

# Package ‘texmex’

October 5, 2018

**Type** Package

**Title** Statistical Modelling of Extreme Values

**Version** 2.4.2

**Date** 2018-10-05

**Author** Harry Southworth [aut, cre],  
Janet E. Heffernan [aut],  
Paul D. Metcalfe [aut],  
Yiannis Papastathopoulos [ctb],  
Alec Stephenson [ctb],  
Stuart Coles [ctb]

**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 (2003) <doi:10.1111/1467-9868.00401> 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 (2004) <doi:10.1111/j.1467-9868.2004.02050.x>, with graphical tools for threshold selection and to diagnose estimation convergence.

**License** GPL (>= 2)

**Depends** mvtnorm, ggplot2, stats

**Suggests** MASS, gridExtra, parallel, lattice, knitr, testthat,  
devtools, survival, ismev

**Imports** Rcpp (>= 0.12.18)

**LinkingTo** Rcpp

**LazyLoad** yes

**LazyData** yes

**URL** <https://github.com/harrysouthworth/texmex>

**RoxygenNote** 6.1.0

**Encoding** UTF-8

**Collate** 'AIC.evm.R' 'Dcond.R' 'Profile\_likelihood\_HT\_unc.R'  
 'Profile\_likelihood\_cd\_nm\_joint\_D\_KT.R'  
 'Profile\_likelihood\_cd\_nm\_joint\_D\_KT\_neg.R' 'RcppExports.R'  
 'bootmex.R' 'gpd.sandwich.R' 'gpd.info.R' 'texmexFamily.R'  
 'ggpd.R' 'chi.R' 'coef.evmSim.R' 'coefficients.evm.R'  
 'coefficients.migpd.R' 'constructObject.R' 'copulas.R'  
 'degp3.R' 'dgev.R' 'dgpd.R' 'efficient.closures.R' 'egp3.R'  
 'egp3RangeFit.R' 'endPoint.R' 'estimate\_HT.R'  
 'estimate\_HT\_KPT\_joint\_posneg\_nm.R' 'evm.R' 'evm.simSetSeed.R'  
 'evmBoot.R' 'evmFit.R' 'evmSim.R' 'extremalIndex.R' 'gev.R'  
 'ggplot.bayes.R' 'ggplot.boot.R' 'ggplot.cluster.R'  
 'ggplot.evm.R' 'ggplot.mex.R' 'ggplot.migpd.R'  
 'ggplot.predict.R' 'ggplot.thresh.R' 'globalVariables.R'  
 'gpd.R' 'gpdProfileLikRetLevels.R' 'gpdRangeFit.R' 'gumbel.R'  
 'hist.evm.R' 'initial\_posneg.R' 'jointExceedanceCurves.R'  
 'mex.R' 'mexDependence.R' 'mexDependenceLowLevelFunctions.R'  
 'mexMonteCarlo.R' 'mexRangeFit.R' 'mexTransform.R' 'migpd.R'  
 'migpdCoefs.R' 'mrl.R' 'mspearman.R' 'multivariate.cauchy.R'  
 'pegp3.R' 'pgev.R' 'pgpd.R' 'plot.bootmex.R' 'plot.evm.R'  
 'plot.evmSim.R' 'plot.mex.R' 'plot.mexPrediction.R'  
 'plot.migpd.R' 'plot.predict.link.evm.R' 'plotrl.evm.R'  
 'ppevm.R' 'predict.evm.R' 'predict.mex.R' 'predictWorkers.R'  
 'print.bootmex.R' 'print.evm.R' 'print.evmSim.R'  
 'print.mexDependence.R' 'print.mexPrediction.R' 'print.migpd.R'  
 'profile\_minmax\_joint\_posneg\_KT.R' 'qegp3.R' 'qgev.R' 'qgpd.R'  
 'qgpd2.R' 'qqevm.R' 'rMaxAR.R' 'regp3.R' 'residuals.evm.R'  
 'revTransform.R' 'rgev.R' 'rgpd.R' 'roots.R' 'simulate.R'  
 'sombrero-internal.R' 'summary.evm.R' 'summary.evmSim.R'  
 'summary.migpd.R' 'summary.predict.mex.R' 'texmex-package.R'  
 'texmexWorkers.R' 'thinAndBurn.evm.sim.R' 'u2gpd.R'  
 'weibull.info.R' 'weibull.R'

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2018-10-05 15:50:03 UTC

## R topics documented:

texmex-package . . . . .	4
.exprel . . . . .	6
.log1mexp . . . . .	7
.log1prel . . . . .	7
.specfun.safe.product . . . . .	8
addExcesses . . . . .	8

AIC.evmOpt . . . . .	9
bootmex . . . . .	10
chi . . . . .	12
copula . . . . .	15
degp3 . . . . .	16
dgev . . . . .	17
dgpd . . . . .	18
edf . . . . .	19
egp3RangeFit . . . . .	20
endPoint . . . . .	22
evm . . . . .	22
evmBoot . . . . .	28
evmSim . . . . .	29
evmSimSetSeed . . . . .	31
extremalIndex . . . . .	31
ggplot.declustered . . . . .	36
ggplot.evmBoot . . . . .	36
ggplot.evmOpt . . . . .	37
ggplot.evmSim . . . . .	38
ggplot.mex . . . . .	39
ggplot.migpd . . . . .	44
ggplot.rl.evmOpt . . . . .	47
gpd.prof . . . . .	47
gpdRangeFit . . . . .	48
JointExceedanceCurve . . . . .	50
liver . . . . .	52
logLik.evmOpt . . . . .	53
makeReferenceMarginalDistribution . . . . .	54
MCS . . . . .	54
mexDependence . . . . .	57
mexMonteCarlo . . . . .	60
mexRangeFit . . . . .	61
migpdCoefs . . . . .	63
mrl . . . . .	64
plot.copula . . . . .	66
plot.evmOpt . . . . .	66
plot.evmSim . . . . .	67
plot.lp.evmOpt . . . . .	68
plot.rl.evmOpt . . . . .	70
print.evmOpt . . . . .	72
rain, wavesurge and portpirie . . . . .	73
rFrechet . . . . .	74
rMaxAR . . . . .	74
simulate.evmOpt . . . . .	75
summer and winter data . . . . .	76
texmexFamily . . . . .	77
thinAndBurn . . . . .	78

---

texmex-package

*Extreme value modelling*

---

## Description

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

## Details

Package: texmex  
Type: Package  
Version: 2.3  
Date: 2014-03-03  
License: GPL (>=2) | BSD

The packages 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’.

**evm**: 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 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>

URL: <https://github.com/harrysouthworth/texmex>

## References

J. E. Heffernan and J. A. Tawn, A conditional approach for multivariate extreme values, Journal of the Royal Statistical society B, 66, 497 – 546, 2004.

C.A.T Ferro and J. Segers, Inference for Clusters of Extreme Values, Journal of the Royal Statistical society B, 65, 545 – 556, 2003.

**Examples**

```

# Analyse the winter data used by Heffernan and Tawn
mymex <- mex(winter, mqu = .7, penalty="none", dqu=.7, which = "NO")
plot(mymex)
# Only do 10 replicates to keep CRAN checks 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$ALP.M > 1,] # Get rid of outlier
liver$ndose <- 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
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)$obj$link[[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

```

---

```
.exprel
```

*Accurately compute  $(\exp(x) - 1) / x$*

---

### Description

Accurately compute  $(\exp(x) - 1) / x$

### Usage

```
.exprel(x)
```

### Arguments

x                    numeric vector

### Value

numeric vector

---

.log1mexp                      *Accurately compute  $\log(1-\exp(x))$*

---

**Description**

Accurately compute  $\log(1-\exp(x))$

**Usage**

.log1mexp(x)

**Arguments**

x                      numeric vector

**Value**

a numeric vector

---

.log1prel                      *Accurately compute  $\log(1+x)/x$*

---

**Description**

Accurately compute  $\log(1+x)/x$

**Usage**

.log1prel(x)

**Arguments**

x                      numeric vector

**Value**

numeric vector

---

```
.specfun.safe.product
```

*Compute  $p_{\max}(x, y, -1)$  in such a way that zeros in  $x$  beat infinities in  $y$ .*

---

### Description

This is a common pattern in much of the distribution code, so it's worth factoring out.

### Usage

```
.specfun.safe.product(x, y)
```

### Arguments

<code>x</code>	a numeric vector
<code>y</code>	a numeric vector

### Value

an appropriate numeric vector

---

```
addExcesses
```

*Annotate a threshold selection ggplot*

---

### Description

Annotate a threshold selection ggplot with the number of exceedances of various thresholds.

### Usage

```
addExcesses(p, x, y, data, textsize)
```

### Arguments

<code>p</code>	An object produced by ggplot
<code>x</code>	Horizontal axis data containing the full range.
<code>y</code>	Verticle axis data containing the full range.
<code>data</code>	The actual data being considered for GPD modelling.
<code>textsize</code>	The size of the text in the annotations.



AIC.evmOpt

*Information Criteria***Description**

Compute AIC and (approximate) DIC for evmOpt objects

**Usage**

```
## S3 method for class 'evmOpt'
AIC(object, penalized = FALSE, nsamp = 1000, ...,
     k = 2)
```

**Arguments**

object	fit model object
penalized	whether to use the penalized log-likelihood
nsamp	Number of approximate Gaussian sample to use in computing DIC. Defaults to nsamp=1e3. Only used when the object has class 'evmOpt'.
...	other arguments currently ignored
k	numeric, the penalty per parameter to be used; the default k = 2 is the classical AIC.

**Details**

If the object has class 'evmOpt', nsamp random draws are made from the Gaussian distribution with mean and covariance inferred from the model object. The result will be an approximate DIC. Note that AIC should not be trusted if priors are not flat. For example, if you use a regularizing prior on  $\xi$ , say  $\xi \sim N(0, 0.25)$ , AIC can be misleading and DIC should be preferred. If the object has class 'evmSim', the actual posterior draws are used in the computation. Also note that sometimes the optimizer returns an approximate covariance that is not positive-semidefinite, in which case the DIC will be reported as NA.

**Value**

The AIC and DIC

**See Also**

[AIC](#)

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,referenceMargin=NULL)

## 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 <a href="#">mex</a> .
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.
referenceMargin	Optional set of reference marginal distributions to use for marginal transformation if the data's own marginal distribution is not appropriate (for instance if only data for which one variable is large is available, the marginal distributions of the other variables will not be represented by the available data). This object can be created from a combination of datasets and fitted GPDs using the function <code>makeReferenceMarginalDistribution</code> .
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 `migpd` 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**

J. E. Heffernan and J. A. Tawn, A conditional approach for multivariate extreme values, *Journal of the Royal Statistical society B*, 66, 497 – 546, 2004

**See Also**

[migpd](#), [mexDependence](#) , [bootmex](#), [predict.mex](#).

