Package 'SpatialExtremes'

January 14, 2010

Version 1.5-1

Date 2009-01-14

Title Modelling Spatial Extremes

Author Mathieu Ribatet <mathieu.ribatet@epfl.ch>

Maintainer Mathieu Ribatet <mathieu.ribatet@epfl.ch>

Depends R (>= 1.8.0), RandomFields

Description This package proposes several approaches for spatial extremes modelling.

License GPL (>= 2)

URL http://people.epfl.ch/mathieu.ribatet,
http://spatialextremes.r-forge.r-project.org/

Repository CRAN

Date/Publication 2010-01-14 11:58:13

R topics documented:

`anova` *Anova Tables*

Description

Computes analysis of deviance for objects of class "maxstab"

Usage

```
## S3 method for class 'maxstab':
anova(object, object2, method = "RJ", square = "chol",
...)
## S3 method for class 'spatgev':
anova(object, object2, method = "RJ", square = "chol",
...)
```

Arguments

object, object2
: Two objects of class 'maxstab' or 'spatgev'.

method
: Character string. Must be one of "CB" or "RJ" for the Chandler and Bate or the Rotnitzky and Jewell approaches respectively. See function `profile`.

square
: The choice for the matrix square root. This is only useful for the 'CB' method. Must be one of 'chol' (Cholesky) or 'svd' (Singular Value Decomposition).

...
: Other options to be passed to the `anova` function.

Details

As "maxstab" objects are fitted using pairwise likelihood, the model is misspecified. As a consequence, the likelihood ratio statistic is no longer χ^2 distributed. To compute the anova table, we use the methodology proposed by Rotnitzky and Jewell to adjust the distribution of the likelihood ratio statistic.

Value

This function returns an object of class anova. These objects represent analysis-of-deviance tables.

Author(s)

Mathieu Ribatet

References

Chandler, R. E. and Bate, S. (2007) Inference for clustered data using the independence loglikelihood *Biometrika*, **94**, 167–183.

Rotnitzky, A. and Jewell, N. (1990) Hypothesis testing of regression parameters in semiparametric generalized linear models for cluster correlated data. *Biometrika* **77**, 485–497.

See Also

`fitmaxstab`, `fitspatgev`, `profile`, `TIC`

Examples

```
##Define the coordinates of each location
n.site <- 30
locations <- matrix(rnorm(2*n.site, sd = sqrt(.2)), ncol = 2)
colnames(locations) <- c("lon", "lat")

##Simulate a max-stable process - with unit Frechet margins
data <- rmaxstab(50, locations, cov.mod = "gauss", cov11 = 100, cov12 =
25, cov22 = 220)

##Now define the spatial model for the GEV parameters
param.loc <- -10 + 2 * locations[,2]
param.scale <- 5 + 2 * locations[,1] + locations[,2]^2
```

```
param.shape <- rep(0.2, n.site)

##Transform the unit Frechet margins to GEV
for (i in 1:n.site)
  data[,i] <- frech2gev(data[,i], param.loc[i], param.scale[i],
param.shape[i])

##Define three models for the GEV margins to be fitted
loc.form <- loc ~ lat
scale.form <- scale ~ lon + I(lat^2)
shape.form <- shape ~ lon

M0 <- fitspatgev(data, locations, loc.form, scale.form, shape.form)
M1 <- fitspatgev(data, locations, loc.form, scale.form, shape.form,
shapeCoeff2 = 0)

##Model selection
anova(M0, M1)
anova(M0, M1, method = "CB", square = "svd")
```

---

condmap *Produces a conditional 2D map from a fitted max-stable process*

Description

Produces a conditional 2D map from a fitted max-stable process.

Usage

```
condmap(fitted, fix.coord, x, y, covariates = NULL, ret.per1 = 100,
ret.per2 = ret.per1, col = terrain.colors(64), plot.contour = TRUE,
...)
```

Arguments

`fitted`	An object of class `maxstab`. Most often, it will be the output of the function `fitmaxstab`.
`fix.coord`	The spatial coordinates of the location from which the conditional quantile is computed.
`x,y`	Numeric vector defining the grid at which the levels are computed.
`covariates`	An array specifying the covariates at each grid point defined by `x` and `y`. If `NULL`, no covariate is needed. See `map` to see how to build it.
`ret.per1,ret.per2`	Numerics giving the return period for which the quantile map is plotted. See details.
`col`	A list of colors such as that generated by 'rainbow', 'heat.colors', 'topo.colors', 'terrain.colors' or similar functions.
`plot.contour`	Logical. If `TRUE` (default), contour lines are added to the plot.
`...`	Several arguments to be passed to the `image` function.

Details

The function solves the following equation:

$$\Pr\left[Z(x_2) > z_2 | Z(x_1) > z_1\right] = \frac{1}{T_2}$$

where $z_1 = -1/\log(1 - 1/T_1)$.

In other words, it computes, given that at location x_1 we exceed the level z_1, the levels which is expected to be exceeded in average every T_2 year.

Value

A plot. Additionally, a list with the details for plotting the map is returned invisibly.

