, 5:6])
#Dmcs
#plot(Dmcs)
```
Methods for texmex objects

Description

Methods for texmex objects.

Usage

```r
## S3 method for class 'evmSim'
coef(object,...)
## S3 method for class 'mex'
coef(object,...)
## S3 method for class 'migpd'
coef(object,...)
## S3 method for class 'evmSim'
coefficients(object,...)
## S3 method for class 'evmOpt'
coefficients(object,...)
## S3 method for class 'mex'
coefficients(object,...)
## S3 method for class 'migpd'
coefficients(object,...)

## S3 method for class 'extremalIndex'
plot(x,...)
## S3 method for class 'mexRangeFit'
plot(x, col=2, bootcol="grey", addNexcesses=TRUE, ...)

## S3 method for class 'MCS'
print(x,...)
## S3 method for class 'bootMCS'
print(x,...)
## S3 method for class 'chi'
print(x,...)
## S3 method for class 'copula'
print(x,...)
## S3 method for class 'declustered'
print(x,...)
## S3 method for class 'extremalIndex'
print(x,...)
## S3 method for class 'mex'
print(x,...)
## S3 method for class 'mexDependence'
print(x,...)
## S3 method for class 'migpd'
```

print(x,...)
## S3 method for class 'predict.mex'
print(x,...)
## S3 method for class 'summary.evmSim'
print(x,...)
## S3 method for class 'summary.evmOpt'
print(x,digits=3,...)
## S3 method for class 'summary.predict.mex'
print(x,...)
## S3 method for class 'MCS'
summary(object,...)
## S3 method for class 'copula'
summary(object,...)
## S3 method for class 'mex'
summary(object,...)
## S3 method for class 'migpd'
summary(object,verbose=TRUE,...)

Arguments

x A texmex object of class indicated by method type.
object A texmex object of class indicated by method type.
digits Number of digits for printing.
verbose Logical, whether to show model fitting summary in addition to just model parameter estimates.
col The colour of points on the plot (the point estimates in plot.mexRangeFit).
bootcol The colour of points for point estimates from bootstrap samples in plot.mexRangeFit.
addNexcesses Whether or not to print the number of threshold excesses in plot.mexRangeFit. Defaults to TRUE.
... Further arguments to methods.

Author(s)

Harry Southworth, Janet E. Heffernan

mex

Conditional multivariate extreme values modelling

Description

Fit the conditional multivariate extreme value model of Heffernan and Tawn
Usage
mex(data, which, mth, mqu, dqu, margins = "laplace", constrain = TRUE,
  v = 10, penalty = "gaussian", maxit = 10000, trace = 0,
  verbose = FALSE, priorParameters = NULL)

## S3 method for class 'mex'
plot(x, quantiles = seq(0.1, by = 0.2, len = 5), col = "grey", ...)
## S3 method for class 'mex'
predict(object, which, pqu=0.99, nsim=1000, trace=10, ...)
## S3 method for class 'predict.mex'
summary(object, mth, probs=c(0.05, 0.5, 0.95), ...)
## S3 method for class 'predict.mex'
plot(x, pch=c(1, 3), col=c(2, 8), cex=c(1, 1), ask=TRUE, ...)

Arguments

**data**
A numeric matrix or data.frame, the columns of which are to be modelled.

**which**
The variable on which to condition. This can be either scalar, indicating the
column number of the conditioning variable, or character, giving the column name of the conditioning variable.

**mth**
Marginal thresholds. In *mex*, the threshold above which to fit generalized Pareto
distributions. If this is a vector of length 1, the same threshold will be used
for each variable. Otherwise, it should be a vector whose length is equal to the
number of columns in *data*.

In *summary.predict.mex*, the thresholds over which to simulate data from the
fitted multivariate model. If not supplied, it is taken to be the thresholds that
were used to fit the dependence model on the scale of the original data.

**mqu**
Marginal quantiles As an alternative to specifying the marginal GPD fitting
thresholds via *mth*, you can specify the quantile (a probability) above which
to fit generalized Pareto distributions. If this is a vector of length 1, the same
quantile will be used for each variable. Otherwise, it should be a vector whose
length is equal to the number of columns in *data*.

**dqu**
Dependence quantile. Used to specify the quantile at which to threshold the
conditioning variable data when estimating the dependence parameters. For ex-
ample *dqu=0.7* will result in the data with the highest 30% of values of the
conditioning variable being used to estimate the dependence parameters. The
same threshold will be used for each dependent variable. If not supplied then
the default is to set *dqu=mqu[which]* the quantile corresponding to the thresh-
old used to fit the marginal model to the tail of the conditioning variable. Note
that there is no requirement for the quantiles used for marginal fitting (mqu) and
dependence fitting (dqu) to be the same, or for them to be ordered in any way.

**margins**
See documentation for *mexDependence*.

**constrain**
See documentation for *mexDependence*.

**v**
See documentation for *mexDependence*.

**penalty**
How to penalize the likelihood when estimating the marginal generalized Pareto
distributions. Defaults to “gaussian”. See the help file for *evm* for more information.
The function `mex` works as follows. First, Generalized Pareto distributions (GPD) are fitted to the upper tails of each of the marginal distributions of the data: the GPD parameters are estimated for each column of the data in turn, independently of all other columns. Then, the conditional multivariate approach of Heffernan and Tawn is used to model the dependence between variables. The returned object is of class "mex".