## Examples

```
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

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

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

## S3 method for class 'summary.chi'
print(x, digits=3, ...)

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

## S3 method for class 'chi'
plot(x, show=c("Chi"=TRUE,"ChiBar"=TRUE), lty=1,
      cilty=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)), ylabChiBar = expression(bar(chi)(u)),
      ask, ...)

## S3 method for class 'chi'
ggplot(data=NULL, mapping, xlab = "Quantile",
        ylab=c("ChiBar" = expression(bar(chi)(u)), "Chi" = expression(chi(u))),
        main=c("ChiBar" = "Chi Bar", "Chi" = "Chi"),
        xlim = c(0, 1), ylim =list("Chi" = c(-1, 1),"ChiBar" = c(-1, 1)),
        ptcol="blue",fill="orange",show=c("ChiBar"=TRUE,"Chi"=TRUE),
        spcases = TRUE,plot., ..., environment)
```

**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 <code>trunc = TRUE</code> .
x, object	An object of class <code>chi</code> .
digits	Number of digits for printing.
show	Logical, of length 2, names "Chi" and "ChiBar". Defaults to <code>c("Chi" = TRUE, "ChiBar" = TRUE)</code> .
lty, cilty, col, cicol	Line types and colours for the the estimated quantities and their confidence intervals.
xlim, ylimChi, ylimChiBar	Limits for the axes.
mainChi, mainChiBar	Main titles for the plots.
xlab, ylabChi, ylabChiBar	Axis labels for the plots.
mapping, ylab, main, ylim, ptcoll, fill, environment	Arguments to <code>ggplot</code> methods.
spcases	Whether or not to plot special cases of perfect (positive and negative) dependence and independence. Defaults to <code>FALSE</code> .
plot.	whether to plot to active graphics device.
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` and `ggplot` methods show 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.

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

**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**

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

A. G. Stephenson. `evd`: Extreme Value Distributions, *R News*, 2, 2002.

**See Also**

[MCS](#), [rank](#)

**Examples**

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

---

copula	<i>Calculate the copula of a matrix of variables</i>
--------	--

---

### Description

Returns the copula of several random variables.

### Usage

```
copula(x, na.last = NA, ...)  
  
## Default S3 method:  
copula(x, na.last = NA, ...)  
  
## S3 method for class 'data.frame'  
copula(x, na.last = NA, ...)  
  
## S3 method for class 'matrix'  
copula(x, na.last = NA, ...)
```

### Arguments

x	A matrix or data.frame containing numeric variables.
na.last	How to treat missing values. See rank for details.
...	further arguments

### Details

The result is obtained by applying [edf](#) to each column of x in turn.

Print and plot 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.

### Methods (by class)

- default: default method
- data.frame: data frame method
- matrix: matrix method

### Author(s)

Harry Southworth

**See Also**

[edf plot.copula](#)

**Examples**

```
D <- liver[liver$dose == "D",]
Dco <- copula(D)
plot(Dco)
```

---

degp3

*Density, cumulative density, quantiles and random number generation for the extended generalized Pareto distribution 3*

---

**Description**

Density, cumulative density, quantiles and random number generation for the EGP3 distribution of Papastathopoulos and Tawn

**Usage**

```
degp3(x, kappa = 1, sigma, xi, u = 0, log.d = FALSE)
```

```
pegp3(q, kappa = 1, sigma, xi, u = 0, lower.tail = TRUE,
      log.p = FALSE)
```

```
regp3(n, kappa = 1, sigma, xi, u = 0)
```

**Arguments**

x, q, p	Value, quantile or probability respectively.
kappa	The power parameter (Papastathopoulos and Tawn call it the shape parameter and call what we call the shape parameter the tail index.)
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.
n	Number of random numbers to simulate.

**Author(s)**

Harry Southworth



## References

I. Papastathopoulos and J. A. Tawn, Extended generalized Pareto models for tail estimation, *Journal of Statistical Planning and Inference*, 143, 131 – 143, 2013

## Examples

```
x <- regp3(1000, kappa=2, sigma=1, xi=.5)
hist(x)
x <- regp3(1000, kappa=2, sigma=exp(rnorm(1000, 1, .25)), xi=rnorm(1000, .5, .2))
hist(x)
plot(pegp3(x, kappa=2, sigma=1, xi=.5))
```

---

dgev	<i>Density, cumulative density, quantiles and random number generation for the generalized extreme value distribution</i>
------	---

---

## Description

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.
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.
n	Number of random numbers to simulate.

## 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))
```

---

dgpd

*Density, cumulative density, quantiles and random number generation  
for the generalized Pareto distribution*


---

**Description**

Density, cumulative density, quantiles and random number generation for the generalized Pareto distribution

**Usage**

```
dgpd(x, sigma, xi, u = 0, log.d = FALSE)
pgpd(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**

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

**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(pgpd(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 <a href="#">rank</a> for details.

**Value**

A vector of quantiles relating to the observations in x.

**Author(s)**

Harry Southworth

**See Also**

[copula](#)

**Examples**

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

---

egp3RangeFit	<i>Estimate the EGP3 distribution power parameter over a range of thresholds</i>
--------------	--

---

### Description

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

### Usage

```
egp3RangeFit(data, umin=quantile(data, .05), umax=quantile(data,
.95), nint = 10, penalty = "gaussian", priorParameters = NULL, alpha=0.05)
## S3 method for class 'egp3RangeFit'
print(x, ...)
## S3 method for class 'egp3RangeFit'
plot(x, xlab = "Threshold", ylab = "kappa", main = NULL, addNexcesses=TRUE, log="", ...)
## S3 method for class 'egp3RangeFit'
ggplot(data, mapping, xlab = "Threshold", ylab = expression(kappa),
main=NULL,fill="orange", col="blue",addNexcesses=TRUE, textsize=4, ..., environment)
```

### 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 <a href="#">evm</a> for more information.
alpha	100(1 - alpha)% confidence intervals will be plotted with the point estimates. Defaults to alpha = 0.05.
x	Argument to the print functions.
xlab	Label for the x-axis.
ylab	Label for the y-axis.
main	The main title.
textsize	Size of text for annotation showing number of threshold excesses.
addNexcesses	Annotate top axis with numbers of threshold excesses arising with the corresponding values of threshold on the bottom axis.
log.	Argument passed through to plot. Can take values "x" for plotting the x-axis on the log scale, "y" for plotting the y-axis on the log scale, "xy" for both, or "" (the default) for neither.

```
mapping, fill, col, environment
      Arguments to ggplot method.
...
      Arguments to plot.
```

## Details

Papastathopoulos and Tawn present 3 extended versions of the generalized Pareto distribution. Using the `egp3` texmex family object, the power parameter in the EGP3 distribution is estimated on the log scale, a confidence interval is calculated and the result is transformed back to the scale of the power parameter and returned to the user.

When the power parameter,  $\kappa$ , is equal to 1, the EGP3 distribution is identical to the generalized Pareto distribution. Therefore, the plot of the estimated parameter over a range of thresholds provides a diagnostic for threshold selection: the lowest value of  $\kappa$  whose confidence interval includes 1 is suggested as the threshold for generalized Pareto modelling.

If lower thresholds are used and the EGP3 distribution itself is used for modelling, some care should be taken to ensure the model provides a reasonable degree of fit to the data. Limited experience suggests that such models seldom fit well and the main value of the EGP3 distribution is as a diagnostic for threshold selection as described here.

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

## Author(s)

Harry Southworth

## References

I. Papastathopoulos and J. A. Tawn, Extended generalized Pareto models for tail estimation, *Journal of Statistical Planning and Inference*, 143, 131 – 143, 2013

## See Also

[evm](#), [gpdRangeFit](#), [mrl](#)

## Examples

```
erf <- egp3RangeFit(rain)
plot(erf)
ggplot(erf)
```

---

endPoint	<i>Calculate upper end point for a fitted extreme value model</i>
----------	---

---

**Description**

Calculate upper end point for fitted extreme value model

**Usage**

```
endPoint(y, verbose=TRUE, .unique=TRUE, ...)  
  
## 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 <code>evmOpt</code> or <code>evmSim</code> , as returned by <code>evm</code> .
verbose	Whether to print output.
.unique	Whether or not to use only unique values of y.
...	further arguments to be passed to the <code>signif</code> 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	<i>Extreme value modelling</i>
-----	--------------------------------

---

**Description**

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

**Usage**

```
evm(y, data, family = gpd, ...)

## 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 = 1000, cores = NULL, verbose = TRUE)
```

**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 (GPD) is fit to the data. Alternatively the family could be gev, weibull or gumbel, resulting in a generalized extreme value distribution, Weibull or Gumbell distribution being fit. Family cgpd fits the generalized Pareto distribution but with the shape parameter constrained to be > 0.5 by using the link function suggested by Yee and Stephenson (2007), $\eta = \log(\xi + 0.5)$ . Family egp3 fits the extended GP family 3 of Papastathopoulos and Tawn (2013). No other families are currently available in texmex, but users may write their own.
...	In evm, formulae for the parameters in the family, e.g. $\text{phi} = \sim x$ . If none are specified, they all default to $\sim 1$ .
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 $\text{quantile}(y, \text{qu})$ equals th.
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 priorParameters 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	<p>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. Alternatives are cov = "numeric" in which case a numerical approximation of the Hessian is used (see the help for optim), or cov = "sandwich" if the sandwich element of the texmexFamily object is implemented. The cov = "sandwich" method implements the Huber sandwich correction to the covariance matrix for data which are not independent and in which case the likelihood function no longer has the interpretation of a joint likelihood, but instead should be interpreted as a pseudo-likelihood.</p> <p>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.</p>
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	<p>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 <math>10^4</math>; all covariances are zero.</p>
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	Control parameter for the Metropolis algorithm.
R	The number of parametric bootstrap samples to run when method = "bootstrap" is requested. Defaults to 1000.
cores	The number of cores to use when bootstrapping. Defaults to cores=NULL and the function guesses how many cores are available and uses them all.



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

## Details

The main modelling function is `evm` (extreme value model) and the distribution to be used is specified by passing an object of class `texmexFamily` to the `family` argument.

The default `texmexFamily` object used by `evm` is `gpd`. Currently, the other `texmexFamily` objects available are `gev` which results in fitting a generalized extreme value (GEV) distribution to the data, and `egp3` which fits the extended generalized Pareto distribution version 3 of Papastathopoulos and Tawn (2013).

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 `evm`:

$$P(Y \leq y) = 1 - (1 + \xi y / \sigma)^{-1/\xi}$$

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

For the GEV model, we use:

$$P(Y \leq y) = \exp(-(1 + \xi(y - \mu)/\sigma)^{-1/\xi})$$

In each case, the scale parameter is  $\sigma$  and the shape parameter is  $\xi$ . The GEV distribution also has location parameter  $\mu$ . See Papastathopoulos and Tawn (2013) for specification of the EGP3 model.

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 = "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`:

<code>call</code>	The call to <code>evmSim</code> that produced the object.
<code>data</code>	The original data (above and below the threshold for fitting if a distribution for threshold excesses has been used). In detail, <code>data</code> is a list with elements <code>y</code> and <code>D</code> . <code>y</code> is the response variable and <code>D</code> is a list containing the design matrices implied by any formulae used in the call to <code>evm</code> .
<code>convergence</code>	Output from <code>optim</code> relating to whether or not the optimizer converged.
<code>message</code>	A message telling the user whether or not convergence was achieved.
<code>threshold</code>	The threshold of the data above which the <code>evmSim</code> model was fit.
<code>penalty</code>	The type of penalty function used, if any.
<code>coefficients</code>	The parameter estimates as computed under maximum likelihood or maximum penalized likelihood.
<code>rate</code>	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.
<code>priorParameters</code>	See above.
<code>residuals</code>	Residuals computed using the residual function in the <code>texmexFamily</code> object, if any. These are used primarily for producing QQ and PP plots via <code>plot.evmOpt</code> or <code>ggplot.evmOpt</code> . The residuals are transformed values of the raw data, accounting for the parameter estimates: see the <code>residuals</code> component of the <code>texmexFamily</code> object for the calculations. For the generalized Pareto family, they are (if the model fits well) standard exponential variates; for the GEV family, standard Gumbel variates.
<code>ploglik</code>	The value of the optimized penalized log-likelihood.
<code>loglik</code>	The value of the optimized (unpenalized) log-likelihood. If <code>penalty='none'</code> is used, this will be identical to <code>ploglik</code> , above.
<code>cov</code>	The estimated covariance of the parameters in the model.
<code>se</code>	The estimated standard errors of the parameters in the model.
<code>xlevels</code>	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 <code>predict</code> method to ensure all factor levels are known, even if they don't appear in <code>newdata</code> .

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

<code>call</code>	The call to <code>evmSim</code> that produced the object.
<code>threshold</code>	The threshold above which the model was fit.
<code>map</code>	The point estimates found by maximum penalized likelihood and which were used as the starting point for the Markov chain. This is of class <code>evmOpt</code> and methods for this class (such as <code>resid</code> and <code>plot</code> ) 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 <code>evmBoot</code> that produced the object.
replicates	The parameter estimates from the bootstrap fits.
map	The fit by by maximum penalized likelihood to the original data. This is of class <code>evmOpt</code> and methods for this class (such as <code>resid</code> and <code>plot</code> ) may be useful.

There are `summary`, `plot`, `print`, `residuals` 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). To constrain the parameter space of the GP shape parameter, use `family = cgpd` in the call to `evm` and the transformation  $\eta = \log(\xi + 0.5)$  is used, as suggested by Yee and Stephenson (2007).

### 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

- S. Coles. An Introduction to Statistical Modelling of Extreme Values. Springer, 2001.
- I. Papastathopoulos and J. A. Tawn, Extended generalised Pareto models for tail estimation, *Journal of Statistical Planning and Inference*, 143, 131 - 143, 2013.
- T. W. Yee and A. G. Stephenson, Vector generalized linear and additive extreme value models, *Extremes*, 10, 1 – 19, 2007.

### See Also

[plot.evmOpt](#) [ggplot.evmOpt](#) [rl.evmOpt](#), [predict.evmOpt](#), [evm.declustered](#).

**Examples**