Author(s)

Mathieu Ribatet

See Also

`map`, `filled.contour`, `heatmap`, `heat.colors`, `topo.colors`, `terrain.colors`, `rainbow`

Examples

```
##Define the coordinate of each location
n.site <- 30
locations <- matrix(runif(2*n.site, 0, 10), ncol = 2)
colnames(locations) <- c("lon", "lat")

##Simulate a max-stable process - with unit Frechet margins
data <- rmaxstab(50, locations, cov.mod = "whitmat", sill = 1, range =
2, smooth = 1)

##Now define the spatial model for the GEV parameters
param.loc <- -10 - 4 * locations[,1] + locations[,2]^2
param.scale <- 5 + locations[,2] + locations[,1]^2 / 10
param.shape <- rep(.2, n.site)

##Transform the unit Frechet margins to GEV
for (i in 1:n.site)
  data[,i] <- frech2gev(data[,i], param.loc[i], param.scale[i],
param.shape[i])

##Define a model for the GEV margins to be fitted
##shape ~ 1 stands for the GEV shape parameter is constant
##over the region
loc.form <- loc ~ lon + I(lat^2)
scale.form <- scale ~ lat + I(lon^2)
shape.form <- shape ~ 1
```

```
##  1- Fit a max-stable process
fitted <- fitmaxstab(data, locations, "whitmat", loc.form, scale.form,
                     shape.form, sill = 1, std.err.type = "none")

cond.coord <- c(5.1, 5.1)
condmap(fitted, cond.coord, seq(0, 10, length = 25), seq(0,10, length
 =25), ret.per1 = 100, ret.per2 = 1.5)
points(t(cond.coord), pch = "*", col = 2, cex = 2)
```

`covariance` *Defines and computes covariance functions*

Description

This function defines and computes several covariance function either from a fitted "max-stable" model; either by specifying directly the covariance parameters.

Usage

```
covariance(fitted, sill, range, smooth, smooth2 = NULL, cov.mod =
"whitmat", plot = TRUE, dist, xlab, ylab, ...)
```

Arguments

`fitted`	An object of class "maxstab". Most often this will be the output of the `fitmaxstab` function. May be missing if `sill`, `range`, `smooth` and `cov.mod` are given.
`sill,range,smooth,smooth2`	The sill, scale and smooth parameters for the covariance function. May be missing if `fitted` is given.
`cov.mod`	Character string. The name of the covariance model. Must be one of "whitmat", "cauchy", "powexp", "bessel" or "caugen" for the Whittle-Matern, Cauchy, Powered Exponential, Bessel and Generalized Cauchy models. May be missing if `fitted` is given.
`plot`	Logical. If `TRUE` (default) the covariance function is plotted.
`dist`	A numeric vector corresponding to the distance at which the covariance function should be evaluated. May be missing.
`xlab,ylab`	The x-axis and y-axis labels. May be missing.
`...`	Several option to be passed to the `plot` function.

Details

Currently, four covariance functions are defined: the Whittle-Matern, powered exponential (also known as "stable"), Cauchy and Bessel models. These covariance functions are defined as follows:

Whittle-Matern $\rho(h) = sill \frac{2^{1-smooth}}{\Gamma(smooth)} \left(\frac{h}{range}\right)^{smooth} K_{smooth}\left(\frac{h}{range}\right)$

Powered Exponential $\rho(h) = sill \exp\left[-\left(\frac{h}{range}\right)^{smooth}\right]$

Cauchy $\rho(h) = sill\left[1 + \left(\frac{h}{range}\right)^2\right]^{-smooth}$

Bessel $\rho(h) = sill\left(\frac{2range}{h}\right)^{smooth}\Gamma(smooth+1)J_{smooth}\left(\frac{h}{range}\right)$

Generalized Cauchy $\rho(h) = sill\left\{1 + \left(\frac{h}{range}\right)^{smooth2}\right\}^{-smooth/smooth2}$

where Γ is the gamma function, K_{smooth} and J_{smooth} are both modified Bessel functions of order $smooth$.

Value

This function returns the covariance function. Eventually, if `dist` is given, the covariance function is computed for each distance given by `dist`. If `plot = TRUE`, the covariance function is plotted.

Author(s)

Mathieu Ribatet

Examples

```
## 1- Calling covariance using fixed covariance parameters
covariance(sill = 1, range = 1, smooth = 0.5, cov.mod = "whitmat")
covariance(sill = 0.5, range = 1, smooth = 0.5, cov.mod = "whitmat",
  dist = seq(0,5, 0.2), plot = FALSE)

## 2- Calling covariance from a fitted model
##Define the coordinate of each location
n.site <- 30
locations <- matrix(runif(2*n.site, 0, 10), ncol = 2)
colnames(locations) <- c("lon", "lat")

##Simulate a max-stable process - with unit Frechet margins
data <- rmaxstab(30, locations, cov.mod = "whitmat", sill = 1, range =
3, smooth = 1)

##Fit a max-stable model
fitted <- fitmaxstab(data, locations, "whitmat", std.err.type = "none",
sill = 1)
covariance(fitted, ylim = c(0, 1))
covariance(sill = 1, range = 3, smooth = 1, cov.mod = "whitmat", add =
TRUE, col = 3)
title("Whittle-Matern covariance function")
legend("topright", c("Theo.", "Fitted"), lty = 1, col = c(3,1), inset =
.05)
```

cv *Estimates the penalty coefficient from the cross-validation criterion*

Description

Estimates the penalty coefficient from the cross-validation criterion.

