Package ‘tsxtreme’

October 14, 2022

Type Package

Title Bayesian Modelling of Extremal Dependence in Time Series

Version 0.3.3

Date 2021-04-23

Author Thomas Lugrin

Maintainer Thomas Lugrin <thomas.lugrin@alumni.epfl.ch>

Imports evd, mvtnorm, stats, MASS, graphics, tictoc

Description Characterisation of the extremal dependence structure of time series, avoiding pre-processing and filtering as done typically with peaks-over-threshold methods. It uses the conditional approach of Heffernan and Tawn (2004) <DOI:10.1111/j.1467-9868.2004.02050.x> which is very flexible in terms of extremal and asymptotic dependence structures, and Bayesian methods improve efficiency and allow for deriving measures of uncertainty. For example, the extremal index, related to the size of clusters in time, can be estimated and samples from its posterior distribution obtained.

License GPL (>= 2)

NeedsCompilation yes

Repository CRAN

Date/Publication 2021-04-23 21:20:03 UTC

R topics documented:

---

tsxtreme-package *Bayesian Modelling of Extremal Dependence in Time Series*

Description

Characterisation of the extremal dependence structure of time series, avoiding pre-processing and filtering as done typically with peaks-over-threshold methods. It uses the conditional approach of Heffernan and Tawn (2004) <DOI:10.1111/j.1467-9868.2004.02050.x> which is very flexible in terms of extremal and asymptotic dependence structures, and Bayesian methods improve efficiency and allow for deriving measures of uncertainty. For example, the extremal index, related to the size of clusters in time, can be estimated and samples from its posterior distribution obtained.

Details

Index of help topics:

```
bayesfit                Traces from MCMC output
bayesparams             Parameters for the semi-parametric approach
dep2fit                 Dependence model fit (stepwise)
depfit                  Dependence model fit
depmeasure              Dependence measures estimates
depmeasures             Estimate dependence measures
dlapl                   The Laplace Distribution
stepfit                 Estimates from stepwise fit
theta2fit               Fit time series extremes
thetaruns               Runs estimator
tsxtreme-package        Bayesian Modelling of Extremal Dependence in
                        Time Series
```

The Heffernan–Tawn conditional formulation for a stationary time series (X_t) with Laplace marginal distribution states that for a large enough threshold u there exist scale parameters $-1 \leq \alpha_j \leq 1$ and $0 \leq \beta_j \leq 1$ such that

$$Pr\left(\frac{X_j - \alpha_j X_0}{(X_j)^{\beta_j}} < z_j, j = 1, \ldots, m \mid X_0 > u\right) = H(z_1, \ldots, z_m),$$

with H non-degenerate; the equality holds exactly only when u tends to infinity.

There are mainly 3 functions provided by this package, which allow estimation of extremal dependence measures and fitting the Heffernan–Tawn model using Dirichlet processes.

`depfit` fits the Heffernan–Tawn model using a Bayesian semi-parametric approach.

`thetafit` computes posterior samples of the threshold-based index of Ledford and Tawn (2003) based on inference in `depfit`.

`chifit` computes posterior samples of the extremal measure of dependence of Coles, Heffernan and Tawn (1999) at any extremal level.

Some corresponding functions using the stepwise approach of Heffernan and Tawn (2004) are also part of the package, namely `dep2fit` and `theta2fit`.

The empirical estimation of the extremal index can be done using `thetaruns` and some basic functions handling the Laplace distribution are also available in `dlapl`.

Author(s)

Thomas Lugrin

Maintainer: Thomas Lugrin <thomas.lugrin@alumni.epfl.ch>

References

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

Davison, A. C. and Smith, R. L. (1990) Models for exceedances over high thresholds. *Journal of the Royal Statistical Society Series B*, **52**, 393–442.

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

Ledford, W. A. and Tawn, J. A. (2003) Diagnostics for dependence within time series extremes. *Journal of the Royal Statistical Society Series B*, **65**, 521–543.