This function is a wrapper for calls to `migpd` and `mexDependence`, which estimate parameters of the marginal and dependence components of the Heffernan and Tawn model respectively. See documentation of these functions for details of modelling issues including the use of penalties / priors, threshold choice and checking for convergence of parameter estimates.
The `plot` method produces diagnostic plots for the fitted dependence model described by Heffernan and Tawn, 2004. The plots are best viewed by using the plotting area split by `par(mfcol=c(.,..))` rather than `mfrow`, see examples below. Three diagnostic plots are produced for each dependent variable:

1) Scatterplots of the residuals $Z$ from the fitted model of Heffernan and Tawn (2004) are plotted against the quantile of the conditioning variable, with a lowess curve showing the local mean of these points. 2) The absolute value of $Z - \text{mean}(Z)$ is also plotted, again with the lowess curve showing the local mean of these points. Any trend in the location or scatter of these variables with the conditioning variable indicates a violation of the model assumption that the residuals $Z$ are independent of the conditioning variable. This can be indicative of the dependence threshold used being too low. 3) The final plots show the original data (on the original scale) and the fitted quantiles (specified by `quantiles`) of the conditional distribution of each dependent variable given the conditioning variable. A model that fits well will have good agreement between the distribution of the raw data (shown by the scatter plot) and the fitted quantiles. Note that the raw data are a sample from the joint distribution, whereas the quantiles are those of the estimated conditional distribution given the value of the conditioning variable, and while these two distributions should move into the same part of the sample space as the conditioning variable becomes more extreme, they are not the same thing!

The `predict` method for `mex` works as follows. The returned object has class "predict.mex". Simulated values of the dependent variables are created, given that the conditioning variable is above its 100th quantile. If `predict.mex` is passed an object object of class "mex" then the simulated values are based only on the point estimate of the dependence model parameters, and the original data. If `predict.mex` is passed an object object of class "bootmex" then the returned value additionally contains simulated replicate data sets corresponding to the bootstrap model parameter estimates. In both cases, the simulated values based on the original data and point estimates appear in component `object$data$simulated`. The simulated data from the bootstrap estimates appear in `object$replicates`.

The `plot` method for class "predict.mex" displays both the original data and the simulated data generated above the threshold for prediction; it shows the threshold for prediction (vertical line) and also the curve joining equal quantiles of the marginal distributions – this is for reference: variables that are perfectly dependent will lie exactly on this curve.

Value

A call to `mex` returns an list of class `mex` containing the following three items:

- `margins` An object of class `migpd`.
- `dependence` An object of class `mexDependence`.
- `call` This matches the original function call.

There are `plot`, `summary`, `coef` and `predict` methods for this class.

A call to `predict.mex` does the importance sampling for prediction, and returns a list of class "predict.mex" for which there are print and plot methods available. The summary method for this class of object is intended to be used following a call to the predict method, to estimate quantiles or probabilities of threshold exceedances for the fitted conditional distributions given the conditioning variable above the threshold for prediction. See examples below.

There are `print`, `summary` and `plot` methods available for the class "predict.mex".
Note

The package `texmex` is equipped to fit GPD models to the upper marginal tails only, not the lower tails. This is appropriate for extrapolating into the tails of any dependent variable when dependence between this variable and the conditioning variable is positive. In the case of negative dependence between the conditioning variable and any dependent variable, estimation of the conditional distribution of the dependent variable for extreme values of the conditioning variable would naturally visit the lower tail of the dependent variable. Extrapolation beyond the range of the observed lower tail is not supported in the current version of `texmex`. In cases where negative dependence is observed and extrapolation is required into the lower tail of the dependent variable, the situation is trivially resolved by working with a reflection of the dependent variable (Y becomes -Y and so the upper and lower tails are swapped). Results can be calculated for the reflected variable then reflected back to the correct scale. This is satisfactory when only the pair of variables (the conditioning and single dependent variable) are of interest, but when genuine multivariate (as opposed to simply bivariate) structure is of interest, this approach will destroy the dependence structure between the reflected dependent variable and the remaining dependent variables.

Author(s)

Harry Southworth, Janet E. Heffernan

References


See Also

`migpd, mexDependence, bootmex`

Examples