```
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)
```

---

evmBoot

*Bootstrap an evmOpt fit*


---

**Description**

This runs a parametric bootstrap simulating from an optimized model.

**Usage**

```
evmBoot(o, R=1000, trace=100, cores=NULL, theCall)
## S3 method for class 'evmBoot'
summary(object,...)
## S3 method for class 'evmBoot'
plot(x,col=4,border=NULL,...)
## S3 method for class 'evmBoot'
coef(object,...)
## S3 method for class 'summary.evmBoot'
print(x,...)
## S3 method for class 'evmBoot'
print(x,...)
```

**Arguments**

o a fit evmOpt object

R	the number of parametric bootstrap samples to run
trace	the frequency of trace messages
cores	The number of cores to use when bootstrapping. Defaults to cores=NULL and the function guesses how many cores are available and uses them all.
theCall	(for internal use)
x	an <a href="#">evmBoot</a> object
col	colour used to fill histogram
border	the colour of the border around the bars
object	a <a href="#">evmBoot</a> object
...	other arguments passed to internal functions

**Value**

An object of class `evmBoot`; a list with

call	The call to <code>evmBoot</code> that produced the object.
replicates	The parameter estimates from the bootstrap fits.
map	The fit by maximum penalized likelihood to the original data.

**Note**

It is not expected that a user will need to call this function directly; you are directed to [evm](#).

**See Also**

[evm](#)

---

 evmSim

---

*MCMC simulation around an evmOpt fit*


---

**Description**

MCMC simulation around an `evmOpt` fit

**Usage**

```
evmSim(o, priorParameters, prop.dist, jump.const, jump.cov, iter, start,
       thin, burn, verbose, trace, theCall, ...)
```

**Arguments**

<code>o</code>	a fit <code>evmOpt</code> object
<code>priorParameters</code>	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 <code>method = "simulate"</code> then these represent the parameters in the Gaussian prior distribution. If <code>method = 'optimize'</code> 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.
<code>prop.dist</code>	The proposal distribution to use, either multivariate gaussian or a multivariate Cauchy.
<code>jump.const</code>	Control parameter for the Metropolis algorithm.
<code>jump.cov</code>	Covariance matrix for proposal distribution of Metropolis algorithm. This is scaled by <code>jump.const</code> .
<code>iter</code>	Number of simulations to generate
<code>start</code>	Starting values for the chain; if missing, defaults to the MAP/ML estimates in <code>o</code> .
<code>thin</code>	The degree of thinning of the resulting Markov chains.
<code>burn</code>	The number of initial steps to be discarded.
<code>verbose</code>	Whether or not to print progress to screen. Defaults to <code>verbose=TRUE</code> .
<code>trace</code>	How frequently to talk to the user
<code>theCall</code>	(internal use only)
<code>...</code>	ignored

**Value**

an object of class `evmSim`:

<code>call</code>	The call to <code>evmSim</code> that produced the object.
<code>threshold</code>	The threshold above which the model was fit.
<code>map</code>	The point estimates found by maximum penalized likelihood and which were used as the starting point for the Markov chain. This is of class <code>evmOpt</code> and methods for this class (such as <code>resid</code> and <code>plot</code> ) may be useful.
<code>burn</code>	The number of steps of the Markov chain that are to be treated as the burn-in and not used in inferences.
<code>thin</code>	The degree of thinning used.
<code>chains</code>	The entire Markov chain generated by the Metropolis algorithm.
<code>y</code>	The response data above the threshold for fitting.
<code>seed</code>	The seed used by the random number generator.
<code>param</code>	The remainder of the chain after deleting the burn-in and applying any thinning.

**Note**

it is not expected that the user should call this directly

---

evmSimSetSeed	<i>Set the seed from a fitted evmSim object.</i>
---------------	--

---

**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**

```
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	<i>Extremal index estimation and automatic declustering</i>
---------------	---

---

**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**

```

extremalIndex(y, data = NULL, threshold)

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

bootExtremalIndex(x)

declust(y, r=NULL, data = NULL, ...)

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

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

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

## S3 method for class 'extremalIndexRangeFit'
plot(x,addNexcesses=TRUE,estGPD=TRUE,...)

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

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

## S3 method for class 'extremalIndexRangeFit'
ggplot(data=NULL, mapping, xlab, ylab, main,
ylim = "auto",ptcol="dark blue",col="dark blue",fill="orange",
textsize=4,addNexcesses=TRUE,estGPD=TRUE,..., environment)

```

**Arguments**

y	Argument to function <code>extremalIndex</code> : either a numeric vector or the name of a variable in data.
data	A data frame containing y and any covariates. In <code>evm.declustered</code> , it should be NULL and is included to match the arguments of generic <code>evm</code> .
threshold	The threshold for y, exceedances above which will be used to estimate the extremal index and carry out automatic declustering.
family	The type of extreme value model. The user should not change this from its default in <code>evm.declustered</code> .
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 estimate the parameters.
umax	The maximum threshold above which to estimate 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 (ggplot).
ylab	Label for the y-axis (ggplot).
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.
mapping, main, ylim, ptc, col, fill, textsize, environment	Further arguments to ggplot method.
...	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 interexceedance 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. There are `plot` and `ggplot` methods for output of this function, which is of class `extremalIndexRangeFit`.

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":

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

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

<code>clusters</code>	integer labels assigning threshold exceedances to clusters
<code>sizes</code>	number of exceedances in each cluster
<code>clusterMaxima</code>	vector made up of the largest observation from each distinct cluster. In the case of ties, the first value is taken.
<code>isClusterMax</code>	logical; length equal to number of threshold exceedances, value is TRUE for threshold exceedances which correspond to cluster maxima
<code>y</code>	see entry for object of class "extremalIndex" above
<code>data</code>	see entry for object of class "extremalIndex" above
<code>threshold</code>	see entry for object of class "extremalIndex" above
<code>EIintervals</code>	see entry for object of class "extremalIndex" above
<code>call</code>	see entry for object of class "extremalIndex" above
<code>InterExceedTimes</code>	times between threshold exceedances, length is one less than the number of threshold exceedances
<code>InterCluster</code>	logical: indicates inter exceedance times larger than <code>r</code> the run length used for declustering
<code>thExceedances</code>	see entry for object of class "extremalIndex" above
<code>exceedanceTimes</code>	see entry for object of class "extremalIndex" above
<code>r</code>	run length used for declustering
<code>nClusters</code>	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

Ferro, C.A.T. and Segers, J., (2003) "Inference for clusters of Extreme Values", JRSS B 65, Part 2, pp 545–556.

### See Also

[evm](#)

### 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=~N0)
evm(so2.d,phi=~N02)
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)
```

---

`ggplot.declustered`      *Diagnostic plots for an declustered object*

---

### Description

Create and display diagnostic plots for a declustered object.

### Usage

```
## S3 method for class 'declustered'
ggplot(data = NULL, mapping, xlab, ylab, main,
        ptcol = c("blue", "orange"), col = "light blue", plot. = TRUE, ...,
        environment)

## S3 method for class 'extremalIndex'
ggplot(data = NULL, mapping, xlab, ylab, main,
        ptcol = "blue", col = "light blue", plot. = TRUE, ..., environment)
```

### Arguments

<code>data</code>	An object of class <code>declustered</code> or <code>extremalIndex</code> .
<code>mapping</code>	Not used.
<code>xlab</code>	Label for the x-axis.
<code>ylab</code>	Label for the y-axis.
<code>main</code>	Plot title.
<code>ptcol</code>	Colour for points. Defaults to <code>ptcol="blue"</code> .
<code>col</code>	Colour for lines. Defaults to <code>col="light blue"</code> .
<code>plot.</code>	Whether or not to display the output. Defaults to <code>plot.=TRUE</code> .
<code>...</code>	Other arguments passed through to underlying plot functions.
<code>environment</code>	Not used.

---

`ggplot.evmBoot`      *Diagnostic plots for the replicate estimated parameter values in an evmBoot object*

---

### Description

Diagnostic plots for the replicate estimated parameter values in an `evmBoot` object

**Usage**