Lugrin, T., Davison, A. C. and Tawn, J. A. (2016) Bayesian uncertainty management in temporal dependence of extremes. *Extremes*, **19**, 491–515.

See Also

`thetafit`, `chifit`, `depfit`

`bayesfit` *Traces from MCMC output*

Description

Test or show objects of class "bayesfit".

Usage

```
is.bayesfit(x)
```

Arguments

`x`	an arbitrary R object.

Details

Default plot shows samples of residual densities (`which==1`), residual distribution with credible interval (5% and 95% posterior quantiles; `which==2`), and joint posterior distribution of α and β (`which==3`) for each lag successively. `which` can be any composition of 1,2 and 3.

Value

An object of class "bayesfit" is a list containing MCMC traces for:

`a,b`	Heffernan-Tawn parameters.
`sd,mean,w`	standard deviations, means and weights of the mixture components.
`prec`	precision parameter of the Dirichlet process.
`ci`	auxiliary variable; components' indices for each observation.
`noo`	number of observations in each mixture component.
`noc`	number of non-empty components in the mixture.
`prop.sd`	standard deviations of the proposal distributions for `a` and `b`.

And `len`, the length of the traces, i.e., the number of iterations saved.

See Also

`bayesparams`, `stepfit`

`bayesparams` *Parameters for the semi-parametric approach*

Description

Create, test or show objects of class "bayesparams".

Usage

```
bayesparams(prop.a = 0.02, prop.b = 0.02,
  prior.mu = c(0, 10), prior.nu = c(2, 1/2), prior.eta = c(2, 2),
  trunc = 100, comp.saved = 15, maxit = 30000,
  burn = 5000, thin = 1,
  adapt = 5000, batch.size = 125,
  mode = 1)

is.bayesparams(x)
```

Arguments

`prop.a, prop.b`	standard deviation for the Gaussian proposal of the Heffernan–Tawn parameters.
`prior.mu`	mean and standard deviation of the Gaussian prior for the components' means.
`prior.nu`	shape and rate of the inverse gamma prior for the components' variances.
`prior.eta`	shape and scale of the gamma prior for the precision parameter of the Dirichlet process.
`trunc`	integer; value of the truncation for the approximation of the infinite sum in the stick-breaking representation.

`comp.saved`	number of first components to be saved and returned.
`maxit`	maximum number of iterations.
`burn`	number of first iterations to discard.
`thin`	positive integer; spacing between iterations to be saved. Default is 1, i.e., all iterations are saved.
`adapt`	integer; number of iterations during which an adaption algorithm is applied to the proposal variances of α and β.
`batch.size`	size of batches used in the adaption algorithm. It has no effect if `adapt==0`.
`mode`	verbosity; 0 for debug mode, 1 (default) for standard output, and 2 for silent.
`x`	an arbitrary R object.

Details

`prop.a` is a vector of length 5 with the standard deviations for each region of the RAMA for the (Gaussian) proposal for α. If a scalar is given, 5 identical values are assumed.

`prop.b` is a vector of length 3 with the standard deviations for each region of the RAMA for the (Gaussian) proposal for β. If a scalar is provided, 3 identical values are assumed.

`comp.saved` has no impact on the calculations: its only purpose is to prevent from storing huge amounts of empty components.

The regional adaption scheme targets a 0.44 acceptance probability. It splits $[-1; 1]$ in 5 regions for α and $[0; 1]$ in 3 regions for β. The decision to increase/decrease the proposal standard deviation is based on the past `batch.size` MCMC iterations, so too low values yield inefficient adaption, while too large values yield slow adaption.

Default values for the hyperparameters are chosen to get reasonably uninformative priors.

See Also

`bayesfit`, `depmeasure`

Examples

```
is.bayesparams(bayesparams()) # TRUE
## use defaults, change max number of iteration of MCMC
par <- bayesparams(maxit=1e5)
```

`dep2fit` *Dependence model fit (stepwise)*

Description

The conditional Heffernan–Tawn model is used to fit the dependence in time of a stationary series. A standard 2-stage procedure is used.