```r
w <- mex(winter, mqu=.7)
w
par(mfcol=c(3, 2))
plot(w)

par(mfcol=c(2,2))
p <- predict(w)
summary(p)
summary(p, probs=c(0.01,0.2,0.5,0.8,0.99))
summary(p, probs=0.5,mth=c(40,50,150,25,50))
p
plot(p)
```
mexDependence

Estimate the dependence parameters in a conditional multivariate extreme values model

Description

Estimate the dependence parameters in a conditional multivariate extreme values model using the approach of Heffernan and Tawn, 2004.

Usage

mexDependence(x, which, dqu, margins="laplace", constrain=TRUE, v = 10, maxit=1000000, start=c(.01, .01), marTransform="mixture", nOptim = 1, PlotLikDo=FALSE, PlotLikRange=list(a=c(-1,1),b=c(-3,1)), PlotLikTitle=NULL)

Arguments

x
An object of class "migpd" as returned by migpd.

which
The name of the variable on which to condition. This is the name of a column of the data that was passed into migpd.

dqu
See documentation for this argument in mex.

margins
The form of margins to which the data are transformed for carrying out dependence estimation. Defaults to "laplace", with the alternative option being "gumbel". The choice of margins has an impact on the interpretation of the fitted dependence parameters. Under Gumbel margins, the estimated parameters a and b describe only positive dependence, while c and d describe negative dependence in this case. For Laplace margins, only parameters a and b are estimated as these capture both positive and negative dependence.

constrain
Logical value. Defaults to constrain=TRUE although this will subsequently be changed to FALSE if margins="gumbel" for which constrained estimation is not implemented. If margins="laplace" and constrain=TRUE then the dependence parameter space is constrained to allow only combinations of parameters which give the correct stochastic ordering between (positively and negatively) asymptotically dependent variables and variables which are asymptotically independent.

v
Scalar. Tuning parameter used to carry out constrained estimation of dependence structure under constrain=TRUE. Takes positive values greater than 1; values between 2 and 10 are recommended.

maxit
The maximum number of iterations to be used by the optimizer. Defaults to maxit = 1000000.

start
Optional starting value for dependence estimation. This can be: a vector of length two, with values corresponding to dependence parameters a and b respectively, and in which case start is used as a starting value for numerical estimation of each of the dependence models to be estimated; a matrix with two
rows corresponding to dependence parameters a and b respectively and number of columns equal to the number of dependence models to be estimated (the ordering of the columns will be as in the original data matrix); or a previously estimated object of class "mex" whose dependence parameter estimates are used as a starting point for estimation. Note that under constrain=TRUE, if supplied, start must lie within the permitted area of the parameter space.

marTransform Optional form of transformation to be used for probability integral transform of data from original to Gumbel or Laplace margins. Takes values marTransform="mixture" (the default) or marTransform="empirical". When marTransform="mixture", the rank transform is used below the corresponding GPD fitting threshold used in x, and the fitted gpd tail model is used above this threshold. When marTransform="empirical" the rank transform is used for the entire range of each marginal distribution.

nOptim Number of times to run optimiser when estimating dependence model parameters. Defaults to 1. In the case of nOptim > 1 the first call to the optimiser uses the value start as a starting point, while subsequent calls to the optimiser are started at the parameter value to which the previous call converged.

PlotLikDo Logical value: whether or not to plot the profile likelihood surface for dependence model parameters under constrained estimation.

PlotLikRange This is used to specify a region of the parameter space over which to plot the profile log-likelihood surface. List of length 2; each item being a vector of length two corresponding to the plotting ranges for dependence parameters a and b respectively. If this argument is not missing, then PlotLikDo is set equal to TRUE.

PlotLikTitle Used only if PlotLikDo=TRUE. Character string. Optional title added to the profile log-likelihood surface plot.

... Further arguments to be passed to methods.

Details

Estimates the extremal dependence structure of the data in x. The precise nature of the estimation depends on the value of margins. If margins="laplace" (the default) then dependence parameters a and b are estimated after transformation of the data to Laplace marginal distributions. These parameters can describe both positive and negative dependence. If margins="gumbel" then the parameters a, b, c and d in the dependence structure described by Heffernan and Tawn (2004) are estimated in the following two steps: first, a and b are estimated; then, if a=0 and b is negative, parameters c and d are estimated (this is the case of negative dependence). Otherwise c and d will be fixed at zero (this is the case of positive dependence).

If margins="laplace" then the option of constrained parameter estimation is available by setting argument constrain=TRUE. The default is to constrain the values of the parameters (constrain=TRUE). This constrained estimation ensures validity of the estimated model, and enforces the consistency of the fitted dependence model with the strength of extremal dependence exhibited by the data. More details are given in Keef et al. (2013). The effect of this constraint is to limit the shape of the dependence parameter space so that its boundary is curved rather than following the original box constraints suggested by Heffernan and Tawn (2004). The constraint brings with it some performance issues for the optimiser used to estimate the dependence parameters, in particular sensitivity to choice of starting value which we describe now.
The dependence parameter estimates returned by this function can be particularly sensitive to the choice of starting value used for the optimisation. This is especially true when `margins="laplace"` and `constrain=TRUE`, in which case the maximum of the objective function can lie on the edge of the (possibly curved) constrained parameter space. It is therefore up to the user to check that the reported parameter estimates really do correspond to the maximum of the profile likelihood surface. This is easily carried out by using the visual diagnostics invoked by setting `PlotLikDo=TRUE` and adjusting the plotting area by using the argument `PlotLikRange` to focus on the region containing the surface maximum. See an example below which illustrates the use of this diagnostic.

Value

An object of class `mex` which is a list containing the following three objects:

- margins: An object of class `migpd`.
- dependence: An object of class `mexDependence`.
- call: This matches the original function call.

Author(s)

Harry Southworth, Janet E. Heffernan

References


See Also

`migpd`, `bootmex`, `predict.mex`, `plot.mex`

Examples

data(winter)
mygpd <- migpd(winter, mqu=.7, penalty="none")
mexDependence(mygpd, which = "NO", dqu=.7)

# focus on 2-d example with parameter estimates on boundary of constrained parameter space:
NO.NO2 <- migpd(winter[,2:3], mqu=.7, penalty="none")

# starting value gives estimate far from true max:
mexDependence(NO.NO2, which = "NO", dqu=0.7, start=c(0.01,0.01),
PlotLikDo=TRUE,PlotLikTitle=c("NO2 | NO"))

# zoom in on plotting region containing maximum:
mexDependence(NO.NO2, which = "NO", dqu=0.7, start=c(0.01,0.01),
PlotLikDo=TRUE,PlotLikTitle=c("NO2 | NO"),
PlotLikRange = list(a=c(0,0.8),b=c(-0.2,0.6)))
mexRangeFit

Estimate dependence parameters in a conditional multivariate extreme values model over a range of thresholds.

Description

Diagnostic tool to aid the choice of threshold to be used for the estimation of the dependence parameters in the conditional multivariate extreme values model of Heffernan and Tawn, 2004.

Usage

mexRangeFit(x, which, quantiles = seq(0.5, 0.9, length = 9), start = c(0.1, 0.1), 
R = 10, nPass = 3, trace = 10, margins = "laplace", constrain = TRUE, v = 10)

Arguments

x An object of class mex or migpd.
which The variable on which to condition.
quantiles A numeric vector specifying the quantiles of the marginal distribution of the conditioning variable at which to fit the dependence model.
start See documentation for this argument in mexDependence.
R The number of bootstrap runs to perform at each threshold. Defaults to R=10.
nPass Argument passed to function bootmex.
trace Argument passed to function bootmex.
margins Argument passed to function mexDependence.
constrain Argument passed to function mexDependence.
v Argument passed to function mexDependence.
... Further graphical parameters may be passed, which will be used for plotting.

Details

Dependence model parameters are estimated using a range of threshold values. The sampling variability of these estimates is characterised using the bootstrap. Point estimates and bootstrap estimates are finally plotted over the range of thresholds. Choice of threshold should be made such that the point estimates at the chosen threshold and beyond are constant, up to sampling variation.

Value

NULL.
40

**Author(s)**

Harry Southworth, Janet E. Heffernan

**References**


**See Also**

mexDependence, bootmex

**Examples**

```r
# Example commented out to reduce R CMD check time
# w <- migpd(winter, mqu=.7)
# w
# par(mfrow=c(4,2))
# mexRangeFit(w,which=1,main="Winter data, Heffernan and Tawn 2004",cex=0.5)
```

---

**migpd**

*Fit multiple independent generalized Pareto models*

**Description**

Fit multiple independent generalized Pareto models as the first step of conditional multivariate extreme values modelling following the approach of Heffernan and Tawn, 2004.

**Usage**