```
## S3 method for class 'evmBoot'
ggplot(data = NULL, mapping, denscol = "light blue",
        histcol = "dark blue", linecol = "orange", plot.it = TRUE, ...,
        environment)
```

**Arguments**

<code>data</code>	An object of class 'evmBoot'.
<code>mapping, environment</code>	ignored
<code>denscol</code>	Colour for the densities. Defaults to 'light blue'.
<code>histcol</code>	Colour for the histograms. Defaults to 'dark blue'.
<code>linecol</code>	Colour for the point estimate lines. Defaults to 'orange'.
<code>plot.it</code>	Whether or not to actually print the plots. Defaults to <code>plot.it=TRUE</code> . If <code>plot.it=FALSE</code> , you might want to control the layout. Do this with <code>do.call("grid.arrange", c(plots, ncol=2))</code> , for example, where <code>plots</code> is the object returned by <code>ggplot.evmBoot</code> .
<code>...</code>	Additional arguments to <code>ggplot</code> , currently unused.

---

 ggplot.evmOpt

*Diagnostic plots for an evm object*


---

**Description**

Create and display diagnostic plots for an evm object. See [plot.evmOpt](#) for further details on what is being plotted.

**Usage**

```
## S3 method for class 'evmOpt'
ggplot(data, mapping, which = 1:4, main = rep(NULL,
  4), xlab = rep(NULL, 4), nsim = 1000, alpha = 0.05,
  jitter.width = 0, ptc = "blue", span = 2/3, col = "light blue",
  fill = "orange", plot. = TRUE, ncol = 2, nrow = 2, ...,
  environment)
```

**Arguments**

<code>data</code>	An object of class evm.
<code>mapping, environment</code>	ignored
<code>which</code>	Which plots to produce. Defaults to <code>which=1:4</code> .
<code>main</code>	Main titles. Should have length 4.
<code>xlab</code>	Labels for x-axes.

<code>nsim</code>	Number of simulated datasets to create to form tolerance regions.
<code>alpha</code>	Used to compute coverage of pointwise confidence intervals.
<code>jitter.width</code>	Used to control the amount of horizontal jittering of points in the plots of the residuals versus covariates (when covariates are in the model). Defaults to <code>jitter.width=0</code> .
<code>ptcol</code>	Colour for points. Defaults to <code>ptcol="blue"</code> .
<code>span</code>	Passed to the loess smoother and defaults to <code>span=2/3</code> . Sometimes this choice is poor: if the loess smoother looks wrong, try <code>span=1</code> .
<code>col</code>	Colour for lines. Defaults to <code>col="light blue"</code> .
<code>fill</code>	Colour for confidence regions. Defaults to <code>fill="orange"</code>
<code>plot.</code>	Whether or not to display the output. Defaults to <code>plot.=TRUE</code> . If the display doesn't have the desired row and column layout, the user should specify <code>plot.=FALSE</code> , assign the output to an object, and use <code>grid.arrange</code> to display it.
<code>ncol</code>	The number of columns wanted in the resulting plot. Defaults to <code>ncol=2</code> . This argument is passed into <code>grid.arrange</code> .
<code>nrow</code>	The number of rows wanted in the resulting plot. Defaults to <code>nrow=2</code> . This argument is passed into <code>grid.arrange</code> .
<code>...</code>	Other arguments passed through to underlying plot functions.

### Details

The function attempts to arrange the plots nicely. If the output isn't what was wanted, the function returns the graphs to the user as a list so that the user can use `grid.arrange` directly. Also, if you have one or more covariates in the model and the loess smoother looks wrong, try setting `span=1`.

### See Also

[plot.evmOpt](#)

---

`ggplot.evmSim`

*Diagnostic plots for the Markov chains in an evmSim object*

---

### Description

Diagnostic plots for the Markov chains in an evmSim object

### Usage

```
## S3 method for class 'evmSim'
ggplot(data = NULL, mapping, which.plots = 1:3,
        denscol = "dark blue", acfcol = "light blue", plot.it = TRUE, ...,
        environment)
```

**Arguments**

data	An object of class 'evmSim'.
mapping, environment	ignored
which.plots	Which plots to produce. Density plots correspond to 1, trace plots of the Markov chains to 2 and autocorrelation function plots to 3.
denscol	Colour for the density plots. Defaults to 'dark blue'.
acfcoll	Colour for the ACF plots. Defaults to 'light blue'.
plot.it	Whether or not to actually print the plots. Defaults to plot.it=TRUE. If plot.it=FALSE, you might want to control the layout. Do this with do.call("grid.arrange", c(plots, ncol=2)), for example, where plots is the object returned by ggplot.evmSim.
...	Additional arguments to ggplot, currently unused.

ggplot.mex

*Conditional multivariate extreme values modelling***Description**

Fit the conditional multivariate extreme value model of Heffernan and Tawn

**Usage**

```
## S3 method for class 'mex'
ggplot(data = NULL, mapping, pcol = "blue",
  col = "cornflowerblue", fill = "orange", plot. = TRUE,
  quantiles = seq(0.1, by = 0.2, len = 5), ..., environment)

mex(data, which, mth, mqu, dqu, margins = "laplace", constrain = TRUE,
  v = 10, penalty = "gaussian", maxit = 10000, trace = 0,
  verbose = FALSE, priorParameters = NULL)

mexAll(data, mqu, dqu)

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

## S3 method for class 'mex'
plot(x, quantiles = seq(0.1, by = 0.2, len = 5),
  col = "grey", ...)

## S3 method for class 'predict.mex'
plot(x, pch = c(1, 3, 20), col = c(2, 8, 3),
  cex = c(1, 1, 1), ask = TRUE, ...)

## S3 method for class 'predict.mex'
```

```

ggplot(data = NULL, mapping, xlab, ylab, main,
       ptcol = c("grey", "dark blue", "orange"), col = "dark blue",
       fill = "orange", shape = 16:18, size = rep(1, 3), plot. = TRUE,
       ..., environment)

## S3 method for class 'mex'
predict(object, which, pqu = 0.99, nsim = 1000,
       trace = 10, smoothZdistribution = FALSE, ...)

## S3 method for class 'predict.mex'
summary(object, mth, probs = c(0.05, 0.5, 0.95),
       ...)

```

### Arguments

<code>data</code>	A numeric matrix or data.frame, the columns of which are to be modelled.
<code>col</code>	In plot method for objects of class <code>mex</code> , the colour for points on scatterplots of residuals and original data respectively. In plot method for objects of class <code>predict.mex</code> , the colours of points for observed, and simulated data (conditioning variable not the largest) and simulated data (conditioning variable is the largest) respectively.
<code>quantiles</code>	A vector of quantiles taking values between 0 and 1 specifying the quantiles of the conditional distributions which will be plotted.
<code>...</code>	Further arguments to be passed to methods.
<code>which</code>	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.
<code>mth</code>	Marginal thresholds. In <code>mex</code> , 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 <code>data</code> .  In <code>summary.predict.mex</code> , 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.
<code>mqu</code>	Marginal quantiles As an alternative to specifying the marginal GPD fitting thresholds via <code>mth</code> , 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 <code>data</code> .
<code>dqu</code>	Dependence quantile. Used to specify the quantile at which to threshold the conditioning variable data when estimating the dependence parameters. For example <code>dqu=0.7</code> 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 <code>dqu=mqu[which]</code> the quantile corresponding to the threshold 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 <a href="#">mexDependence</a> .
constrain	See documentation for <a href="#">mexDependence</a> .
v	See documentation for <a href="#">mexDependence</a> .
penalty	How to penalize the likelihood when estimating the marginal generalized Pareto distributions. Defaults to "gaussian". See the help file for <a href="#">evm</a> for more information.
maxit	The maximum number of iterations to be used by the optimizer. defaults to maxit = 10000.
trace	Passed internally to <a href="#">optim</a> . Whether or not to inform the user of the progress of the optimizer. Defaults to 0, indicating no trace.
verbose	Whether or not to keep the user informed of progress. Defaults to verbose = FALSE.
priorParameters	Parameters of prior/penalty used for estimation of the GPD parameters. This is only used if penalty = "gaussian". It is a named list, each element of which contains two components: the first component should be a vector of length 2 corresponding to the location of the Gaussian distribution; the second 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, the default priors are independent normal, centred at zero, with variance 10000 for $\phi = \log(\sigma)$ and 0.25 for $\xi$ . See the details section.
x, object	Object of class mex or summary.mex as returned by these functions respectively.
pch, cex	Plotting characters: colours and symbol expansion. The observed and simulated data are plotted using different symbols, controlled by these arguments and col, each of which should be of length 2.
ask	Whether or not to ask before changing the plot. Defaults to ask = TRUE.
shape, size, mapping, ptc, fill, plot., environment, xlab, ylab, main	Further arguments to plot and ggplot methods.
pqu	Prediction quantile. Argument to predict method. The quantile of the conditioning variable above which it will be simulated for importance sampling based prediction. Defaults to pqu = .99 .
nsim	Argument to predict method. The number of simulated observations to be generated for prediction.
smoothZdistribution	In predict.mex, whether or not to sample from the smoothed distribution of the underlying residuals. Defaults to FALSE, in which case no smoothing is carried out. If TRUE then each margin of the underlying multivariate residual is smoothed independently, by using kernel smoothing with a normal kernel, and bandwidth chosen using the <a href="#">bw.nrd</a> function. This can be useful for removing "stripeyness" in importance samples which have few values in the conditional tails.
probs	In summary method for objects of class predict.mex: the quantiles of the conditional distribution(s) to calculate. Defaults to 5%, 50% and 95%.

## Details

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 100<sup>th</sup> 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. Original data are shown with one plotting character and simulated data with another; colours of simulated point distinguish those points which have the conditioning variable as the largest (on a quantile scale) or not the largest.

The function `mexAll` fits a collection of GPD and conditional dependence models, the same fitted GPD being used for all of the dependence model fits. This can be used in turn to generate Monte Carlo samples from the entire sample space using the collected dependence models.

**Value**

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

<code>margins</code>	An object of class <code>migpd</code> .
<code>dependence</code>	An object of class <code>mexDependence</code> .
<code>call</code>	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 excesses 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**

J. E. Heffernan and J. A. Tawn, A conditional approach for multivariate extreme values, *Journal of the Royal Statistical Society B*, 66, 497 - 546, 2004

**See Also**

[migpd](#), [mexDependence](#), [bootmex](#), [mexMonteCarlo](#)

**Examples**

```
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)
```

---

ggplot.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**

```
## S3 method for class 'migpd'
ggplot(data, mapping = NULL,
  main = c("Probability plot", "Quantile plot", "Return level plot",
  "Histogram and density"), xlab = rep(NULL, 4), nsim = 1000,
  alpha = 0.05, ..., environment)

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 = 0.05, ...)
```

**Arguments**

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

**mapping, environment** Further arguments to ggplot method.

**main** Character vector of length four: titles for plots produced by plot and ggplot methods.

xlab	As main but for x-axes labels.
nsim	Number of simulations on which to base tolerance envelopes in plot and ggplot methods.
alpha	Significance level for tolerance and confidence intervals in plot and ggplot methods.
...	Further arguments to be passed to methods.
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 <a href="#">evm</a> .
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 <a href="#">optim</a> – 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 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.

## 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 of  $(-1, 1)$  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 `coef`, `print`, `plot`, `ggplot` and `summary` functions available.

### Author(s)

Harry Southworth

### References

J. E. Heffernan and J. A. Tawn, A conditional approach for multivariate extreme values, *Journal of the Royal Statistical Society B*, 66, 497 – 546, 2004

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

### See Also

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

### Examples

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

---

ggplot.rl.evmOpt      *Plotting function for return level estimation*

---

**Description**

Plotting function for return level estimation

**Usage**