Usage

```
dep2fit(ts, u.mar = 0, u.dep,
        lapl = FALSE, method.mar = c("mle","mom","pwm"),
        nlag = 1, conditions = TRUE)
```

Arguments

`ts`	numeric vector; time series to be fitted.
`u.mar`	marginal threshold; used when transforming the time series to Laplace scale.
`u.dep`	dependence threshold; level above which the dependence is modelled. `u.dep` can be lower than `u.mar`.
`lapl`	logical; is `ts` on the Laplace scale already? The default (FALSE) assumes unknown marginal distribution.
`method.mar`	a character string defining the method used to estimate the marginal GPD; either `"mle"` for maximum likelihood of `"mom"` for method of moments. Defaults to `"mle"`.
`nlag`	integer; number of lags to be considered when modelling the dependence in time.
`conditions`	logical; should conditions on α and β be set? (see Details) Defaults to `TRUE`.

Details

Consider a stationary time series (X_t) with Laplace marginal distribution; the fitting procedure consists of fitting

$$X_t = \alpha_t \times x_0 + x_0^{\beta_t} \times Z_t, \quad t = 1, \ldots, m,$$

with m the number of lags considered. A likelihood is maximised assuming $Z_t \sim N(\mu_t, \sigma_t^2)$, then an empirical distribution for the Z_t is derived using the estimates of α_t and β_t and the relation

$$\hat{Z}_t = \frac{X_t - \hat{\alpha}_t \times x_0}{x_0^{\hat{\beta}_t}}.$$

`conditions` implements additional conditions suggested by Keef, Papastathopoulos and Tawn (2013) on the ordering of conditional quantiles. These conditions help with getting a consistent fit by shrinking the domain in which (α, β) live.

Value

`alpha`	parameter controlling the conditional extremal expectation.
`beta`	parameter controlling the conditional extremal expectation and variance.
`res`	empirical residual of the model.
`pars.se`	vector of length 2 giving the estimated standard errors for `alpha` and `beta` given by the hessian matrix of the likelihood function used in the first step of the inference procedure.

See Also

`depfit`, `theta2fit`

Examples

```
## generate data from an AR(1)
## with Gaussian marginal distribution
n   <- 10000
dep <- 0.5
ar    <- numeric(n)
ar[1] <- rnorm(1)
for(i in 2:n)
  ar[i] <- rnorm(1, mean=dep*ar[i-1], sd=1-dep^2)
plot(ar, type="l")
plot(density(ar))
grid <- seq(-3,3,0.01)
lines(grid, dnorm(grid), col="blue")

## rescale margin
ar <- qlapl(pnorm(ar))

## fit model without constraints...
fit1 <- dep2fit(ts=ar, u.mar = 0.95, u.dep=0.98, conditions=FALSE)
fit1$a; fit1$b

## ...and compare with a fit with constraints
fit2 <- dep2fit(ts=ar, u.mar = 0.95, u.dep=0.98, conditions=TRUE)
fit2$a; fit2$b# should be similar, as true parameters lie well within the constraints
```

`depfit` *Dependence model fit*

Description

Bayesian semiparametrics are used to fit the Heffernan–Tawn model to time series. Options are available to impose a structure in time on the model.

Usage

```
depfit(ts, u.mar = 0, u.dep=u.mar,
    lapl = FALSE, method.mar=c("mle","mom","pwm"),  nlag = 1,
    par = bayesparams(),
    submodel = c("fom", "none", "ugm"))
```

Arguments

`ts`	numeric vector; time series to be fitted.
`u.mar`	marginal threshold; used when transforming the time series to Laplace scale.
`u.dep`	dependence threshold; level above which the dependence is modelled. `u.dep` can be lower than `u.mar`.
`lapl`	logical; is `ts` on the Laplace scale already? The default (FALSE) assumes unknown marginal distribution.