```r
migpd(data, mth, mqu, penalty = "gaussian", maxit = 10000, 
trace = 0, verbose = FALSE, priorParameters = NULL)
## S3 method for class 'migpd' 
plot(x, main=c("Probability plot","Quantile plot", 
  "Return level plot","Histogram and density"), 
  xlab=rep(NULL,4), nsim=1000, alpha=.05, ...) 
```

**Arguments**

- **data**
  A matrix or data.frame, each column of which is to be modelled.

- **mth**
  Marginal thresholds. Thresholds above which to fit the models. Only one of `mth` and `mqu` should be supplied. Length one (in which case a common threshold is used) or length equal to the number of columns of `data` (in which case values correspond to thresholds for each of the columns respectively).

- **mqu**
  Marginal quantiles. Quantiles above which to fit the models. Only one of `mth` and `mqu` should be supplied. Length as for `mth` above.
penalty  How the likelihood should be penalized. Defaults to "gaussian". See documentation for evm.
maxit  The maximum number of iterations to be used by the optimizer.
trace  Whether or not to tell the user how the optimizer is getting on. The argument is passed into optim – see the help for that function.
verbose  Controls whether or not the function prints to screen every time it fits a model. Defaults to FALSE.
priorParameters  Only used if penalty = 'gaussian'. A named list, each element of which contains two components: the first should be a vector of length 2 corresponding to the location of the Gaussian distribution; the second should be a 2x2 matrix corresponding to the covariance matrix of the distribution. The names should match the names of the columns of data. If not provided, it defaults to independent priors being centred at zero, with variance 10000 for log(sigma) and 0.25 for xi. See the details section.
x  Object of class migpd as returned by function migpd.
main  Character vector of length four: titles for plots produced by plot method.
xlab  As main but for x-axes labels.
nsim  Number of simulations on which to base tolerance envelopes in plot method.
alpha  Significance level for tolerance and confidence intervals in plot method.
...  Further arguments to be passed to methods.

Details

The parameters in the generalized Pareto distribution are estimated for each column of the data in turn, independently of all other columns. Note, covariate modelling of GPD parameters is not supported.

Maximum likelihood estimation often fails with generalized Pareto distributions because of the likelihood becoming flat (see, for example, Hosking et al, 1985). Therefore the function allows penalized likelihood estimation, which is the same as maximum a posteriori estimation from a Bayesian point of view.

By default quadratic penalization is used, corresponding to using a Gaussian prior. If no genuine prior information is available, the following argument can be used. If xi = -1, the generalized Pareto distribution corresponds to the uniform distribution; and if xi is 1 or greater, the expectation is infinite. Therefore, xi is likely to fall in the region (-1, 1). A Gaussian distribution centred at zero and with standard deviation 0.5 will have little mass outside (-1, 1) and so will often be a reasonable prior for xi. For log(sigma) a Gaussian distribution, centred at zero and with standard deviation 100 will often be vague. If a Gaussian penalty is specified but no parameters are given, the function will assume such independent priors.

Note that internally the function works with log(sigma), not sigma. The reasons are that quadratic penalization makes more sense for phi=log(sigma) than for sigma (because the distribution of log(sigma) will be more nearly symmetric), and because it was found to stabilize computations.

The associated coef, print and summary functions exponentiate the log(sigma) parameter to return results on the expected scale. If you are accessing the parameters directly, however, take care to be sure what scale the results are on.

Threshold selection can be carried out with the help of functions mrl and gpdRangeFit.
Value

An object of class "migpd". There are coefficients, print, plot and summary functions available.

Author(s)

Harry Southworth

References


See Also

mex, mexDependence, bootmex, predict.mex, gpdRangeFit, mrl

Examples

mygpd <- migpd(winter, mqu=.7, penalty = "none")
mygpd
summary(mygpd)
plot(mygpd)

migpdCoefs

Change values of parameters in a migpd object

Description

Change the values of parameters in a migpd object. You might want to do this after modelling marginal distributions as functions of covariates.

Usage

migpdCoefs(object, which, coefs)

Arguments

object An object of class migpd.
which Which models in the migpd object you want to change.
coefs The coefficients that you want to change to. If which has length 1, coefs can be a vector of parameters. Otherwise, it should be a list of vectors, and the list should have the same length as which
### Value

A `migpd` object. See the help for `migpd`.

### Author(s)

Harry Southworth

### See Also

`migpd`

### Examples

```r
library(MASS)
liver <- liver
liver$dose <- as.numeric(liver$dose)
d <- data.frame(alt = resid(rlm(log(ALT.M) ~ log(ALT.B) + ndose, data=liver)),
                ast = resid(rlm(log(AST.M) ~ log(AST.B) + ndose, data=liver)),
                alp = resid(rlm(log(ALP.M) ~ log(ALP.B) + ndose, data=liver)),
                tbl = resid(rlm(log(TBL.M) ~ log(TBL.B) + ndose, data=liver)))

Dgpds <- migpd(d[liver$dose == "D", 1:4], mqu=.7)

d$dose <- liver$dose
galt <- evm(alt, data=d, qu=.7, xi = - ndose)
gast <- evm(ast, data=d, qu=.7, xi = - ndose)
galp <- evm(alp, data=d, qu=.7, xi = - ndose)

altco <- predict(galt,type="lp",newdata=data.frame(ndose=4))$link[1:2]
astco <- predict(gast,type="lp",newdata=data.frame(ndose=4))$link[1:2]
alpco <- predict(galp,type="lp",newdata=data.frame(ndose=4))$link[1:2]

Dgp <- migpdCoefs(Dgpds, which=c("alt", "ast", "alp"),
                   coefs=list(altco, astco, alpco))

summary(Dgp)
summary(Dgpds)
```

---

### Description

Calculate mean residual life and plot it to aid the identification of a threshold over which to fit a generalized Pareto distribution.
Usage

mrl(data, umin = min(data), umax = max(data) - 0.1, nint = 100, alpha=.050)
## S3 method for class 'mrl'
print(x, ...)
## S3 method for class 'mrl'
summary(object, ...)
## S3 method for class 'mrl'
plot(x, xlab="Threshold", ylab="Mean excess", ...)

Arguments

data A numeric vector.

umin The minimum value over which to threshold the data.

umax The maximum value over which to threshold the data.

alpha Used to determine coverage of confidence interval to plot. Defaults to plotting a
95% interval.

nint The number of points at which to compute the plot.

x, object Arguments to print, summary and plot functions.

xlab Label for the x-axis. Defaults to xlab="Threshold".

ylab Label for the y-axis. Defaults to ylab="Mean excess".

... Optional arguments to plot

Details

Threshold choice for the fitting of the GPD is guided by the shape of the Mean Residual Life plot. A threshold which is suitably high will have a corresponding mrl plot which is approximately linear in shape above the threshold (up to sampling variation).

Value

A list with two components. data is the original data, mrl is a matrix containing information to produce the mean residual life plot.

Note

The function was originally written by Stuart Coles and appears in the ismev package. This version modified by Harry Southworth to allow more control over the appearance of the plot.

Author(s)

Janet E. Heffernan, Harry Southworth

References

predict.evmOpt

Predict return levels from extreme value models, or obtain the linear predictors.

Description

Predict return levels from extreme value models, or obtain the linear predictors.

Usage

```r
## S3 method for class 'evmOpt'
predict(object, M = 1000, newdata = NULL,
type = "return level", se.fit = FALSE,
        ci.fit = FALSE, alpha = 0.05, unique. = TRUE,...)