```
## S3 method for class 'rl.evmOpt'
ggplot(data = NULL, mapping, xlab, ylab, main,
        ylim = "auto", ptc col = "blue", col = "light blue",
        fill = "orange", alpha = 0.5, ..., environment)
```

**Arguments**

data	An object of class <code>rl.evmOpt</code> , <code>rl.evmBoot</code> , <code>rl.evmSim</code> , <code>lp.evmOpt</code> , <code>lp.evmBoot</code> or <code>lp.evmSim</code> .
mapping	Not used.
xlab	Label for the x-axis.
ylab	Label for the y-axis.
main	Plot title.
ylim	Plot limits for y-axis.
ptc col	Colour for points. Defaults to <code>ptc col="blue"</code> .
col	Colour for lines. Defaults to <code>col="light blue"</code> .
fill	Colour for shading polygons.
alpha	Transparency.
...	Other arguments passed through to underlying plot functions.
environment	Not used.

---

gpd.prof      *Profile likelihood based confidence intervals for GPD*

---

**Description**

Calculates profile likelihood based confidence intervals for a given fitted GPD model – this is only implemented for two parameter GPD with no covariates in the model.

**Usage**

```
gpd.prof(z, m, xmax, xlow, conf = 0.95, nint = 50, PlotIt = FALSE,
        mult = 2, priorParameters = NULL)
```

**Arguments**

<code>z</code>	a fitted <code>evmOpt</code> object
<code>m</code>	return period : units are number of observations
<code>xmax</code>	point estimate of the return level, this is used to bracket the roots of the equation used to calculate the ends of the profile likelihood based confidence interval. The value need not be exact.
<code>xlow</code>	value lower than the lower end of the confidence interval, for bracketing in root finding
<code>conf</code>	confidence level, defaults to 0.95
<code>nint</code>	used for plotting if required, number of points at which to calculate the profile likelihood for plotting, defaults to 50
<code>PlotIt</code>	logical, whether or not to plot the profile likelihood, defaults to FALSE
<code>mult</code>	used to calculate the starting point for the root finding for solving to find the upper end of the confidence interval. The starting point is <code>mult*xmax</code> minus the lower end point. If this starting point is beyond the estimated upper endpoint of the fitted distribution then this can cause an error, and the value of <code>mult</code> should be reduced
<code>priorParameters</code>	optional, value of prior/penalty parameters used for penalised likelihood estimation, default to NULL

**Value**

Numeric vector of length two, with lower and upper ends of the estimated confidence intervals respectively.

---

<code>gpdRangeFit</code>	<i>Estimate generalized Pareto distribution parameters over a range of values</i>
--------------------------	---

---

**Description**

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

**Usage**

```
gpdRangeFit(data, umin=quantile(data, .05), umax=quantile(data, .95),
nint = 10, penalty = "gaussian", priorParameters = NULL, alpha=0.05,
cov="observed")
## S3 method for class 'gpdRangeFit'
print(x, ...)
## S3 method for class 'gpdRangeFit'
summary(object, ...)
## S3 method for class 'summary.gpdRangeFit'
```



```

print(x, ...)
## S3 method for class 'gpdRangeFit'
plot(x, xlab = "Threshold", ylab = NULL, main = NULL, addNexcesses=TRUE, ...)
## S3 method for class 'gpdRangeFit'
ggplot(data, mapping, xlab="Threshold", ylab=NULL,
main=NULL, fill="orange", col="blue", addNexcesses = TRUE, textsize=4, ...,
environment)

```

## Arguments

<code>data</code>	The data vector to be modelled.
<code>umin</code>	The minimum threshold above which to estimate the parameters.
<code>umax</code>	The maximum threshold above which to estimate the parameters.
<code>nint</code>	The number of thresholds at which to perform the estimation.
<code>penalty</code>	The type of penalty to be used in the maximum penalized likelihood estimation. Should be either "gaussian" or "none". Defaults to "gaussian".
<code>priorParameters</code>	Parameters to be used for the penalty function. See the help for <a href="#">evm</a> for more information.
<code>alpha</code>	100(1 - alpha)% confidence intervals will be plotted with the point estimates. Defaults to alpha = 0.05.
<code>cov</code>	How to compute the covariance matrix of the parameters. Defaults to cov = "observed" in which case the observed information matrix is used, if the <code>info</code> element of the <code>texmexFamily</code> object is present. See more detailed documentation of this argument in <a href="#">evm</a> .
<code>x, object</code>	Arguments to print and summary functions.
<code>xlab</code>	Label for the x-axis.
<code>ylab</code>	Label for the y-axis.
<code>main</code>	The main title.
<code>addNexcesses</code>	Annotate top axis with numbers of threshold excesses arising with the corresponding values of threshold on the bottom axis.
<code>col</code>	Colour of the line on the threshold stability plot.
<code>fill</code>	Colour of the pointwise confidence region on the threshold stability plots.
<code>textsize</code>	Size of text on the plot (ggplot). Defaults to textsize=4.
<code>...</code>	Arguments to plot.
<code>mapping, environment</code>	Not used.

## 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**

```
par(mfrow=c(1,2))
plot(gpdRangeFit(rain))
```

---

JointExceedanceCurve    *Joint exceedance curves*

---

**Description**

Calculate bivariate joint exceedance curves

**Usage**

```
JointExceedanceCurve(Sample, ExceedanceProb,...)
## S3 method for class 'jointExcCurve'
print(x,...)

## Default S3 method:
JointExceedanceCurve(Sample, ExceedanceProb, n = 50,
  x = NULL, ...)

## S3 method for class 'mexMC'
JointExceedanceCurve(Sample, ExceedanceProb, n = 50,
  x = NULL, which = 1:2, ...)

## S3 method for class 'predict.mex'
JointExceedanceCurve(Sample, ExceedanceProb,
  n = 50, x = NULL, which = 1:2, ...)

calcJointExceedanceCurve(Sample, ExceedanceProb, n = 50, x = NULL)

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

geom_jointExcCurve(x, ...)
```

**Arguments**

Sample	Monte Carlo (or other) sample from which to calculate joint exceedance curve
ExceedanceProb	Takes values between 0 and 1, constant value of joint exceedance probability for which the curve will be calculated
...	Further arguments to be passed to methods
n	If x=NULL then this is HALF the number of points at which the curve will be estimated (ie the curve is calculated at 2n locations)
x	If specified by the user, the values of in the first dimension of Sample at which to calculate the curve. Defaults to NULL otherwise should be a numeric vector within the range of the first dimension of Sample.
which	Vector length two identifying which margins to use for joint exceedance curve estimation. Can be integer vector, giving column numbers of original data matrix, or character vector identifying variables by name (these must match column names in original data).

**Details**

Calculates pairs of points (x,y) for which the point exceedance probability  $P(X>x \text{ and } Y>y)$  is constant. This is available only in two dimensions: for higher dimensional data, the bivariate margin will be used and other variables ignored. Takes as input either a two column matrix of observations, output from `mexMonteCarlo` (in which case samples from all fitted models are used to calculate curves) or output from a call to the `predict` method for an object of class `mex` (in which case just the single fitted model is used for estimation, with the importance sample generated in the call to `predict` being used to calculate the joint exceedance curve).

**Value**

Returns an object of class `jointExcCurve`. This is a list of length two, one for each variable for which the curve is calculated. Each item of the list is a vector of coordinate values for the variable in question. Attributes include names and the exceedance probability used to calculate the curve `ExceedanceProb`.

The curve is calculated by finding pairs of points (x,y) for which the empirical probability  $P(X>x, Y>y)$  of both variables exceeding their corresponding value is equal to the specified `ExceedanceProb`. Note that when this is calculated for an object of class `predict.mex` (returned by a call to the `predict` method for an object of class `mex`) then the exceedance probability is interpreted as the UNCONDITIONAL exceedance probability of the importance sample, ie the probability of sampled values occurring from the original modelled joint distribution, and NOT the conditional distribution used to generate the importance sample.

Estimated curve can be added to a `ggplot` of the data (and/or importance sample) by using the function `geom_jointExcCurve`, see examples below.

**Examples**

```
# for data frame of raw data
m1 <- rmvnorm(5000, sigma=Sigma)
m1 <- as.data.frame(m1)
```

```

j1 <- JointExceedanceCurve(m1,0.01)
j2 <- JointExceedanceCurve(m1,0.005)
j3 <- JointExceedanceCurve(m1,0.001)
ggplot(m1,aes(V1,V2)) + geom_point(colour="dark blue",alpha=0.5) +
  geom_jointExcCurve(j1,colour="orange") +
  geom_jointExcCurve(j2,colour="orange") +
  geom_jointExcCurve(j3,colour="orange")

# using importance sample generated by call to predict for object of class mex
m <- mex(winter,mqu=0.7,dqu=0.7,which="NO")
m2 <- predict(m,nsim=5000,pqu=0.999)
g <- ggplot(m2,plot.=FALSE)
j4 <- JointExceedanceCurve(m2,0.0005,which=c("NO","NO2"))
j5 <- JointExceedanceCurve(m2,0.0002,which=c("NO","NO2"))
j6 <- JointExceedanceCurve(m2,0.0001,which=c("NO","NO2"))
g[[2]] +
  geom_jointExcCurve(j4,aes(NO,NO2),col="orange") +
  geom_jointExcCurve(j5,aes(NO,NO2),col="orange") +
  geom_jointExcCurve(j6,aes(NO,NO2),col="orange")

# for augmented dataset, generated by MC sampling from collection of fitted H+T models
m <- mexAll(winter,mqu=0.7,dqu=rep(0.7,5))
m3 <- mexMonteCarlo(nSample=5000,mexList=m)
j7 <- JointExceedanceCurve(m3,0.05,which=c("NO","NO2"))
j8 <- JointExceedanceCurve(m3,0.02,which=c("NO","NO2"))
j9 <- JointExceedanceCurve(m3,0.01,which=c("NO","NO2"))
ggplot(as.data.frame(m3$MCsample[,c("NO","NO2")]),aes(NO,NO2)) +
  geom_point(col="light blue",alpha=0.5) +
  geom_jointExcCurve(j7,col="orange") +
  geom_jointExcCurve(j8,col="orange") +
  geom_jointExcCurve(j9,col="orange")

```

---

liver

*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.

**ALPB** Alkaline phosphatase at baseline. A numeric vector.

**ALT.B** Alanine aminotransferase at baseline. A numeric vector.  
**AST.B** Aspartate aminotransferase at baseline. A numeric vector.  
**TBL.B** Total bilirubin at baseline. A numeric vector.  
**ALP.M** Alkaline phosphatase after treatment. A numeric vector.  
**ALT.M** Alanine aminotransferase after treatment. A numeric vector.  
**AST.M** Aspartate aminotransferase after treatment. A numeric vector.  
**TBL.M** 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.

---

logLik.evmOpt

*Log-likelihood for evmOpt objects*

---

### Description

Return the log-likelihood or penalized log-likelihood for evmOpt objects.

### Usage