`method.mar`	a character string defining the method used to estimate the marginal GPD; either `"mle"` for maximum likelihood or `"mom"` for method of moments or `"pwm"` for probability weighted moments. Defaults to `"mle"`.
`nlag`	integer; number of lags to be considered when modelling the dependence in time.
`par`	an object of class 'bayesparams'.
`submodel`	a character string; "fom" for *first order Markov*, "none" for *no particular time structure*, or "ugm" for *univariate Gaussian mixture* (see details).

Details

`submodel` can be `"fom"` to impose a first order Markov structure on the model parameters α_j and β_j (see `thetafit` for more details); it can take the value `"none"` to impose no particular structure in time; it can also be `"ugm"` which can be applied to density estimation, as it corresponds to setting $\alpha = \beta = 0$ (see examples).

Value

An object of class 'bayesfit' with elements:

`a`	posterior trace of α.
`b`	posterior trace of β.
`sd`	posterior trace of the components' standard deviations.
`mean`	posterior trace of the components' means.
`w`	posterior trace of the components' assigned weights.
`prec`	posterior trace of the precision parameter.
`noo`	posterior trace of the number of observations per component.
`noc`	posterior trace of the number of components containing at least one observation.
`prop.sd`	trace of proposal standard deviations in the 5+3 regions of the adaption scheme for α and β.
`len`	length of the returned traces.

See Also

`thetafit`, `chifit`

Examples

```
## generate data from an AR(1)
## with Gaussian marginal distribution
n   <- 10000
dep <- 0.5
ar    <- numeric(n)
ar[1] <- rnorm(1)
for(i in 2:n)
  ar[i] <- rnorm(1, mean=dep*ar[i-1], sd=1-dep^2)
```

```
## rescale the margin
ar <- qlapl(pnorm(ar))

## fit the data
params <- bayesparams()
params$maxit <- 100# bigger numbers would be
params$burn  <- 10 # more sensible...
params$thin  <- 4
fit <- depfit(ts=ar, u.mar=0.95, u.dep=0.98, par=params)

########
## density estimation with submodel=="ugm"
data <- MASS::galaxies/1e3
dens <- depfit(ts=data, par=params, submodel="ugm")
```

`depmeasure` *Dependence measures estimates*

Description

Test or show objects of class "depmeasure".

Usage

```
is.depmeasure(x)
```

Arguments

`x`	an arbitrary R object.

Value

An object of class 'depmeasure' is a list which contains:

`fit`	an object of class 'bayesfit'
`distr`	an array with the samples used for the estimation.
`probs,levels`	points —probability and original scale respectively— at which the dependence measure is estimated

Depending on the dependence measure, `theta` or `chi`, a matrix with levels on row-entries and mean, median and specified quantiles of the posterior distribution of theta or chi respectively.

See Also

`depmeasures`

`depmeasures` *Estimate dependence measures*

Description

Appropriate marginal transforms are done before the fit using standard procedures, before the dependence model is fitted to the data. Then the posterior distribution of a measure of dependence is derived. `thetafit` gives posterior samples for the extremal index $\theta(x, m)$ and `chifit` does the same for the coefficient of extremal dependence $\chi_m(x)$.

Usage

```
thetafit(ts, lapl = FALSE, nlag = 1,
      R = 1000, S = 500,
      u.mar = 0, u.dep,
      probs = seq(u.dep, 0.9999, length.out = 30),
      method.mar = c("mle", "mom","pwm"), method = c("prop", "MCi"),
      silent = FALSE,
      fit = TRUE, prev.fit=bayesfit(), par = bayesparams(),
      submodel = c("fom", "none"), levels=c(.025,.975))

chifit(ts, lapl = FALSE, nlag = 1,
      R = 1000, S = 500,
      u.mar = 0, u.dep,
      probs = seq(u.dep, 0.9999, length.out = 30),
      method.mar = c("mle", "mom","pwm"), method = c("prop", "MCi"),
      silent = FALSE,
      fit = TRUE, prev.fit=bayesfit(), par = bayesparams(),
      submodel = c("fom", "none"), levels=c(.025,.975))
```