## S3 method for class 'evmSim'
predict(object, M = 1000, newdata = NULL,
type = "return level", se.fit = FALSE,
        ci.fit = FALSE, alpha = 0.050, unique. = TRUE, all = FALSE,
        sumfun = NULL,...)

## S3 method for class 'evmBoot'
predict(object, M = 1000, newdata = NULL,
type = "return level", se.fit = FALSE,
        ci.fit = FALSE, alpha = 0.050, unique. = TRUE, all = FALSE,
        sumfun = NULL,...)

linearPredictors(object, newdata = NULL,
        se.fit = FALSE, ci.fit = FALSE,
        alpha = 0.050, unique. = TRUE, ...)

## S3 method for class 'evmOpt'
linearPredictors(object, newdata = NULL,
        se.fit = FALSE, ci.fit = FALSE,
        alpha = 0.05, unique. = TRUE, full.cov = FALSE,...)

## S3 method for class 'evmSim'
linearPredictors(object, newdata = NULL,
        se.fit = FALSE, ci.fit = FALSE,
        alpha = 0.050, unique. = TRUE, all = FALSE, sumfun = NULL,...)

## S3 method for class 'evmBoot'
linearPredictors(object, newdata = NULL,
        ci.fit = FALSE, alpha = 0.050, unique. = TRUE,
        all = FALSE, sumfun = NULL,...)

## S3 method for class 'lp.evmOpt'
print(x, digits=3,...)

## S3 method for class 'lp.evmSim'
print(x, digits=3,...)

## S3 method for class 'lp.evmBoot'
print(x, digits=3,...)
```
predict.evmOpt

## Arguments

- `object`: An object of class `evmOpt`, `evmSim` or `evmBoot`.
- `newdata`: The new data that you want to make the prediction for. Defaults in `newdata = NULL` in which case the data used in fitting the model will be used. Column names must match those of the original data matrix used for model fitting.
- `type`: For the predict methods, the type of prediction, either "return level" (or "rl") or "link" (or "lp"). Defaults to `type = "return level"`. When a return level is wanted, the user can specify the associated return period via the `M` argument. If `type = "link"` the linear predictor(s) for `phi` and `xi` (or whatever other parameters are in your `texmexFamily` are returned.
- For the plot methods for simulation based estimation of underlying distributions i.e. objects derived from "evmSim" and "evmBoot" classes, whether to use the sample median `type="median"` or mean `type="mean"` estimate of the parameter.
- `se.fit`: Whether or not to return the standard error of the predicted value. Defaults to `se.fit = FALSE` and is not implemented for `predict.evmSim` or `predict.evmBoot`.
- `ci.fit`: Whether or not to return a confidence interval for the predicted value. Defaults to `ci.fit = FALSE`. For objects of class `evmOpt`, if set to `TRUE` then the confidence interval is a simple symmetric confidence interval based on the estimated approximate standard error. For the `evmSim` and `evmBoot` methods, the confidence interval represents quantiles of the simulated distribution of the parameters.
- `M`: The return level: units are number of observations. Defaults to `M = 1000`. If a vector is passed, a list is returned, with items corresponding to the different values of the vector `M`.
- `alpha`: If `ci.fit = TRUE`, a `100(1 - alpha)%` confidence interval is returned. Defaults to `alpha = 0.05`.
- `unique`: If `unique. = TRUE`, predictions for only the unique values of the linear predictors are returned, rather than for every row of `newdata`. Defaults to `unique. = TRUE`. 
Worker functions not intended for direct use.
predictWorkers

Usage

addCov(res, X)
texmexMakeParams(co, data)
texmexMakeCovariance(cov, data)
texmexPredictSE(x)
texmexMakeCICI(params, ses, alpha)
texmexMakeNewdataD(x, newdata)
texmexMakeCISim(x, alpha, object, sumfun, M)

Arguments

res The actual predictions to have covariates added.
x A design matrix.
co Model coefficients.
data A list of design matrices.
cov Full covariance matrix for all parameters.
x A covariance matrix. Otherwise an "evm" object.
params Matrix of parameter values.
eses Standard errors.
alpha 100(1-alpha)% confidence intervals will be produced.
newdata Data for which predictions are required.
object An "evm" object.
sumfun A function for summarizing data.
M The return level required.

Details

These are functions used for processing information by predict.evm and are not intended for end-user usage.

Value

Processed data and other information.

Author(s)

Harry Southworth
rain, wavesurge and portpirie

Rain, wavesurge and portpirie datasets.

Description

Rainfall, wave-surge and Port Pirie data sets.

Usage

data(rain)
data(wavesurge)
data(portpirie)

Format

The format of the rain data is: num [1:17531] 0 2.3 1.3 6.9 4.6 0 1.5 1.8 1.8 ...
The wave-surge data is bivariate and is used for testing functions in texmex.
The Port Pirie data has two columns: 'Year' and 'SeaLevel'.

Details

The datasets are used by Coles and appear in the ismev package.

Source

Copied from the ismev package.

References


r1

Return levels

Description

Computation of return levels and confidence intervals for extreme value models.
Usage

```r
rl(object, M = 1000, newdata = NULL, se.fit = FALSE, ci.fit = FALSE,
    alpha = 0.050, unique. = TRUE, ...)  
```

```r
## S3 method for class 'evmOpt'
rl(object, M = 1000, newdata = NULL, se.fit = FALSE, ci.fit = FALSE,
    alpha = 0.050, unique. = TRUE, ...)  
```

```r
## S3 method for class 'evmSim'
rl(object, M = 1000, newdata = NULL, se.fit = FALSE, ci.fit = FALSE,
    alpha = 0.050, unique. = TRUE, all = FALSE, sumfun = NULL, ...)  
```

```r
## S3 method for class 'evmBoot'
rl(object, M = 1000, newdata = NULL, se.fit = FALSE, ci.fit = FALSE,
    alpha = 0.050, unique. = TRUE, all = FALSE, sumfun = NULL, ...)  
```

```r
## S3 method for class 'rl.evmOpt'
plot(x, xlab, ylab, main, pch=1, ptcol =2 , cex=.75, linecol = 4 ,
    cicol = 0, polycol = 15, smooth = FALSE, sameAxes=TRUE, type="median", ...)  
```