Usage

```
cv(y, x, knots, degree, plot = TRUE, n.points = 150, ...)
```

Arguments

`y`	The response vector.
`x`	A vector/matrix giving the values of the predictor variable(s). If `x` is a matrix, each row corresponds to one observation.
`knots`	A vector givint the coordinates of the knots.
`degree`	The degree of the penalized smoothing spline.
`plot`	Logical. If `TRUE` (default), the evolution of the CV score as the penalty increases is plotted.
`n.points`	A numeric giving the number of CV computations needed to produce the plot.
`...`	Options to be passed to the `nlm` function.

Details

For every linear smoother e.g. $\hat{y} = S_\lambda y$, the cross-validation criterion consists in minimizing the following quantity:

$$CV(\lambda) = \sum_{i=1}^{n} \left(\frac{y_i - \hat{y}_i}{1 - S_{\lambda,ii}} \right)^2$$

where λ is the penalty coefficient, n the number of observations and $S_{\lambda,ii}$ the i-th diagonal element of the matrix S_λ.

Value

A list with components 'penalty', 'cv' and 'nlm.code' which give the location of the minimum, the value of the cross-validation criterion at that point and the code returned by the `link{nlm}` function - useful to assess for convergence issues.

Author(s)

Mathieu Ribatet

References

Ruppert, D. Wand, M.P. and Carrol, R.J. (2003) *Semiparametric Regression* Cambridge Series in Statistical and Probabilistic Mathematics.

See Also

cv

Examples

```
n <- 200
x <- runif(n)
fun <- function(x) sin(3 * pi * x)
y <- fun(x) + rnorm(n, 0, sqrt(0.4))
knots <- quantile(x, prob = 1:(n/4) / (n/4 + 1))
cv(y, x, knots = knots, degree = 3)
```

`distance` *Computes distance between pairs of locations*

Description

This function computes euclidean distance or distance vector for each pair of a set of spatial locations.

Usage

```
distance(coord, vec = FALSE)
```

Arguments

`coord`	A matrix representing the coordinates of each locations. Each row corresponds to one location.
`vec`	Logical. If `FALSE` (default), the euclidean distance is computed. Otherwise, “distance vectors” are returned.

Value

If `vec = FALSE`, this function returns a vector that gives the euclidean distance for each pair of locations. Otherwise, this is a matrix where each column correspond to one dimension - i.e. longitude, latitude, . . .

Author(s)

Mathieu Ribatet

See Also

dist

Examples

```
coords <- cbind(seq(0, 10, by = 2), seq(10, 0, by = -2))
distance(coords)
distance(coords, vec = TRUE)
```

---

extcoeff *Plots the extremal coefficient*

Description

Plots the extremal coefficient evolution as the coordinates evolves.

Usage

```
extcoeff(fitted, cov.mod, param, n = 200, xlab, ylab, ...)
```

Arguments

fitted	A object of class "maxstab". Most often, it will be the output of the function fitmaxstab. If missing, then cov.mod and param should be supplied.
cov.mod	A character string corresponding the the covariance model in the max-stable representation. Must be one of "gauss" for the Smith's model; or "whitmat", "cauchy" or "powexp" for the Whittle-Matern, the Cauchy and the Powered Exponential covariance family with the Schlather's model. May be missing if fitted is given.
param	Numeric vector of length 3. The parameters for the Smith's or Schlather model - i.e. c(cov11, cov12, cov22) or c(sill, range, smooth). Please respect this order.
n	Numeric. n^2 corresponds to the total number of estimated extremal coefficients for the contour plot.
xlab,ylab	The x-axis and y-axis labels. May be missing.
...	Several options to be passed to the contour function.

Value

A plot.

Author(s)

Mathieu Ribatet

See Also

fitmaxstab

Examples

```
## 1- Random field generation
n.site <- 30
locations <- matrix(runif(2*n.site, 0, 10), ncol = 2)
colnames(locations) <- c("lon", "lat")

data <- rmaxstab(60, locations, cov.mod = "whitmat", sill = 1, range =
3, smooth = 1)

## 2- Fit a max-stable processes
schlather <- fitmaxstab(data, locations, "whitmat", sill = 1)

## 3- Plot the extremal coefficient
extcoeff(schlather)
```

`fitcovariance` *Estimates the covariance function for the Schlather's model*

Description

Estimates the covariance function for the Schlather's model using non-parametric estimates of the pairwise extremal coefficients.

Usage

```
fitcovariance(data, coord, cov.mod, marge = "emp", control = list(),
..., start, weighted = TRUE)
```

Arguments

`data`	A matrix representing the data. Each column corresponds to one location.
`coord`	A matrix that gives the coordinates of each location. Each row corresponds to one location.
`cov.mod`	A character string corresponding the the covariance model in the Schlather's model. Must be one of "whitmat", "cauchy", "powexp", "bessel" or "caugen" for the Whittle-Matern, the Cauchy, the Powered Exponential, the Bessel and the Generalized Cauchy correlation families.
`marge`	Character string specifying how margins are transformed to unit Frechet. Must be one of "emp", "frech" or "mle" - see function `fitextcoeff`.
`control`	The control arguments to be passed to the `optim` function.
`...`	Optional arguments to be passed to the `optim` function.
`start`	A named list giving the initial values for the parameters over which the weighted sum of square is to be minimized. If `start` is omitted the routine attempts to find good starting values.
`weighted`	Logical. Should weighted least squares be used?