Arguments

`ts`	a vector, the time series for which to estimate the extremal index $\theta(x, m)$ or the coefficient of extremal dependence $\chi_m(x)$, with x a probability level and m a run-length (see details).
`lapl`	logical; `TRUE` indicates that `ts` has a marginal Laplace distribution. If `FALSE` (default), `method.mar` is used to transform the marginal distribution of `ts` to Laplace.
`nlag`	the run-length; an integer larger or equal to 1.
`R`	the number of samples per MCMC iteration drawn from the sampled posterior distributions; used for the estimation of the dependence measure.
`S`	the number of posterior distributions sampled to be used for the estimation of the dependence measure.
`u.mar`	probability; threshold used for marginal transformation if `lapl` is `FALSE`. Not used otherwise.

`u.dep`	probability; threshold used for the extremal dependence model.
`probs`	vector of probabilities; the values of x for which to evaluate $\theta(x, m)$ or $\chi_m(x)$.
`method.mar`	a character string defining the method used to estimate the marginal GPD; either `"mle"` for maximum likelihood of `"mom"` for method of moments or `"pwm"` for probability weighted moments methods. Defaults to `"mle"`.
`method`	a character string defining the method used to estimate the dependence measure; either `"prop"` for proportions or `"MCi"` for Monte Carlo integration (see details).
`silent`	logical (`FALSE`); verbosity.
`fit`	logical; `TRUE` means that the dependence model must be fitted and the values in `par` are used. Otherwise the result from a previous call to `depfit`.
`prev.fit`	an object of class 'bayesfit'. Needed if `fit` is FALSE. Typically returned by a previous call to `depfit`.
`par`	an object of class '`bayesparams`' to be used for the fit of dependence model.
`submodel`	a character string, either `"fom"` for *first order Markov* or `"none"` for no specification.
`levels`	vector of probabilites; the quantiles of the posterior distribution of the extremal measure to be computed.

Details

The sub-asymptotic extremal index is defined as

$$\theta(x, m) = Pr(X_1 < x, \ldots, X_m < x | X_0 > x),$$

whose limit as x and m go to ∞appropriately is the extremal index θ. The extremal index can be interpreted as the inverse of the asymptotic mean cluster size (see `thetaruns`).

The sub-asymptotic coefficient of extremal dependence is

$$\chi_m(x) = Pr(X_m > x | X_0 > x),$$

whose limit χ defines asymptotic dependence ($\chi > 0$) or asymptotic independence ($\chi = 0$).

Both types of extremal dependence measures can be estimated either using a

* proportion method (`method == "prop"`), sampling from the conditional probability given $X_0 > x$ and counting the proportion of sampled points falling in the region of interest, or

* Monte Carlo integration (`method == "MCi"`), sampling replicates from the marginal exponential tail distribution and evaluating the conditional tail distribution in these replicates, then taking their mean as an approximation of the integral.

`submodel == "fom"` imposes a first order Markov structure to the model, namely a geometrical decrease in α and a constant β across lags, i.e. $\alpha_j = \alpha^j$ and $\beta_j = \beta$, $j = 1, \ldots, m$.