```
## S3 method for class 'evmOpt'
logLik(object, penalized = FALSE, ...)
```

### Arguments

object	fit model object
penalized	whether to return the penalized log-likelihood
...	some methods need more arguments

### Value

an object of class logLik

### See Also

[logLik](#)

---

```
makeReferenceMarginalDistribution
```

*Provide full marginal reference distribution for for maringal transformation*

---

### Description

This gives the option of providing a set of reference marginal distributions to use for marginal transformation if the data's own marginal distribution is not appropriate (for instance if only data for which one variable is large is available, the marginal distributions of the other variables will not be represented by the available data). In such situations, the user can supply the full marginal information of the non-thresholded variables which are necessary to transform these variables correctly from the original margins to Gumbel/Laplace for estimation of dependence model parameters.

### Usage

```
makeReferenceMarginalDistribution(x, r, whichNoChange = NULL)
```

### Arguments

x	output from migpd fit to the original data which does not represent at least one marginal distribution
r	output from migpd fit to the reference data which does represent the correct marginal distribution of the variable with incomplete representation in x
whichNoChange	Margins which are not to use the supplied reference distribution r have numeric indices, giving column numbers in original dataframe, listed in whichNoChange.

### Value

An object of class "migpd".

---

MCS

*Multivariate conditional Spearman's rho*

---

### Description

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

**Usage**

```

MCS(X, p = seq(0.1, 0.9, by = 0.1))

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

## S3 method for class 'MCS'
ggplot(data, mapping, main = "", ..., environment)

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

## S3 method for class 'bootMCS'
ggplot(data, mapping, main = "", alpha = 0.05, ylim,
..., environment)

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

## S3 method for class 'bootMCS'
summary(object, alpha = 0.05, ...)

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

```

**Arguments**

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

**Details**

The method is described in detail by Schmid and Schmidt (2007). The main code was written by Yiannis Papastathopoulos, 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 print 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**

F. Schmid and R. Schmidt, Multivariate conditional versions of Spearman's rho and related measures of tail dependence, *Journal of Multivariate Analysis*, 98, 1123 – 1140, 2007

**See Also**

[chi](#)

**Examples**

```
D <- liver[liver$dose == "D",]
plot(D)

Dmcs <- bootMCS(D[, 5:6])
Dmcs
plot(Dmcs)
```



---

mexDependence	<i>Estimate the dependence parameters in a conditional multivariate extreme values model</i>
---------------	--

---

### 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", referenceMargin = NULL, 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 <a href="#">migpd</a> .
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 <a href="#">mex</a> .
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 <i>a</i> and <i>b</i> 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 <code>constrain=TRUE</code> , if supplied, <code>start</code> must lie within the permitted area of the parameter space.
<code>marTransform</code>	Optional form of transformation to be used for probability integral transform of data from original to Gumbel or Laplace margins. Takes values <code>marTransform="mixture"</code> (the default) or <code>marTransform="empirical"</code> . When <code>marTransform="mixture"</code> , the rank transform is used below the corresponding GPD fitting threshold used in <code>x</code> , and the fitted gpd tail model is used above this threshold. When <code>marTransform="empirical"</code> the rank transform is used for the entire range of each marginal distribution.
<code>referenceMargin</code>	Optional set of reference marginal distributions to use for marginal transformation if the data's own marginal distribution is not appropriate (for instance if only data for which one variable is large is available, the marginal distributions of the other variables will not be represented by the available data). This object can be created from a combination of datasets and fitted GPDs using the function <code>makeReferenceMarginalDistribution</code> .
<code>nOptim</code>	Number of times to run optimiser when estimating dependence model parameters. Defaults to 1. In the case of <code>nOptim &gt; 1</code> the first call to the optimiser uses the value <code>start</code> as a starting point, while subsequent calls to the optimiser are started at the parameter value to which the previous call converged.
<code>PlotLikDo</code>	Logical value: whether or not to plot the profile likelihood surface for dependence model parameters under constrained estimation.
<code>PlotLikRange</code>	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 <i>a</i> and <i>b</i> respectively. If this argument is not missing, then <code>PlotLikDo</code> is set equal to <code>TRUE</code> .
<code>PlotLikTitle</code>	Used only if <code>PlotLikDo=TRUE</code> . Character string. Optional title added to the profile log-likelihood surface plot.
<code>...</code>	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:

<code>margins</code>	An object of class <code>migpd</code> .
<code>dependence</code>	An object of class <code>mexDependence</code> .
<code>call</code>	This matches the original function call.

### Author(s)

Harry Southworth, Janet E. Heffernan

### References

J. E. Heffernan and J. A. Tawn, A conditional approach for multivariate extreme values, *Journal of the Royal Statistical society B*, 66, 497 – 546, 2004.

C. Keef, I. Papastathopoulos and J. A. Tawn. Estimation of the conditional distribution of a multivariate variable given that one of its components is large: Additional constraints for the Heffernan and Tawn model, *Journal of Multivariate Analysis*, 115, 396 – 404, 2013

### 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)))

# try different starting value:
mexDependence(NO.NO2, which = "NO",dqu=0.7,start=c(0.1,0.1),
              PlotLikDo=TRUE,PlotLikTitle=c("NO2 | NO"),
              PlotLikRange = list(a=c(0,0.8),b=c(-0.2,0.6)))
```

---

mexMonteCarlo

*Simulation from dependence models*


---

## Description

Simulate Monte Carlo sample from a collection of fitted conditional dependence models.

## Usage

```
mexMonteCarlo(nSample,mexList,mult=10)
```

## Arguments

nSample	Required sample size.
mexList	List of fitted dependence models (returned by <a href="#">mexAll</a> ).
mult	Integer specifying what multiple of the total number of points should be generated for rejection sample

## Details

Generates a Monte Carlo sample of the required size from a collection of conditional multivariate extreme values model of Heffernan and Tawn, 2004. For each marginal variable, the model that conditions on that margin is used to simulate values in the part of the sample space for which that margin is the largest of all marginal variables (measured on a quantile scale).

## Value

A list with the following components:

nR	For each margin, number of original Monte Carlo points replaced by points generated under the corresponding conditional model.
----	--

MCsample	Matrix containing the Monte Carlo sample, dimension nSample by dimension of original dataset.
whichMax	Vector of indices indicating which variable is largest (on the quantile scale)
whichMaxAboveThresh	Logical vector indicating which of the variables identified by whichMax are additionally above the corresponding threshold for dependence estimation.

**Author(s)**

Harry Southworth, Janet E. Heffernan

**References**

J. E. Heffernan and J. A. Tawn, A conditional approach for multivariate extreme values, Journal of the Royal Statistical society B, 66, 497 – 546, 2004

**Examples**

```
mAll <- mexAll(winter,mqu=0.7,dqu=c(0.7,0.7,0.7,0.7,0.7))
mexMC <- mexMonteCarlo(5000,mAll)
pairs(mexMC$MCsample)
```

---

mexRangeFit	<i>Estimate dependence parameters in a conditional multivariate extreme values model over a range of thresholds.</i>
-------------	--

---

**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(.01, .01), R = 10, nPass=3, trace=10, margins = "laplace", constrain
= TRUE, v = 10, referenceMargin=NULL)
```

**Arguments**

x	An object of class <a href="#">mex</a> or <a href="#">migpd</a> .
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 <a href="#">mexDependence</a> .

R	The number of bootstrap runs to perform at each threshold. Defaults to R=10.
nPass	Argument passed to function <a href="#">bootmex</a> .
trace	Argument passed to function <a href="#">bootmex</a> .
margins	Argument passed to function <a href="#">mexDependence</a> .
constrain	Argument passed to function <a href="#">mexDependence</a> .
v	Argument passed to function <a href="#">mexDependence</a> .
referenceMargin	Optional set of reference marginal distributions to use for marginal transformation if the data's own marginal distribution is not appropriate (for instance if only data for which one variable is large is available, the marginal distributions of the other variables will not be represented by the available data). This object can be created from a combination of datasets and fitted GPDs using the function <a href="#">makeReferenceMarginalDistribution</a> .
...	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.

### Author(s)

Harry Southworth, Janet E. Heffernan

### References

J. E. Heffernan and J. A. Tawn, A conditional approach for multivariate extreme values, *Journal of the Royal Statistical society B*, 66, 497 – 546, 2004

### See Also

[mexDependence](#), [bootmex](#)

### Examples

```
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)
```

---

 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**

```
library(MASS)
liver <- liver
liver$ndose <- as.numeric(liver$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$ndose <- liver$ndose
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))$obj$link[1:2]
astco <- predict(gast, type="lp", newdata=data.frame(ndose=4))$obj$link[1:2]
alpc0 <- predict(galp, type="lp", newdata=data.frame(ndose=4))$obj$link[1:2]

Dgpd <- migpdCoefs(Dgpds, which=c("alt", "ast", "alp"),
                  coefs=list(altco, astco, alpc0))

summary(Dgpd)
summary(Dgpds)
```

---

mrl

*Mean residual life plot*


---

## 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 'summary.mrl'
print(x, ...)
## S3 method for class 'mrl'
summary(object, ...)
## S3 method for class 'mrl'
plot(x, xlab="Threshold", ylab="Mean excess", ...)
## S3 method for class 'mrl'
ggplot(data, mapping, xlab = "Threshold",
        ylab = "Mean excess", main=NULL, fill="orange", col="blue",
        rug=TRUE, addNexcesses=TRUE, textsize=4, ..., environment)
```

## 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.
nint	The number of points at which to compute the plot.
alpha	Used to determine coverage of confidence interval to plot. Defaults to plotting a 95% interval.



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.
col	Colour of the line on the MRL plot.
rug	Whether to add raw data as a rug along axis of plot.
fill	Colour of the pointwise confidence region on the MRL plot.
main	Main title.
addNexcesses	Whether to annotate the plot with the numbers of excesses over increasing thresholds. Defaults to addNexcesses=TRUE.
textsize	Size of text on the plot (ggplot). Defaults to textsize=4.
mapping, environment	Not used.

### 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

S. Coles, *An Introduction to Statistical Modeling of Extreme Values*, Springer, 2001

---

plot.copula	<i>Plot copulas</i>
-------------	---------------------

---

### Description

Plot copulas

### Usage

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

### Arguments

x	A copula object
jitter.	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.

---

plot.evmOpt	<i>Plots for evmOpt objects</i>
-------------	---------------------------------

---

### Description

Various plots for evmOpt objects. 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 plotr1.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.

### Usage

```
## S3 method for class 'evmOpt'
plot(x, main = rep(NULL, 4), xlab = rep(NULL, 4),
     nsim = 1000, alpha = 0.05, ...)
```

**Arguments**

x	an object of class evmOpt
main	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	The number of replicates to be simulated to produce the simulated tolerance intervals.
alpha	A $100(1 - \alpha)\%$ simulation envelope is produced.
...	FIXME

**Details**

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

**See Also**

[evm](#)

---

plot.evmSim                      *Plots for evmSim objects*

---

**Description**

This 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.