Details

The fitting procedure is based on weighted least squares. More precisely, the fitting criteria is to minimize:

$$\sum_{i,j} \left(\frac{\tilde{\theta}_{i,j} - \hat{\theta}_{i,j}}{s_{i,j}} \right)^2$$

where $\tilde{\theta}_{i,j}$ is a non parametric estimate of the extremal coefficient related to location `i` and `j`, $\hat{\theta}_{i,j}$ is the fitted extremal coefficient derived from the Schlather's model and $s_{i,j}$ are the standard errors related to the estimates $\tilde{\theta}_{i,j}$.

Value

An object of class maxstab.

Author(s)

Mathieu Ribatet

References

Smith, R. L. (1990) Max-stable processes and spatial extremes. Unpublished manuscript.

See Also

`fitcovmat`, `fitmaxstab`, `fitextcoeff`

Examples

```
n.site <- 50
locations <- matrix(runif(2*n.site, 0, 10), ncol = 2)
colnames(locations) <- c("lon", "lat")

##Simulating a max-stable process using RandomFields
##This is the Schlather's approach
data <- rmaxstab(50, locations, cov.mod = "whitmat", sill = 1, range =
30, smooth = 1)

fitcovariance(data, locations, "whitmat")
```

`fitcovmat` *Estimates the covariance matrix for the Smith's model*

Description

Estimates the covariance matrix for the Smith's model using non-parametric estimates of the pairwise extremal coefficients.

Usage

```
fitcovmat(data, coord, marge = "emp", iso = FALSE, control = list(),
...., start, weighted = TRUE)
```

Arguments

`data`	A matrix representing the data. Each column corresponds to one location.
`coord`	A matrix that gives the coordinates of each location. Each row corresponds to one location.
`marge`	Character string specifying how margins are transformed to unit Frechet. Must be one of "emp", "frech" or "mle" - see function `fitextcoeff`.
`iso`	Logical. If `TRUE`, isotropy is supposed. Otherwise (default), anisotropy is allowed.
`control`	The control arguments to be passed to the `optim` function.
`...`	Optional arguments to be passed to the `optim` function.
`start`	A named list giving the initial values for the parameters over which the weighted sum of square is to be minimized. If `start` is omitted the routine attempts to find good starting values.
`weighted`	Logical. Should weighted least squares be used?

Details

The fitting procedure is based on weighted least squares. More precisely, the fitting criteria is to minimize:

$$\sum_{i,j} \left(\frac{\tilde{\theta}_{i,j} - \hat{\theta}_{i,j}}{s_{i,j}} \right)^2$$

where $\tilde{\theta}_{i,j}$ is a non parametric estimate of the extremal coefficient related to location `i` and `j`, $\hat{\theta}_{i,j}$ is the fitted extremal coefficient derived from the Smith's model and $s_{i,j}$ are the standard errors related to the estimates $\tilde{\theta}_{i,j}$.

Value

An object of class maxstab.

Author(s)

Mathieu Ribatet

References

Smith, R. L. (1990) Max-stable processes and spatial extremes. Unpublished manuscript.

See Also

`fitcovariance`, `fitmaxstab`, `fitextcoeff`

Examples

```
n.site <- 50
n.obs <- 100
locations <- matrix(runif(2*n.site, 0, 40), ncol = 2)
colnames(locations) <- c("lon", "lat")

## Simulate a max-stable process - with unit Frechet margins
data <- rmaxstab(50, locations, cov.mod = "gauss", cov11 = 200, cov12 =
0, cov22 = 200)

fitcovmat(data, locations)

##Force an isotropic model
fitcovmat(data, locations, iso = TRUE)
```

---

fitextcoeff *Non parametric estimators of the extremal coefficient function*

Description

Estimates non parametrically the extremal coefficient function.

Usage

```
fitextcoeff(data, coord, ..., estim = "ST", marge = "emp", prob = 0,
plot = TRUE, loess = TRUE, method = "BFGS", std.err = TRUE, xlab,
ylab, angles = NULL, identify = FALSE)
```

Arguments

`data`	A matrix representing the data. Each column corresponds to one location.
`coord`	A matrix that gives the coordinates of each location. Each row corresponds to one location.
`...`	Additional options to be passed to the `plot` function.
`estim`	Character string specifying the estimator to be used. Must be one of "ST" (Schlather and Tawn) or "Smith".
`marge`	Character string specifying how margins are transformed to unit Frechet. Must be one of "emp", "mle" or "none" - see Details
`prob`	The probability related to the threshold. Only useful with the `ST` estimator.
`plot`	Logical. If `TRUE` (default), the extremal coefficient function is plotted
`loess`	If `TRUE` (default), a local polynomial curve is plotted - see function `loess`.
`method`	The optimizer used when fitting the GEV distribution to data. See function `gevmle`.
`std.err`	Logical. If `TRUE`, standard errors are computed. Note that standard errors are not available with the "ST" estimator.

`xlab,ylab`	The x-axis and y-axis labels. May be missing.
`angles`	A numeric vector. A partition of the interval $(-\pi, \pi)$ to help detecting anisotropy.
`identify`	Logical. If `TRUE`, users can use the `identify` function to identify pairs of stations on the plot.

Details

During the estimation procedure, data need to be transformed to unit Frechet margins firts. This can be done in two different ways ; by using the empirical CDF or the GEV ML estimates.

If `marge = "emp"`, then the data are transformed using the following relation:

$$z_i = -\frac{1}{\log(F(y_i))}$$

where y_i are the observations available at location i, F is the empirical CDF and z_i are the observations transformed to unit Frechet scale.