Value

An object of class 'depmeasure', containing a subset of:

`bayesfit`	An object of class 'bayesfit'

theta	An array with dimensions m $\times$ length(probs) $\times$ (2+length(levels)), with the last dimension listing the posterior mean and median, and the level posterior quantiles
distr	An array with dimensions m $\times$ length(probs) $\times$ S; posterior samples of theta
chi	An array with dimensions m $\times$ length(probs) $\times$ (2+length(levels)), with the last dimension listing the posterior mean and median, and the level posterior quantiles
probs	probs
levels	probs transformed to original scale of ts

See Also

depfit, theta2fit, thetaruns

Examples

```
## generate data from an AR(1)
## with Gaussian marginal distribution
n   <- 10000
dep <- 0.5
ar    <- numeric(n)
ar[1] <- rnorm(1)
for(i in 2:n)
  ar[i] <- rnorm(1, mean=dep*ar[i-1], sd=1-dep^2)
plot(ar, type="l")
plot(density(ar))
grid <- seq(-3,3,0.01)
lines(grid, dnorm(grid), col="blue")

## rescale the margin (focus on dependence)
ar <- qlapl(pnorm(ar))

## fit the data
params <- bayesparams()
params$maxit <- 100 # bigger numbers would be
params$burn  <- 10  # more sensible...
params$thin  <- 4
theta <- thetafit(ts=ar, R=500, S=100, u.mar=0.95, u.dep=0.98,
                  probs = c(0.98, 0.999), par=params)
## or, same thing in two steps to control fit output before computing theta:
fit <- depfit(ts=ar, u.mar=0.95, u.dep=0.98, par=params)
plot(fit)
theta <- thetafit(ts=ar, R=500, S=100, u.mar=0.95, u.dep=0.98,
                  probs = c(0.98, 0.999), fit=FALSE, prev.fit=fit)
```

Laplace *The Laplace Distribution*

Description

Density, distribution function, quantile function and random generation for the Laplace distribution with location parameter `loc` and scale parameter `scale`.

Usage

```
dlapl(x, loc = 0, scale = 1, log = FALSE)
plapl(q, loc = 0, scale = 1, lower.tail = TRUE, log.p = FALSE)
qlapl(p, loc = 0, scale = 1, lower.tail = TRUE, log.p = FALSE)
rlapl(n, loc = 0, scale = 1)
```

Arguments

`x,q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of samples to generate.
`loc`	vector of location parameters.
`scale`	vector of scale parameters. These must be positive.
`lower.tail`	logical; if TRUE (default), probabilities are $Pr(X \leq x)$, otherwise $Pr(X > x)$.
`log,log.p`	logical; if TRUE, probabilities p are given as $\log(p)$.

Details

If `loc` or `scale` are not specified, they assume the default values of 0 and 1 respectively.

The Laplace distribution has density

$$f(x) = \exp(-|x - \mu|/\sigma)/(2\sigma)$$

where μ is the location parameter and σ is the scale parameter.

Value

`dlapl` gives the density, `plapl` gives the distribution function, `qlapl` gives the quantile function, and `rlapl` generates random deviates.

The length of the result is determined by `n` in `rlapl`, and is the maximum of the lengths of the numerical arguments for the other functions. Standard `R` vector operations are to be assumed.

If `sd`=0, the limit as `sd` decreases to 0 is returned, i.e., a point mass at `loc`. The case `sd`<0 is an error and nothing is returned.

Warning

Some checks are done previous to standard evaluation, but vector computations have not yet been tested thoroughly! Typically vectors not having lengths multiple of each other return an error.

See Also

dexp for the exponential distribution which is the positive part of the Laplace distribution.

Examples

```
## evaluate the density function on a grid of values
x  <- seq(from=-5, to=5, by=0.1)
fx <- dlapl(x, loc=1, scale=.5)

## generate random samples of a mixture of Laplace distributions
rnd <- rlapl(1000, loc=c(-5,-3,2), scale=0.5)

## an alternative:
rnd <- runif(1000)
rnd <- qlapl(rnd, loc=c(-5,-3,2), scale=0.5)

## integrate the Laplace density on [a,b]
a <- -1
b <- 7
integral <- plapl(b)-plapl(a)
```

---

stepfit *Estimates from stepwise fit*

Description

Create, test or show objects of class "stepfit".

Usage

```
stepfit()

is.stepfit(x)
```

Arguments

`x`	an arbitrary R object.