**Usage**

```
## S3 method for class 'evmSim'
plot(x, which.plots = 1:3, density.adjust = 2,
     print.seed = FALSE, ...)
```

**Arguments**

x	an object of class evmSim
which.plots	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.
density.adjust	In plot method for class evmSim. Passed into density. Controls the amount of smoothing of the kernel density estimate.
print.seed	Whether or not to print the seed used in the simulations, or to annotate the plots with it.
...	ignored

**See Also**

[evm](#)  
[density](#)

---

plot.lp.evmOpt	<i>Predict return levels from extreme value models, or obtain the linear predictors.</i>
----------------	--

---

**Description**

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

**Usage**

```
## S3 method for class 'lp.evmOpt'
plot(x, main = NULL, pch = 1, ptcol = 2,
     cex = 0.75, linecol = 4, cicol = 1, polycol = 15, plot. = TRUE,
     ...)

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

linearPredictors(object, newdata = NULL, 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 = 0.05, unique. = TRUE, all = FALSE, sumfun = NULL, ...)

## S3 method for class 'evmSim'
linearPredictors(object, newdata = NULL,
                 se.fit = FALSE, ci.fit = FALSE, alpha = 0.05, 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 = 0.05, unique. = TRUE, all = FALSE, sumfun = NULL, ...)
```

```
## S3 method for class 'evmBoot'
linearPredictors(object, newdata = NULL,
  se.fit = FALSE, ci.fit = FALSE, alpha = 0.05, unique. = TRUE,
  all = FALSE, sumfun = NULL, ...)

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

## Arguments

x	An object of class lp.evmOpt, lp.evmSim or lp.evmBoot, to be passed to methods for these classes.
main, pch, ptcol, cex, linecol, cicol, polycol, plot, plot.	Further arguments to plot methods.
...	Further arguments to methods.
object	An object of class evmOpt, evmSim or evmBoot.
M	The return period: 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.
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.
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 newdata. Defaults to unique. = TRUE.
full.cov	Should the full covariance matrix be returned as part of a list object. This is used internally and not intended for direct use. Defaults to full.cov = FALSE

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.
digits	Number of digits to show when printing objects.

### Details

By default, return levels predicted from the unique values of the linear predictors are returned. For the evmBoot method, estimates of confidence intervals are simply quantiles of the bootstrap sample. The evmBoot method is just a wrapper for the evmSim method.

### Value

A list with two entries: the first being the call and the second being a further list with one entry for each value of M.

### Note

At present, the confidence intervals returned for an object of class evmOpt are simple confidence intervals based on assumptions of normality that are likely to be far from the truth in many cases. A better approach would be to use profile likelihood, and we intend to implement this method at a future date. Alternatively, the credible intervals returned by using Bayesian estimation and the predict method for class "evmSim" will tend to give a better representation of the asymmetry of the estimated intervals around the parameter point estimates.

### Author(s)

Harry Southworth and Janet E. Heffernan

---

plot.rl.evmOpt

*Return levels*

---

### Description

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

### Usage

```
## S3 method for class 'rl.evmOpt'
plot(x, xlab, ylab, main, pch = 1, pcol = 2,
     cex = 0.75, linecol = 4, cicol = 0, polycol = 15,
     smooth = FALSE, sameAxes = TRUE, type = "median", ylim = NULL,
     plot. = TRUE, ...)
```

```

## S3 method for class 'rl.evmSim'
plot(x, xlab, ylab, main, pch = 1, ptcol = 2,
     cex = 0.75, linecol = 4, cicol = 0, polycol = 15,
     smooth = FALSE, sameAxes = TRUE, type = "median", ylim = NULL,
     plot. = TRUE, ...)

## S3 method for class 'rl.evmBoot'
plot(x, xlab, ylab, main, pch = 1, ptcol = 2,
     cex = 0.75, linecol = 4, cicol = 0, polycol = 15,
     smooth = FALSE, sameAxes = TRUE, type = "median", ylim = NULL,
     plot. = TRUE, ...)

rl(object, M = 1000, newdata = NULL, se.fit = FALSE,
    ci.fit = FALSE, alpha = 0.05, unique. = TRUE, ...)

## S3 method for class 'evmOpt'
rl(object, M = 1000, newdata = NULL, se.fit = FALSE,
    ci.fit = FALSE, alpha = 0.05, unique. = TRUE, ...)

## S3 method for class 'evmSim'
rl(object, M = 1000, newdata = NULL, se.fit = FALSE,
    ci.fit = FALSE, alpha = 0.05, unique. = TRUE, all = FALSE,
    sumfun = NULL, ...)

## S3 method for class 'evmBoot'
rl(object, M = 1000, newdata = NULL,
    se.fit = FALSE, ci.fit = FALSE, alpha = 0.05, unique. = TRUE,
    all = FALSE, sumfun = NULL, ...)

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

```

## Arguments

<code>x</code>	Object passed to plot and print methods.
<code>xlab, ylab, main, pch, ptcol, cex, linecol, cicol, polycol, smooth, sameAxes, ylim</code>	Further arguments to plot methods.
<code>type</code>	For calls to plot methods for objects of class <code>rl.evmSim</code> or <code>rl.evmBoot</code> , specifies whether to use the sample mean ( <code>type="mean"</code> ) or median ( <code>type="median"</code> ) estimate of the return levels.
<code>plot.</code>	Parameter for plot method, whether to produce plots.
<code>...</code>	Further arguments to be passed to methods.
<code>object</code>	An object of class <code>evmOpt</code> , <code>evmSim</code> or <code>evmBoot</code> .
<code>M</code>	The $M$ -observation return level is computed by the function. Defaults to $M = 1000$ .
<code>newdata</code>	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.

<code>se.fit</code>	Whether or not to return the standard error of the predicted value. Defaults to <code>se.fit = FALSE</code> .
<code>ci.fit</code>	Whether or not to return a confidence interval for the predicted value. Defaults to <code>ci.fit = FALSE</code> . For objects of class <code>evmOpt</code> , if set to <code>TRUE</code> then the confidence interval is a simple symmetric confidence interval based on the estimated approximate standard error. For the <code>evmSim</code> and <code>evmBoot</code> methods, the confidence interval represents quantiles of the simulated distribution of the parameters.
<code>alpha</code>	If <code>ci.fit = TRUE</code> , a $100(1 - \alpha)\%$ confidence interval is returned. Defaults to <code>alpha = 0.050</code> .
<code>unique.</code>	If <code>unique. = TRUE</code> , predictions for only the unique values of the linear predictors are returned, rather than for every row of the original dataframe or of <code>newdata</code> if this latter is specified. Defaults to <code>unique. = TRUE</code> .
<code>all</code>	For the <code>evmSim</code> and <code>evmBoot</code> methods, if <code>all = TRUE</code> , the predictions are returned for every simulated parameter vector. Otherwise, only a summary of the posterior/bootstrap distribution is returned. Defaults to <code>all = FALSE</code> .
<code>sumfun</code>	For the <code>evmSim</code> and <code>evmBoot</code> methods, a summary function can be passed in. If <code>sumfun = FALSE</code> , the default, the summary function used returns the estimated mean and median, and quantiles implied by <code>alpha</code> .
<code>digits</code>	Number of digits to show when printing output.

### 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 `r1` 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
r1(mod, M=100*365) # 100-year return level
```

---

```
print.evmOpt          Print evmOpt objects
```

---

### Description

Print `evmOpt` objects

### Usage

```
## S3 method for class 'evmOpt'
print(x, digits = max(3, getOption("digits") - 3), ...)
```



**Arguments**

x	a fit evmOpt object
digits	number of digits used for printing
...	further arguments passed to <code>format</code>

---

rain, wavesurge and portpirie

*Rain, wavesurge, portpirie and nidd datasets.*

---

**Description**

Rainfall, wave-surge, Port Pirie and River Nidd data sets.

**Format**

The format of the rain data is: num [1:17531] 0 2.3 1.3 6.9 4.6 0 1 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'.

The River Nidd data represents 154 measurements of the level of the River Nidd at Hunsingore Weir (Yorkshire, UK) between 1934 and 1969. Each measurement breaches the threshold of  $\$65 \text{ m}^3/2\$$ . Various authors have analysed this dataset, as described by Papastathopoulos and Tawn~*egp*, there being some apparent difficulty in identifying a threshold above which GPD models are suitable.

**Details**

The rain, wave-surge and Port Pirie datasets are used by Coles and appear in the `ismev` package. The River Nidd data appear in the `evir` package.

**Source**

Copied from the `ismev` package and the `evir` package

**References**

S. Coles, *An Introduction to Statistical Modeling of Extreme Values*, Springer, 2001

I. Papastathopoulos and J. A. Tawn, Extended Generalised Pareto Models for Tail Estimation, *Journal of Statistical Planning and Inference*, 143, 134 – 143, 2011

---

rFrechet	<i>Extreme Value random process generation.</i>
----------	---

---

**Description**

Extreme Value random process generation.

**Usage**

rFrechet(n)

**Arguments**

n                      Number of samples to generate.

**Details**

Generation of samples from unit Frechet processes.

**Examples**

rFrechet(1000)

---

rMaxAR	<i>Extreme Value random process generation.</i>
--------	---

---

**Description**

Extreme Value random process generation.

**Usage**

rMaxAR(n, theta)

**Arguments**

n                      Number of samples to generate.  
 theta                  Parameter of the MAX AR process, takes values between 0 and 1.

**Details**

Generation of samples from Max AR(theta) processes.

**Examples**

rMaxAR(1000, 0.2)

---

simulate.evmOpt	<i>Simulate from a fitted evm object</i>
-----------------	--

---

### Description

Simulate random numbers from a fitted evm object

### Usage

```
## S3 method for class 'evmOpt'
simulate(object, nsim = 1, seed = NULL,
         param = NULL, ...)

## S3 method for class 'evmSim'
simulate(object, nsim = 1, seed = NULL, ...)

## S3 method for class 'evmBoot'
simulate(object, nsim = 1, seed = NULL, ...)
```

### Arguments

object	A fitted evm object having class 'evmOpt', 'evmSim' or 'evmBoot'.
nsim	The number of simulations to perform. Defaults to nsim=1. A single simulation involves simulating a new set of responses from the data that was provided to evm (after thresholding if thresholding is performed.)
seed	An integer to be passed to set.seed. Defaults to seed=NULL.
param	Parameters to use in the random number generator. Defaults to param=NULL in which case the parameters from the fitted model are used. For simulate.evmSim and simulate.evmBoot, this argument is not available and the simulated parameters or replicates are used.
...	Unused.

### Details

For simulate.evmSim and simulate.evmBoot, the parameters from the Markov chains or bootstrap replicates are randomly permuted prior to each set of simulated responses being computed. In this way, reusing the same set of values is avoided.

### Value

If nsim=1, a vector or random numbers simulated from the fitted model object. If nsim > 1, a matrix with each column being a set of simulated responses.

### Author(s)

Paul Metcalfe, Harry Southworth

**See Also**[evm](#)**Examples**

```
mod <- evm(rain, qu=.95)
hist(simulate(mod, 100))
```

---

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.