If `marge = "mle"`, then the data are transformed using the MLE of the GEV distribution - see function `gev2frech`.

Lastly, if data are already supposed to be unit Frechet, then no transformation is performed if one passed the option `marge = "frech"`.

If `data` are already componentwise maxima, `prob` should be zero. Otherwise, users have to define a threshold z (large enough) where univariate extreme value arguments are relevant. We define `prob` such that $\Pr[Z \leq z] = prob$.

Value

Plots the extremal coefficient function and returns the points used for the plot. If `loess = TRUE`, the output is a list with argument "ext.coeff" and "loess".

Author(s)

Mathieu Ribatet

References

Schlather, M. and Tawn, J. A. (2003) A dependence measure for multivariate and spatial extreme values: Properties and inference. Biometrika **90**(1):139–156.

Smith, R. L. (1990) Max-stable processes and spatial extremes. Unpublished manuscript.

See Also

`madogram`

Examples

```
n.site <- 30
locations <- matrix(runif(2*n.site, 0, 10), ncol = 2)
colnames(locations) <- c("lon", "lat")

##Simulate a max-stable process - with unit Frechet margins
data <- rmaxstab(50, locations, cov.mod = "gauss", cov11 = 10, cov12 =
40, cov22 = 220)

##Plot the extremal coefficient function
op <- par(mfrow=c(1,2))
fitextcoeff(data, locations, estim = "Smith")
fitextcoeff(data, locations, angles = seq(-pi, pi, length = 4), estim = "Smith")
par(op)
```

`fitmaxstab` *Fits a max-stable process to data*

Description

This function fits max-stable processes to data using pairwise likelihood. Two max-stable characterisations are available: the Smith and Schlather representations.

Usage

```
fitmaxstab(data,  coord, cov.mod, loc.form, scale.form,
shape.form, marg.cov = NULL, iso = FALSE, ..., fit.marge = FALSE, warn
= TRUE, method = "Nelder", start, control = list(), weights = NULL,
std.err.type = "score", corr = FALSE)
```

Arguments

`data` A matrix representing the data. Each column corresponds to one location.

`coord` A matrix that gives the coordinates of each location. Each row corresponds to one location.

`cov.mod` A character string corresponding the the covariance model in the max-stable representation. Must be one of "gauss" for the Smith's model; or "whitmat", "cauchy", "powexp", "bessel" or "caugen" for the Whittle-Matern, the Cauchy, the Powered Exponential, the Bessel and the Generalized Cauchy correlation families with the Schlather's model; "brown" for Brown-Resnick processes.

`loc.form, scale.form, shape.form`
R formulas defining the spatial linear model for the GEV parameters. May be missing. See section details.

`marg.cov` Matrix with named columns giving additional covariates for the GEV parameters. If `NULL`, no extra covariates are used.

`iso` Logical. If `TRUE` an isotropic model is fitted to data. Otherwise (default), anisotropy is allowed. Currently, this is only implemented for the Smith's model.

`...`	Several arguments to be passed to the `optim`, `nlm` or `nlminb` functions. See section details.
`fit.marge`	Logical. If `TRUE`, the GEV parameters are estimated pointwise or using the formulas given by `loc.form`, `scale.form` and `shape.form`. If `FALSE`, observations are supposed to be unit Frechet distributed. Note that when formulas are given, `fit.marge` is automatically set to `TRUE`.
`warn`	Logical. If `TRUE` (default), users are warned if the log-likelihood is infinite at starting values and/or problems arised while computing the standard errors.
`method`	The method used for the numerical optimisation procedure. Must be one of `BFGS`, `Nelder-Mead`, `CG`, `L-BFGS-B`, `SANN`, `nlm` or `nlminb`. See `optim` for details. Please note that passing `nlm` or `nlminb` will use the `nlm` or `nlminb` functions instead of `optim`.
`start`	A named list giving the initial values for the parameters over which the pairwise likelihood is to be minimized. If `start` is omitted the routine attempts to find good starting values - but might fail.
`control`	A list giving the control parameters to be passed to the `optim` function.
`weights`	A numeric vector specifying the weights in the pairwise likelihood - and so has length the number of pairs. If `NULL` (default), no weighting scheme is used.
`std.err.type`	Character string. Must be one of "none", "score", "grad". If "none", standard errors are not computed. Otherwise, standard errors are estimated using the sandwich estimates - see section Details.
`corr`	Logical. If `TRUE` (non default), the asymptotic correlation matrix is computed.

Details

As spatial data often deal with a large number of locations, it is impossible to write analytically the joint distribution. Consequently, the fitting procedure substitutes the "full likelihood" for the pairwise likelihood.

Let define $L_{i,j}(x_{i,j}, \theta)$ the likelihood for site i and j, where $i = 1, \ldots, N-1$, $j = i+1, \ldots, N$, N is the number of site within the region and $x_{i,j}$ are the joint observations for site i and j. Then the pairwise likelihood $PL(\theta)$ is defined by:

$$\ell_P = \log PL(\theta) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \log L_{i,j}(x_{i,j}, \theta)$$

As pairwise likelihood is an approximation of the “full likelihood”, standard errors cannot be computed directly by the inverse of the Fisher information matrix. Instead, a sandwich estimate must be used to account for model mispecification e.g.

$$\hat{\theta} \sim N(\theta, H^{-1} J H^{-1})$$

where H is the Fisher information matrix (computed from the misspecified model) and J is the variance of the score function.