```r
## S3 method for class 'rl.evmSim'
plot(x, xlab, ylab, main, pch=1, ptcol =2 , cex=.75, linecol = 4 ,
    cicol = 0, polycol = 15, smooth = FALSE, sameAxes=TRUE, type="median", ...)  
```

```r
## S3 method for class 'rl.evmBoot'
plot(x, xlab, ylab, main, pch=1, ptcol =2 , cex=.75, linecol = 4 ,
    cicol = 0, polycol = 15, smooth = FALSE, sameAxes=TRUE, type="median", ...)  
```

```r
## S3 method for class 'rl.evmOpt'
print(x, digits=3, ...)  
```

```r
## S3 method for class 'rl.evmSim'
print(x, digits=3, ...)  
```

```r
## S3 method for class 'rl.evmBoot'
print(x, digits=3, ...)  
```

```r
## S3 method for class 'rl.evmOpt'
summary(object, digits=3, ...)  
```

```r
## S3 method for class 'rl.evmSim'
summary(object, digits=3, ...)  
```

```r
## S3 method for class 'rl.evmBoot'
summary(object, digits=3, ...)  
```

Arguments

- **object**  
  An object of class evmOpt, evmSim or evmBoot.

- **M**  
  The M-observation return level is computed by the function. Defaults to $M = 1000$.

- **newdata**  
  Data from which to calculate the return level. If not provided, the original data
  used to fit the model is used. Column names must match those of original data
  matrix used for model fitting.

- **se.fit**  
  Whether or not to return the standard error of the predicted value. Defaults to
  `se.fit = FALSE`. 
ci.fit Whether or not to return a confidence interval for the predicted value. Defaults to ci.fit = FALSE. For objects of class evmOpt, if set to TRUE then the confidence interval is a simple symmetric confidence interval based on the estimated approximate standard error. For the evmSim and evmBoot methods, the confidence interval represents quantiles of the simulated distribution of the parameters.

alpha If ci.fit = TRUE, a 100(1 - alpha)% confidence interval is returned. Defaults to alpha = 0.050.

unique If unique = TRUE, predictions for only the unique values of the linear predictors are returned, rather than for every row of the original dataframe or of newdata if this latter is specified. Defaults to unique = TRUE.

all For the evmSim and evmBoot methods, if all = TRUE, the predictions are returned for every simulated parameter vector. Otherwise, only a summary of the posterior/bootstrap distribution is returned. Defaults to all = FALSE.

sumfun For the evmSim and evmBoot methods, a summary function can be passed in. If sumfun = FALSE, the default, the summary function used returns the estimated mean and median, and quantiles implied by alpha.

type For calls to plot methods for objects of class rl.evmSim or rl.evmBoot, specifies whether to use the sample mean (type=“mean”) or median (type=“median”) estimate of the return levels.

x Object passed to plot and print methods.

xlab, ylab, main, pch, ptcol, cex, linecol, cicol, polycl, smooth, sameAxes Further arguments to plot methods.

digits Number of digits to show when printing output.

... Further arguments to be passed to methods.

Details

The M-observation return level is defined as the value that is expected to be exceeded only once every M observations. Thus, it is an estimate of a high quantile of the fitted distribution.

In models fit by the evm family of functions with family=gpd, only a fraction of the data is actually included in the model; the fitted GPD is a conditional model, conditioning on the threshold having been exceeded. This consideration is taken into account by rl which calculates unconditional return levels from the entire distribution of observations above and below the GPD fitting threshold.

Examples

```
mod <- evm(rain, qu=.8) # daily rainfall observations
rl(mod, M=100*365) # 100-year return level
```
summer and winter data

Air pollution data, separately for summer and winter months

Description

Air pollution data from Leeds (U.K.) city centre, collected from 1994 to 1998. The summer data set corresponds to the months of April to July inclusive. The winter data set corresponds to the months of November to February inclusive. Some outliers have been removed, as discussed by Heffernan and Tawn, 2004.

Usage

data(summer)
data(winter)

Format

Data frames with 578 (summer) and 532 (winter) observations on the following 5 variables.

- O3  Daily maximum ozone in parts per billion.
- NO2  Daily maximum NO2 in parts per billion.
- NO  Daily maximum NO in parts per billion.
- SO2  Daily maximum SO2 in parts per billion.
- PM10  Daily maximum PM10 in micrograms/metre^3

Source

Provided as online supplementary material to Heffernan and Tawn, 2004:
http://www.blackwellpublishing.com/rss/Readmefiles/heffernan.htm

References


Examples

data(summer)
data(winter)
Description

Internal functions used by the texmex package.

Usage