**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<sup>3</sup>

**Source**

Provided as online supplementary material to Heffernan and Tawn, 2004:

<http://www.blackwellpublishing.com/rss/Readmefiles/heffernan.htm>

**References**

J. E. Heffernan and J. A. Tawn, A conditional approach for multivariate extreme values, *Journal of the Royal Statistical society B*, 66, 497 – 546, 2004

**Examples**

```
data(summer)
data(winter)
```

---

texmexFamily	<i>Create families of distributions</i>
--------------	---

---

**Description**

Create families of distributions for use with extreme value modelling.

**Usage**

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

**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.
sandwich	Function to compute the filling in the Huber sandwich estimator of the covariance matrix of parameter estimates, used for dependent data. Only implemented in family gpd.
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, gev, weibull, gumbel and egp3. 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, sandwich, start, resid are not provided, they default to NULL.

**Note**

The gpd, gev, weibull, gumbel and egp3 families are provided. The [evm](#) function defaults to using the gpd family.

**Author(s)**

Harry Southworth

**See Also**

[evm](#)

---

thinAndBurn

*Process Metropolis output from extreme value model fitting to discard unwanted observations.*

---

**Description**

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

**Usage**

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

**Arguments**

object	Object of class 'evmSim' as returned by <code>evm</code> called with <code>method="simulate"</code> .
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 <code>burn=0</code> .
thin	<code>thin</code> 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 <code>thin=1</code> .

**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**

```
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=11000)
mod
par(mfrow=c(3, 2))
plot(mod)
mod1 <- thinAndBurn(mod,burn=1000, thin=5)
plot(mod1)
```

# Index

- \*Topic **datasets**
  - liver, [52](#)
  - rain, wavesurge and portpirie, [73](#)
  - summer and winter data, [76](#)
- \*Topic **hplot**
  - ggplot.evmBoot, [36](#)
  - ggplot.evmOpt, [37](#)
  - ggplot.evmSim, [38](#)
- \*Topic **methods**
  - plot.lp.evmOpt, [68](#)
- \*Topic **models**
  - bootmex, [10](#)
  - degp3, [16](#)
  - dgev, [17](#)
  - dgpdp, [18](#)
  - egp3RangeFit, [20](#)
  - evm, [22](#)
  - ggplot.mex, [39](#)
  - ggplot.migpd, [44](#)
  - gpdRangeFit, [48](#)
  - mexDependence, [57](#)
  - mexMonteCarlo, [60](#)
  - mexRangeFit, [61](#)
  - mrl, [64](#)
  - simulate.evmOpt, [75](#)
  - texmex-package, [4](#)
  - texmexFamily, [77](#)
- \*Topic **multivariate**
  - bootmex, [10](#)
  - chi, [12](#)
  - copula, [15](#)
  - ggplot.mex, [39](#)
  - ggplot.migpd, [44](#)
  - MCS, [54](#)
  - mexDependence, [57](#)
  - mexMonteCarlo, [60](#)
  - mexRangeFit, [61](#)
  - migpdCoefs, [63](#)
  - texmex-package, [4](#)
- \*Topic **package**
  - texmex-package, [4](#)
- \*Topic **univar**
  - edf, [19](#)
  - .exprel, [6](#)
  - .log1mexp, [7](#)
  - .log1prel, [7](#)
  - .specfun.safe.product, [8](#)
- addExcesses, [8](#)
- AIC, [9](#)
- AIC.evmOpt, [9](#)
- AIC.evmSim (AIC.evmOpt), [9](#)
- bootExtremalIndex (extremalIndex), [31](#)
- bootMCS (MCS), [54](#)
- bootmex, [4](#), [10](#), [11](#), [43](#), [46](#), [59](#), [62](#)
- bw.nrd, [41](#)
- calcJointExceedanceCurve (JointExceedanceCurve), [50](#)
- cgpd (texmexFamily), [77](#)
- chi, [12](#), [56](#)
- coef.evmBoot (evmBoot), [28](#)
- copula, [15](#), [19](#)
- declust, [4](#)
- declust (extremalIndex), [31](#)
- degp3, [16](#)
- density, [68](#)
- dgev, [17](#)
- dgpdp, [18](#)
- edf, [15](#), [16](#), [19](#)
- egp3 (texmexFamily), [77](#)
- egp3RangeFit, [20](#)
- endPoint, [22](#)
- evm, [4](#), [20–22](#), [22](#), [29](#), [31](#), [35](#), [41](#), [45](#), [49](#), [50](#), [67](#), [68](#), [76](#), [78](#), [79](#)
- evm.declustered, [27](#)
- evm.declustered (extremalIndex), [31](#)



- evmBoot, 28, 29
- evmSim, 29
- evmSimSetSeed, 31
- extremalIndex, 31
- extremalIndexRangeFit (extremalIndex), 31
- format, 73
- geom\_jointExcCurve (JointExceedanceCurve), 50
- gev (texmexFamily), 77
- ggacfplots (ggplot.evmSim), 38
- ggbootdensplots (ggplot.evmBoot), 36
- ggdensplots (ggplot.evmSim), 38
- ggplot.bootMCS (MCS), 54
- ggplot.chi (chi), 12
- ggplot.declustered, 36
- ggplot.egp3RangeFit (egp3RangeFit), 20
- ggplot.evmBoot, 36
- ggplot.evmOpt, 26, 27, 37
- ggplot.evmOpt, (ggplot.evmOpt), 37
- ggplot.evmSim, 38
- ggplot.extremalIndex (ggplot.declustered), 36
- ggplot.extremalIndexRangeFit (extremalIndex), 31
- ggplot.gpdRangeFit (gpdRangeFit), 48
- ggplot.hist.evmOpt (ggplot.evmOpt), 37
- ggplot.MCS (MCS), 54
- ggplot.mex, 39
- ggplot.migpd, 44
- ggplot.mrl (mrl), 64
- ggplot.ppevm (ggplot.evmOpt), 37
- ggplot.predict.mex (ggplot.mex), 39
- ggplot.qqevm (ggplot.evmOpt), 37
- ggplot.rl.evmBoot (ggplot.rl.evmOpt), 47
- ggplot.rl.evmOpt, 47
- ggplot.rl.evmSim (ggplot.rl.evmOpt), 47
- ggplotr1 (ggplot.evmOpt), 37
- ggtraceplots (ggplot.evmSim), 38
- gpd (texmexFamily), 77
- gpd.prof, 47
- gpdRangeFit, 21, 46, 48
- gumbel (texmexFamily), 77
- JointExceedanceCurve, 50
- linearPredictors (plot.lp.evmOpt), 68
- liver, 52
- logLik, 53
- logLik.evmOpt, 53
- makeReferenceMarginalDistribution, 54
- MCS, 14, 54
- mex, 4, 10, 46, 57, 61
- mex (ggplot.mex), 39
- mexAll, 60
- mexAll (ggplot.mex), 39
- mexDependence, 11, 41–43, 46, 57, 59, 61, 62
- mexMonteCarlo, 43, 60
- mexRangeFit, 61
- migpd, 11, 42, 43, 57, 59, 61, 63
- migpd (ggplot.migpd), 44
- migpdCoefs, 63
- mrl, 21, 46, 64
- nidd (rain, wavesurge and portpirie), 73
- optim, 41, 45
- pegp3 (degp3), 16
- pgev (dgev), 17
- pgpd (dgpdp), 18
- plot.bootMCS (MCS), 54
- plot.bootmex (bootmex), 10
- plot.chi (chi), 12
- plot.copula, 16, 66
- plot.declustered (extremalIndex), 31
- plot.egp3RangeFit (egp3RangeFit), 20
- plot.evmBoot (evmBoot), 28
- plot.evmOpt, 26, 27, 37, 38, 66
- plot.evmSim, 67
- plot.extremalIndexRangeFit (extremalIndex), 31
- plot.gpdRangeFit (gpdRangeFit), 48
- plot.lp.evmOpt, 68
- plot.MCS (MCS), 54
- plot.mex, 59
- plot.mex (ggplot.mex), 39
- plot.migpd (ggplot.migpd), 44
- plot.mrl (mrl), 64
- plot.predict.mex (ggplot.mex), 39
- plot.rl.evmBoot (plot.rl.evmOpt), 70
- plot.rl.evmOpt, 70
- plot.rl.evmSim (plot.rl.evmOpt), 70
- portpirie (rain, wavesurge and portpirie), 73

predict.evmBoot (plot.lp.evmOpt), 68  
 predict.evmOpt, 27  
 predict.evmOpt (plot.lp.evmOpt), 68  
 predict.evmSim (plot.lp.evmOpt), 68  
 predict.mex, 11, 46, 59  
 predict.mex (ggplot.mex), 39  
 print.bootMCS (MCS), 54  
 print.bootmex (bootmex), 10  
 print.chi (chi), 12  
 print.declustered (extremalIndex), 31  
 print.egp3RangeFit (egp3RangeFit), 20  
 print.evmBoot (evmBoot), 28  
 print.evmOpt, 72  
 print.extremalIndex (extremalIndex), 31  
 print.gpdRangeFit (gpdRangeFit), 48  
 print.jointExcCurve  
     (JointExceedanceCurve), 50  
 print.lp.evmOpt (plot.lp.evmOpt), 68  
 print.MCS (MCS), 54  
 print.mex (ggplot.mex), 39  
 print.mexList (ggplot.mex), 39  
 print.mrl (mrl), 64  
 print.rl.evmOpt (plot.rl.evmOpt), 70  
 print.summary.bootMCS (MCS), 54  
 print.summary.chi (chi), 12  
 print.summary.evmBoot (evmBoot), 28  
 print.summary.gpdRangeFit  
     (gpdRangeFit), 48  
 print.summary.mex (ggplot.mex), 39  
 print.summary.mrl (mrl), 64  
 print.summary.texmexFamily  
     (texmexFamily), 77  
 print.texmexFamily (texmexFamily), 77

qgev (dgev), 17  
 qgpd (dgpd), 18

rain (rain, wavesurge and portpirie), 73  
 rain, wavesurge and portpirie, 73  
 rank, 14, 19  
 regp3 (degp3), 16  
 rFrechet, 74  
 rgev (dgev), 17  
 rgpd (dgpd), 18  
 rl (plot.rl.evmOpt), 70  
 rl.evmOpt, 27  
 rMaxAR, 74

signif, 22

simulate.evmBoot (simulate.evmOpt), 75  
 simulate.evmOpt, 75  
 simulate.evmSim (simulate.evmOpt), 75  
 summary.bootMCS (MCS), 54  
 summary.chi (chi), 12  
 summary.evmBoot (evmBoot), 28  
 summary.gpdRangeFit (gpdRangeFit), 48  
 summary.mex (ggplot.mex), 39  
 summary.mrl (mrl), 64  
 summary.predict.mex (ggplot.mex), 39  
 summary.texmexFamily (texmexFamily), 77  
 summer (summer and winter data), 76  
 summer and winter data, 76

texmex (texmex-package), 4  
 texmex-package, 4  
 texmexFamily, 26, 77  
 thinAndBurn, 78

wavesurge (rain, wavesurge and  
     portpirie), 73  
 weibull (texmexFamily), 77  
 winter (summer and winter data), 76