H is easily estimated using the observed Hessian matrix given by the `optim` function. Estimation of J is much more difficult. Currently, we propose two different strategies to estimate J.

`grad` J is estimated from the gradient e.g. $J = \sum_{i=1}^{n} \left\{ \nabla \ell_P(\hat{\theta}; y_i) \nabla \ell_P(\hat{\theta}, y_i)^T \right\}$.

`score` J is estimated directly from the variance of the observed score function.

Value

This function returns a object of class `maxstab`. Such objects are list with components:

`fitted.values` A vector containing the estimated parameters.

`std.err` A vector containing the standard errors.

`fixed` A vector containing the parameters of the model that have been held fixed.

`param` A vector containing all parameters (optimised and fixed).

`deviance` The (pairwise) deviance at the maximum pairwise likelihood estimates.

`corr` The correlation matrix.

`convergence, counts, message` Components taken from the list returned by `optim` - for the `mle` method.

`data` The data analysed.

`model` The max-stable characterisation used.

`fit.marge` A logical that specifies if the GEV margins were estimated.

`cov.fun` The estimated covariance function - for the Schlather model only.

`extCoeff` The estimated extremal coefficient function.

`cov.mod` The covariance model for the spatial structure.

Warning

When using reponse surfaces to model spatially the GEV parameters, the likelihood is pretty rough so that the general purpose optimization routines may fail. It is your responsability to check if the numerical optimization succeeded or not. I tried, as best as I can, to provide warning messages if the optimizers failed but in some cases, no warning will appear!

Author(s)

Mathieu Ribatet

References

Cox, D. R. and Reid, N. (2004) A note on pseudo-likelihood constructed from marginal densities. *Biometrika* **91**: 729–737.

Gholam–Rezaee, M. (2009) Spatial extreme value: A composite likelihood. PhD Thesis. Ecole Polytechnique Federale de Lausanne.

Padoan, S. A. (2008) Computational Methods for Complex Problems in Extreme Value Theory. PhD Thesis. University of Padova.

Padoan, S. A., Ribatet, M. and Sisson, S. A. (2008) Likelihood-based inference for max-stable processes. Under review. *Journal of the American Statistical Association (Theory and Methods)*.

Schlather, M. (2002) Models for Stationary Max-Stable Random Fields. *Extremes* **5**:1, 33–44.

Smith, R. L. (1990) Max-stable processes and spatial extremes. Unpublished.

Examples

```
## Not run:
##Define the coordinate of each location
n.site <- 30
locations <- matrix(runif(2*n.site, 0, 10), ncol = 2)
colnames(locations) <- c("lon", "lat")

##Simulate a max-stable process - with unit Frechet margins
data <- rmaxstab(40, locations, cov.mod = "whitmat", sill = 1, range = 30,
smooth = 0.5)

##Now define the spatial model for the GEV parameters
param.loc <- -10 + 2 * locations[,2]
param.scale <- 5 + 2 * locations[,1] + locations[,2]^2
param.shape <- rep(0.2, n.site)

##Transform the unit Frechet margins to GEV
for (i in 1:n.site)
  data[,i] <- frech2gev(data[,i], param.loc[i], param.scale[i],
param.shape[i])

##Define a model for the GEV margins to be fitted
##shape ~ 1 stands for the GEV shape parameter is constant
##over the region
loc.form <- loc ~ lat
scale.form <- scale ~ lon + I(lat^2)
shape.form <- shape ~ 1

##Fit a max-stable process using the Schlather's model
fitmaxstab(data, locations, "whitmat", loc.form, scale.form,
           shape.form)

## Model without any spatial structure for the GEV parameters
## Be careful this could be *REALLY* time consuming
fitmaxstab(data, locations, "whitmat")

##  Fixing the smooth parameter of the Whittle-Matern family
##  to 0.5 - e.g. considering exponential family. We suppose the data
##  are unit Frechet here.
fitmaxstab(data, locations, "whitmat", smooth = 0.5, fit.marge = FALSE)

##  Fitting a penalized smoothing splines for the margins with the
##     Smith's model
data <- rmaxstab(40, locations, cov.mod = "gauss", cov11 = 100, cov12 =
                 25, cov22 = 220)

##     And transform it to ordinary GEV margins with a non-linear
##     function
fun <- function(x)
  2 * sin(pi * x / 4) + 10
fun2 <- function(x)
  (fun(x) - 7 ) / 15
```

```
param.loc <- fun(locations[,2])
param.scale <- fun(locations[,2])
param.shape <- fun2(locations[,1])

##Transformation from unit Frechet to common GEV margins
for (i in 1:n.site)
  data[,i] <- frech2gev(data[,i], param.loc[i], param.scale[i],
param.shape[i])

##Defining the knots, penalty, degree for the splines
n.knots <- 5
knots <- quantile(locations[,2], prob = 1:n.knots/(n.knots+1))
knots2 <- quantile(locations[,1], prob = 1:n.knots/(n.knots+1))

##Be careful the choice of the penalty (i.e. the smoothing parameter)
##may strongly affect the result Here we use p-splines for each GEV
##parameter - so it's really CPU demanding but one can use 1 p-spline
##and 2 linear models.
##A simple linear model will be clearly faster...
loc.form <- y ~ rb(lat, knots = knots, degree = 3, penalty = .5)
scale.form <- y ~ rb(lat, knots = knots, degree = 3, penalty = .5)
shape.form <- y ~ rb(lon, knots = knots2, degree = 3, penalty = .5)

fitted <- fitmaxstab(data, locations, "gauss", loc.form, scale.form, shape.form,
                     control = list(ndeps = rep(1e-6, 24), trace = 10),
                     std.err.type = "none", method = "BFGS")
fitted
op <- par(mfrow=c(1,3))
plot(locations[,2], param.loc, col = 2, ylim = c(7, 14),
     ylab = "location parameter", xlab = "latitude")
plot(fun, from = 0, to = 10, add = TRUE, col = 2)
points(locations[,2], predict(fitted)[,"loc"], col = "blue", pch = 5)
new.data <- cbind(lon = seq(0, 10, length = 100), lat = seq(0, 10, length = 100))
lines(new.data[,1], predict(fitted, new.data)[,"loc"], col = "blue")
legend("topleft", c("true values", "predict. values", "true curve", "predict. curve"),
       col = c("red", "blue", "red", "blue"), pch = c(1, 5, NA, NA), inset = 0.05,
       lty = c(0, 0, 1, 1), ncol = 2)

plot(locations[,2], param.scale, col = 2, ylim = c(7, 14),
     ylab = "scale parameter", xlab = "latitude")
plot(fun, from = 0, to = 10, add = TRUE, col = 2)
points(locations[,2], predict(fitted)[,"scale"], col = "blue", pch = 5)
lines(new.data[,1], predict(fitted, new.data)[,"scale"], col = "blue")
legend("topleft", c("true values", "predict. values", "true curve", "predict. curve"),
       col = c("red", "blue", "red", "blue"), pch = c(1, 5, NA, NA), inset = 0.05,
       lty = c(0, 0, 1, 1), ncol = 2)

plot(locations[,1], param.shape, col = 2,
     ylab = "shape parameter", xlab = "longitude")
plot(fun2, from = 0, to = 10, add = TRUE, col = 2)
points(locations[,1], predict(fitted)[,"shape"], col = "blue", pch = 5)
lines(new.data[,1], predict(fitted, new.data)[,"shape"], col = "blue")
```

```
legend("topleft", c("true values", "predict. values", "true curve", "predict. curve"),
       col = c("red", "blue", "red", "blue"), pch = c(1, 5, NA, NA), inset = 0.05,
       lty = c(0, 0, 1, 1), ncol = 2)
par(op)

## End(Not run)
```