```r
## S3 method for class 'evmOpt'
hist(x, xlab, ylab, main, ...)  
qqevm(object, nsim = 1000, alpha = 0.05)  
ppevm(object, nsim = 1000, alpha = 0.05)  
qgpd2(N, sigma = 1, xi = 1, u = 0, la = 1)  
u2gpd(u, p=1, th=0, sigma, xi)  
mexTransform(x, method = "mixture", divisor = "n+1", na.rm=TRUE, margins="laplace")  
revTransform(x, data, qu, th=0, sigma=1, xi=0, method="mixture")  
evmFit(data, family, ..., prior="none", start=NULL, priorParameters = NULL, maxit = 10000, trace = 0, hessian = TRUE)  
gpd.info(o, method="observed")  
ConstraintsAreSatisfied(a,b,z,zpos,zneg,v)  
PosGumb.Laplace.negloglik(yex, ydep, a, b, m, s, constrain, v, aLow)  
PosGumb.Laplace.negProfileLogLik(yex, ydep, a, b, constrain, v, aLow)  
namesBoot2sim(bootobject)  
getPlotRLdata(object, alpha, RetPeriodRange)  
plotRLevm(M,xm,polycol,cicol,linecol,ptcol,n,xdat,pch,smooth,xlab,ylab,main,xrange,yrange)  
plotRL1.evmOpt(object, alpha = 0.05, xlab, ylab, main, pch = 1, ptcol = 2, cex = 0.75, linecol = 4, cicol = 0, polycol = 15, smooth = FALSE, RetPeriodRange = NULL)  
rFrechet(n)  
rMaxAR(n,theta)  
addCoefficients(o)  
addCovariance(o, family, cov)  
constructEVM(o, family, th, rate, prior, modelParameters, call, modelData, data, priorParameters, cov)  
texmexPst(msg, Family)  
```

Arguments

- `x`, `object`, `data`  
  Object to be used by plot functions, vector to be converted.
- `xlab`, `ylab`, `main`, `pch`, `cex`, `linecol`, `polycol`, `RetPeriodRange`, ...
  Arguments to plot functions.
- `N`  
  Number of observations corresponding to N-observation return level to be calculated.
modelData List containing response data and design matrices.

la Rate at which threshold is exceeded.

alpha Control nominal coverage of confidence intervals. Defaults to alpha = 0.05.

nsim Number of simulated datasets to use in computing confidence intervals.

u Uniform deviates to be converted to GPD deviates.

p, th, qu, sigma, xi Parameters of GPD distribution.

method Argument to `mexTransform`: how to convert. When method = "mixture", the upper tail of the distribution is modelled using a generalized Pareto distribution and the remainder is approximated using the empirical distribution. Also argument to `gpd.info` which currently does nothing.

divisor Divisor used in estimation of empirical distribution.

na.rm Whether or not to remove missing values.

margins Form of margins to which to transform x. Can take values margins="laplace" or margins="gumbel".

start, priorParameters, maxit, trace Arguments supplied to `gpd`, `migpd` or `mex`, or inferred from those functions after some preprocessing.

hessian Argument passed to `optim`. Logical.

o An object of class `evmOpt`, or one containing elements of such an object.

a, b Dependence parameters of the Heffernan and Tawn dependence model.

m, s Nuisance parameters of the Heffernan and Tawn dependence model.

z, zpos, zneg Quantiles of the residuals under the fitted Heffernan and Tawn model, asymptotic positive dependence and asymptotic negative dependence respectively.

v Positive scalar, tuning parameter for constrained estimation of Heffernan and Tawn dependence model under estimation with Laplace marginal distributions.

constrain Logical. Whether to carry out estimation of Heffernan and Tawn model parameters under correct stochastic ordering of fitted model and asymptotic positive/negative dependence models.

aLow Lower bound for dependence parameter a. This depends on the marginal distribution under which the dependence model is being fitted. Under Gumbel margins, the lower bound is 0 and under Laplace margins, the lower bound is -1.

yex, ydep Data for model estimation: yex is the explanatory variable on which the model conditions, and ydep is the dependent variable.

bootobject Argument to `namesBoot2bgpd` which restructures an object of class `evmBoot` to resemble one of class `evmSim`, which can then use methods for the `evmSim` class.

M, x, cicol, pcol, n, xdat, smooth, xrange, yrange Arguments to `plotRLgpd` which is a worker function, does the actual plotting for `plotRLevm`, `plotRLevmSim`, `plotRLevmBoot`. 
theta  Argument to rFrechet and rMaxAR, the dependence parameter theta. Takes values between 0 and 1, with 0 corresponding to perfect dependence and 1 to independence.

family, rate, prior, modelParameters, call, cov
Information used to obtain various components of an object of class 'evmOpt'.

msg, Family  A message to print and a Family

Details

None of these functions are intended to be used explicitly.
The plotting functions are used internally by plot.evmOpt.

Some of the code is based on code that appears in the ismev package, originally written by Stuart Coles, the evd package by Alec Stephenson and extRemes package by Eric Gilleland, Rick Katz and Greg Young.

Code to carry out estimation of H+T2004 under Laplace margins and constrained estimation was written by Yiannis Papastathopoulos, and is used here for validation purposes.

Author(s)

Harry Southworth, Janet E. Heffernan, Yiannis Papastathopoulos.

texmexFamily  Create families of distributions

Description

Create families of distributions for use with extreme value modelling.

Usage

texmexFamily(name, log.lik, param, info = NULL, start = NULL, resid = NULL, r1, delta, endpoint, density, rng, prob, quant)
## S3 method for class 'texmexFamily'
print(x, ...)
## S3 method for class 'texmexFamily'
summary(object, ...)

Arguments

name  The name of the distribution.
log.lik  The distribution’s log-likelihood function.
param  The names of the parameters in the model.
info  Function to compute the information matrix. If not provided, the modelling functions will work with a numerical approximation.
start Function to compute starting parameters for the model. If not provided, the modelling functions will try to guess.

resid Function to compute residuals for the model.

rl Function to compute return levels.

delta Function to compute adjustments for covariance for return levels.

endpoint Function to compute the upper or lower endpoint of the fitted distribution.

density Function to compute the density.

rng Function for random number generation.

prob Function to compute cumulative probabilities.

quant Function to compute quantiles.

... Additional arguments to the print and summary methods.

x, object An object of class ’texmexFamily’.

details

The density, rng, prob and quant functions can be simple wrappers for the usual d, r, p and q functions. They should take a matrix with number of columns equal to the number of parameters, and a fitted model object even if the model object is not used by the function.

Examples of "texmexFamily" objects are gpd and gev. Take a look at those objects to see how the functions should be constructed.

The functions are used by the modelling functions to create diagnostic plots, predictions, etc..

Value

A object of class "texmexFamily", which is essentially a list containing the input arguments. If info, start, resid are not provided, they default to NULL.

Note

The gpd and gev families are provided. The evm function defaults to using the gpd family.

Author(s)

Harry Southworth

See Also

evm
texmexWorkers Worker functions for texmex

Description
Worker functions not intended for direct use.

Usage

texmexMethod(method)
texmexPrior(prior, penalty, method, pp)
texmexTrace(trace, method)
texmexPrepareData(y, data, params)
texmexThresholdData(threshold, data)
texmexPriorParameters(prior, priorParameters, data)
findFormulae(call, ...)
texmexParameters(call, fam, ...)
texmexGetParam(data, co)

Arguments

method Character string describing the method to use. The function attempts to interpret it and returns "o", "s" or "b".
prior Character string describing the type of prior distribution to use. The function attempts to interpret it and returns a standardized version.
penalty Character string describing the type of penalty to use.
pp The priorParameters object.
trace Information on whether to report progress to the user.
y Vector of data to be modelled.
data A data.frame containing covariates. Otherwise a list of data.frames already preprocessed.
params Names of the parameters in the model.
threshold The threshold above which to model the data.
priorParameters A list containing information on the prior distribution.
call An object of class "call".
fam An object of class "texmexFamily".
co Model coefficients.
... Other arguments being passed between functions.

Details
These are functions used for processing user input and data prior to calling evm.fit and are not intended for end-user usage.
**Description**

Process observations from Metropolis fitting of extreme value models, to thin the output and discard observations from burn-in period.

**Usage**