Value

An object of class "stepfit" is a list containing:

`a,b`	Heffernan–Tawn parameters.
`res`	fitted residuals.
`pars.se`	estimated standard error of a and b.

See Also

bayesfit, depmeasure

---

theta2fit *Fit time series extremes*

Description

Appropriate marginal transforms are done before the fit using standard procedures, before the dependence model is fitted to the data. Then the measure of dependence $\theta(x, m)$ is derived using a method described in Eastoe and Tawn (2012).

Usage

```
theta2fit(ts, lapl = FALSE, nlag = 1, R = 1000,
          u.mar = 0, u.dep, probs = seq(u.dep, 0.9999, length.out = 30),
          method.mar = c("mle","mom","pwm"), method = c("prop", "MCi"),
          silent = FALSE,
          R.boot = 0, block.length = m * 5, levels = c(.025,.975))
```

Arguments

`ts`	numeric vector; time series to be fitted.
`lapl`	logical; is `ts` on the Laplace scale already? The default (FALSE) assumes unknown marginal distribution.
`nlag`	integer; number of lags to be considered when modelling the dependence in time.
`R`	integer; the number of samples used for estimating $\theta(x, m)$.
`u.mar`	marginal threshold; used when transforming the time series to Laplace scale if `lapl` is FALSE; not used otherwise.
`u.dep`	dependence threshold; level above which the dependence is modelled. `u.dep` can be lower than `u.mar`.
`probs`	vector of probabilities; the values of x for which to evaluate $\theta(x, m)$.
`method.mar`	a character string defining the method used to estimate the marginal GPD; either `"mle"` for maximum likelihood of `"mom"` for method of moments or `"pwm"` for probability weighted moments methods. Defaults to `"mle"`.
`method`	a character string defining the method used to estimate the dependence measure; either `"prop"` for proportions or `"MCi"` for Monte Carlo integration (see Details).
`silent`	logical (`FALSE`); verbosity.
`R.boot`	integer; the number of samples used for the block bootstrap for the confidence intervals.
`block.length`	integer; the block length used for the block-bootstrapped confidence intervals.
`levels`	vector of probabilities; the quantiles of the bootstrap distribution of the extremal measure to be computed.

Details

The standard procedure (`method="prop"`) to estimating probabilities from a Heffernan-Tawn fit best illustrated in the bivariate context ($Y \mid X > u$):

1. sample X from an exponential distribution above $v \geq u$,

2. sample Z (residuals) from their empirical distribution,

3. compute Y using the relation $Y = \alpha \times X + X^\beta \times Z$,

4. estimate $Pr(X > v_x, Y > v_y)$ by calculating the proportion p of Y samples above v_y and multiply p with the marginal survival distribution evaluated at v_x.

With `method="MCi"` a Monte Carlo integration approach is used, where the survivor distribution of Z is evaluated at pseudo-residuals of the form

$$\frac{v_y - \alpha \times X}{X^\beta},$$

where X is sampled from an exponential distribution above v_x. Taking the mean of these survival probabilities, we get the Monte Carlo equivalent of p in the proportion approach.

Value

List containing:

`depfit`	an object of class 'stepfit'
`probs`	`probs`
`levels`	`probs` transformed to original scale of `ts`
`theta`	a matrix with *proportion* or *Monte Carlo* estimates of $\theta(x, m)$. Rows correspond to values in `probs`, columns are point estimates and bootstrap quantiles

See Also

`dep2fit`, `thetafit`, `thetaruns`

Examples

```
## generate data from an AR(1)
## with Gaussian marginal distribution
n   <- 10000
dep <- 0.5
ar    <- numeric(n)
ar[1] <- rnorm(1)
for(i in 2:n)
  ar[i] <- rnorm(1, mean=dep*ar[i-1], sd=1-dep^2)
plot(ar, type="l")
plot(density(ar))
grid <- seq(-3,3,0.01)
lines(grid, dnorm(grid), col="blue")

## rescale the margin (focus on dependence)
ar <- qlapl(pnorm(ar))
```

```
## fit the data
fit <- theta2fit(ts=ar, u.mar=0.95, u.dep=0.98)

## plot theta(x,1)
plot(fit)
abline(h=1, lty="dotted")
```

---

thetaruns *Runs estimator*

Description

Compute the empirical estimator of the extremal index using the runs method (Smith & Weissman, 1994, JRSSB).