`fitspatgev` *MLE for a spatial GEV model*

Description

This function derives the MLE of a spatial GEV model.

Usage

```
fitspatgev(data, covariables, loc.form, scale.form, shape.form, ...,
start, control = list(maxit = 10000), method = "Nelder", std.err.type =
"score", warn = TRUE, corr = FALSE)
```

Arguments

`data` A matrix representing the data. Each column corresponds to one location.

`covariables` Matrix with named columns giving required covariates for the GEV parameter models.

`loc.form, scale.form, shape.form` R formulas defining the spatial models for the GEV parameters. See section details.

`start` A named list giving the initial values for the parameters over which the pairwise likelihood is to be minimized. If `start` is omitted the routine attempts to find good starting values - but might fail.

`...` Several arguments to be passed to the `optim` functions. See section details.

`control` The control argument to be passed to the `optim` function.

`method` The method used for the numerical optimisation procedure. Must be one of `BFGS`, `Nelder-Mead`, `CG`, `L-BFGS-B` or `SANN`. See `optim` for details.

`std.err.type` Character string. Must be one of "score", "grad" or "none". If none, no standard errors are computed.

`warn` Logical. If `TRUE` (default), users will be warned if the starting values lie in a zero density region.

`corr` Logical. If `TRUE`, the asymptotic correlation matrix is computed.

Details

A kind of "spatial" GEV model can be defined by using response surfaces for the GEV parameters. For instance, the GEV location parameters are defined through the following equation:

$$\mu = X_\mu \beta_\mu$$

where X_μ is the design matrix and β_μ is the vector parameter to be estimated. The GEV scale and shape parameters are defined accordingly to the above equation.

The log-likelihood for the GEV spatial model is consequently defined as follows:

$$\ell(\beta) = \sum_{i=1}^{n.site} \sum_{j=1}^{n.obs} \log f(y_{i,j}; \theta_i)$$

where θ_i is the vector of the GEV parameters for the i-th site.

Most often, there will be some dependence between stations. However, it can be seen from the log-likelihood definition that we supposed that the stations are mutually independent. Consequently, to get reliable standard error estimates, these standard errors are estimated with their sandwich estimates.

Value

An object of class `spatgev`. Namely, this is a list with the following arguments:

`fitted.values`
: The parameter estimates.

`param`
: All the parameters e.g. parameter estimates and fixed parameters.

`std.err`
: The standard errors.

`var.cov`
: The asymptotic MLE variance covariance matrix.

`counts,message,convergence`
: Some information about the optimization procedure.

`logLik,deviance`
: The log-likelihood and deviance values.

`loc.form, scale.form, shape.form`
: The formulas defining the spatial models for the GEV parameters.

`covariables`
: The covariables used for the spatial models.

`ihessian`
: The inverse of the Hessian matrix of the negative log-likelihood.