```r
## S3 method for class 'evmSim'
thinAndBurn(object, burn, thin)
```

**Arguments**

- `object`: Object of class `evmSim` as returned by `evm` called with `method="simulate"`.  
- `thin`: `thin` or its reciprocal must be a positive integer.  If integer valued, this specifies the frequency of observations from the simulated Markov Chain which will be retained.  If specified as a proportion, this is the proportion of values which will be retained.  For no thinning use `thin=1`.  
- `burn`: The number of observations from the simulated Markov Chain to be discarded as burn-in.  Must be a non-negative integer, for no burn-in use `burn=0`.

**Value**

Object of class `evmSim`.  See Value returned by `evm` using `method = "simulate"` for details.  

Note that the original chain is not discarded when this function is called: `thinAndBurn` can be called recursively on the original object with different values of `burn` and `thin` without the object getting progressively smaller!

**Author(s)**

Harry Southworth, Janet E. Heffernan

**See Also**

`evm`
Examples

```r
x <- rnorm(1000)
# For the values of burn and thin below, we should do many more iterations.
# The number of iterations is kept low here due to the run time allowed
# by CRAN.
mod <- evm(x, qu=.7, method="sim", iter=1000)
mod
par(mfrow=c(3, 2))
plot(mod)
mod1 <- thinAndBurn(mod,burn=1000, thin=5)
plot(mod1)
```
Index

*Topic datasets
  liver, 27
  rain, wavesurge and portpirie, 49
  r1, 49
  summer and winter data, 52

*Topic methods
  predict.evmOpt, 45

*Topic models
  bootmex, 5
  dgev, 11
  evm, 14
  gpdRangeFit, 25
  mex, 31
  mexDependence, 36
  mexRangeFit, 39
  migpd, 40
  mrl, 43
  predictWorkers, 47
  texmex-package, 3
  texmexFamily, 55

*Topic multivariate
  bootmex, 5
  chi, 7
  copula, 10
  MCS, 28
  mex, 31
  mexDependence, 36
  mexRangeFit, 39
  migpd, 40
  migpdCoeffs, 42
  texmex-package, 3
  texmexWorkers, 57

*Topic package
  texmex-package, 3

*Topic univar
  edf, 13

addCoefficients (texmex-internal), 53
addCov (predictWorkers), 47
addCovariance (texmex-internal), 53
AIC.evmOpt (evm), 14
bootExtremalIndex (extremalIndex), 21
bootMCS (MCS), 28
bootmex, 3, 5, 7, 35, 38–40, 42
chi, 7, 29
coeff.evmBoot (methods), 30
coeff.evmOpt (evm), 14
coeff.evmSim (methods), 30
coeff.mex (methods), 30
coeff.migpd (methods), 30
coefficients.evmBoot (methods), 30
coefficients.evmOpt (methods), 30
coefficients.evmSim (methods), 30
coefficients.mex (methods), 30
coefficients.migpd (methods), 30
ConstraintsAreSatisfied (texmex-internal), 53
constructEVM (texmex-internal), 53
copula, 10, 13
declust, 3
declust (extremalIndex), 21
dgev, 11
dgpd, 12
edf, 10, 13
endPoint, 13
evm, 3, 14, 21, 24, 26, 32, 41, 56, 58
evm.declustered, 20
evm.declustered (extremalIndex), 21
evmBoot (evm), 14
evmFit (texmex-internal), 53
evmSim (evm), 14
evmSimSetSeed, 20
extremalIndex, 21
extremalIndexRangeFit (extremalIndex), 21
INDEX

print.rl.evmBoot(rl), 49
print.rl.evmOpt(rl), 49
print.rl.evmSim(rl), 49
print.summary.evmBoot(methods), 30
print.summary.evmOpt(methods), 30
print.summary.evmSim(methods), 30
print.summary.predict.mex(methods), 30
print.texmexFamily(texmexFamily), 55
qgev(dgev), 11
qgpd(dgpd), 12
qgpd2(texmex-internal), 53
qqevm(texmex-internal), 53
rain(rain, wavesurge and portpirie), 49
rain, wavesurge and portpirie, 49
rank, 9, 13
revTransform(texmex-internal), 53
rFrechet(texmex-internal), 53
rgev(dgev), 11
rgpd(dgpd), 12
rl, 49
rl.evmOpt, 20
rMaxAR(texmex-internal), 53
signif, 14
summary.bootMCS(MCS), 28
summary.chi(chi), 7
summary.copula(methods), 30
summary.evmBoot(evm), 14
summary.evmOpt(evm), 14
summary.evmSim(evm), 14
summary.gpdRangeFit(gpdRangeFit), 25
summary.lp.evmBoot(predict.evmOpt), 45
summary.lp.evmOpt(predict.evmOpt), 45
summary.lp.evmSim(predict.evmOpt), 45
summary.MCS(methods), 30
summary.mex(methods), 30
summary.migpd(methods), 30
summary.mrl(mrl), 43
summary.predict.mex(mex), 31
summary.rl.evmBoot(rl), 49
summary.rl.evmOpt(rl), 49
summary.rl.evmSim(rl), 49
summary.texmexFamily(texmexFamily), 55
summer(summer and winter data), 52
summer and winter data, 52
texmex(texmex-package), 3
texmex-internal, 53
texmex-package, 3
texmexCheckMap(evm), 14
texmexFamily, 55
texmexGetParam(texmexWorkers), 57
texmexJumpConst(evm), 14
texmexMakeCI(predictWorkers), 47
texmexMakeCISim(predictWorkers), 47
texmexMakeCovariance(predictWorkers), 47
texmexMakeNewdataD(predictWorkers), 47
texmexMakeParams(predictWorkers), 47
texmexMethod(texmexWorkers), 57
texmexMetropolis(evm), 14
texmexParameters(texmexWorkers), 57
texmexPredictSE(predictWorkers), 47
texmexPrepareData(texmexWorkers), 57
texmexPrior(texmexWorkers), 57
texmexPriorParameters(texmexWorkers), 57
texmexPst(texmex-internal), 53
texmexThresholdData(texmexWorkers), 57
texmexTrace(texmexWorkers), 57
texmexWorkers, 57
thinAndBurn, 58
u2gpd(texmex-internal), 53
wavesurge(rain, wavesurge and portpirie), 49
winter(summer and winter data), 52