Usage

```
thetaruns(ts, lapl = FALSE, nlag = 1,
          u.mar = 0, probs = seq(u.mar, 0.995, length.out = 30),
          method.mar = c("mle", "mom", "pwm"),
          R.boot = 0, block.length = (nlag+1) * 5, levels = c(0.025, 0.975))
```

Arguments

`ts`	a vector, the time series for which to estimate the threshold-based extremal index $\theta(x, m)$, with x a probability level and m a run-length (see details).
`lapl`	logical; is `ts` on the Laplace scale already? The default (FALSE) assumes unknown marginal distribution.
`nlag`	the run-length; an integer larger or equal to 1.
`u.mar`	marginal threshold (probability); used when transforming the time series to Laplace scale if `lapl` is FALSE; if `lapl` is TRUE, it is nevertheless used when bootstrapping, since the bootstrapped series generally do not have Laplace marginal distributions.
`probs`	vector of probabilities; the values of x for which to evaluate $\theta(x, m)$.
`method.mar`	a character string defining the method used to estimate the marginal GPD; either `"mle"` for maximum likelihood or `"mom"` for method of moments or `"pwm"` for probability weighted moments methods. Defaults to `"mle"`.
`block.length`	integer; the block length used for the block-bootstrapped confidence intervals.
`R.boot`	integer; the number of samples used for the block bootstrap.
`levels`	vector of probabilites; the quantiles of the posterior distribution of the extremal index $\theta(x, m)$ to output.

Details

Consider a stationary time series (X_t). A characterisation of the extremal index is

$$\theta(x,m) = Pr(X_1 \leq x, \ldots, X_m \leq x \mid X_0 \geq x).$$

In the limit when x and m tend to ∞ appropriately, θ corresponds to the asymptotic inverse mean cluster size. It also links the generalised extreme value distribution of the independent series (Y_t), with the same marginal distribution as (X_t),

$$G_Y(z) = G_X^{\theta}(z),$$

with G_X and G_Y the extreme value distributions of (X_t) and (Y_t) respectively.

`nlag` corresponds to the *run-length* m and `probs` is a set of values for x. The *runs* estimator is computed, which consists of counting the proportion of clusters to the number of exceedances of a threshold x; two exceedances of the threshold belong to different clusters if there are at least $m+1$ non-exceedances inbetween.

Value

An object of class '`depmeasure`' containing:

`theta`	matrix; estimates of the extremal index $\theta(x,m)$ with rows corresponding to the `probs` values of x and the columns to the runs estimate and the chosen `levels`-quantiles of the bootstrap distribution.
`nbr.exc`	numeric vector; number of exceedances for each threshold corresponding to the elements in `probs`.
`probs`	`probs`.
`levels`	numeric vector; `probs` converted to the original scale of `ts`.
`nlag`	`nlag`.

See Also

`theta2fit`, `thetafit`

Examples

```
## generate data from an AR(1)
## with Gaussian marginal distribution
n   <- 10000
dep <- 0.5
ar    <- numeric(n)
ar[1] <- rnorm(1)
for(i in 2:n)
  ar[i] <- rnorm(1, mean=dep*ar[i-1], sd=1-dep^2)
## transform to Laplace scale
ar <- qlapl(pnorm(ar))
## compute empirical estimate
theta <- thetaruns(ts=ar, u.mar=.95, probs=c(.95,.98,.99))
## output
plot(theta, ylim=c(.2,1))
abline(h=1, lty="dotted")
```

Index