`jacobian`
: The variance covariance matrix of the score.

Author(s)

Mathieu Ribatet

Examples

```
## 1- Simulate a max-stable random field
n.site <- 35
locations <- matrix(runif(2*n.site, 0, 10), ncol = 2)
colnames(locations) <- c("lon", "lat")
data <- rmaxstab(50, locations, cov.mod = "whitmat", sill = 1, range =
  3, smooth = 0.5)

## 2- Transformation to non unit Frechet margins
param.loc <- -10 + 2 * locations[,2]
param.scale <- 5 + 2 * locations[,1]
param.shape <- rep(0.2, n.site)
for (i in 1:n.site)
  data[,i] <- frech2gev(data[,i], param.loc[i], param.scale[i],
param.shape[i])

## 3- Fit a ''spatial GEV'' mdoel to data with the following models for
##    the GEV parameters
form.loc <- loc ~ lat
form.scale <- scale ~ lon
form.shape <- shape ~ 1

fitspatgev(data, locations, form.loc, form.scale, form.shape)
```

`fmadogram` *Computes the F-madogram*

Description

Computes the F-madogram for max-stable processes.

Usage

```
fmadogram(data, coord, fitted, n.bins, which = c("mado", "ext"), xlab,
ylab, col = c(1, 2), angles = NULL, marge = "emp", add = FALSE, ...)
```

Arguments

`data`	A matrix representing the data. Each column corresponds to one location.
`coord`	A matrix that gives the coordinates of each location. Each row corresponds to one location.
`fitted`	An object of class maxstab - usually the output of the `fitmaxstab` function. May be missing.
`n.bins`	The number of bins to be used. If missing, pairwise F-madogram estimates will be computed.
`which`	A character vector of maximum size 2. It specifies if the madogram and/or the extremal coefficient functions have to be plotted.

`xlab,ylab`	The x-axis and y-axis labels. May be missing. Note that `ylab` must have the same length has `which`.
`col`	The colors used for the points and optionnaly the fitted curve.
`angles`	A numeric vector. A partition of the interval $(-\pi, \pi)$ to help detecting anisotropy.
`marge`	Character string. If 'emp', the probabilities of non exceedances are estimated using the empirical CDF. If 'mle' (default), maximum likelihood estimates are used.
`add`	Logical. If `TRUE`, the plot is added to the current figure; otherwhise (default) a new plot is computed.
`...`	Additional options to be passed to the `plot` function.

Details

Let $Z(x)$ be a stationary process. The F-madogram is defined as follows:

$$\nu(h) = \frac{1}{2}\mathrm{E}\left[|F(Z(x+h)) - F(Z(x))|\right]$$

The extremal coefficient $\theta(h)$ satisfies:

$$\theta(h) = \frac{1 + 2\nu(h)}{1 - 2\nu(h)}$$

Value

A graphic and (invisibly) a matrix with the lag distances, the F-madogram and extremal coefficient estimates.

Author(s)

Mathieu Ribatet

References

Cooley, D., Naveau, P. and Poncet, P. (2006) Variograms for spatial max-stable random fields. *Dependence in Probability and Statistics*, 373–390.

See Also

`madogram`, `lmadogram`

Examples

```
n.site <- 15
locations <- matrix(runif(2*n.site, 0, 10), ncol = 2)
colnames(locations) <- c("lon", "lat")

##Simulate a max-stable process - with unit Frechet margins
data <- rmaxstab(40, locations, cov.mod = "whitmat", sill = 1, range = 1,
```

```
smooth = 2)

##Compute the F-madogram
fmadogram(data, locations)

##Compare the F-madogram with a fitted max-stable process
fitted <- fitmaxstab(data, locations, "whitmat", sill = 1)
fmadogram(fitted = fitted, which = "ext")
```

`gcv` *Estimates the penalty coefficient from the generalized cross-validation criterion*

Description

Estimates the penalty coefficient from the generalized cross-validation criterion.

Usage

```
gcv(y, x, knots, degree, plot = TRUE, n.points = 150, ...)
```

Arguments

`y`	The response vector.
`x`	A vector/matrix giving the values of the predictor variable(s). If `x` is a matrix, each row corresponds to one observation.
`knots`	A vector givint the coordinates of the knots.
`degree`	The degree of the penalized smoothing spline.
`plot`	Logical. If `TRUE` (default), the evolution of the CV score as the penalty increases is plotted.
`n.points`	A numeric giving the number of CV computations needed to produce the plot.
`...`	Options to be passed to the `nlm` function.

Details

For every linear smoother e.g. $\hat{y} = S_\lambda y$, the cross-validation criterion consists in minimizing the following quantity:

$$GCV(\lambda) = \frac{n||y - \hat{y}||^2}{(n - tr(S_\lambda))^2}$$

where λ is the penalty coefficient, n the number of observations and $tr(S_\lambda)$ is the trace of the matrix S_λ.

Value

A list with components 'penalty', 'gcv' and 'nlm.code' which give the location of the minimum, the value of the cross-validation criterion at that point and the code returned by the `link{nlm}` function - useful to assess for convergence issues.

Author(s)

Mathieu Ribatet

References

Ruppert, D. Wand, M.P. and Carrol, R.J. (2003) *Semiparametric Regression* Cambridge Series in Statistical and Probabilistic Mathematics.

See Also

`cv`

Examples

```
n <- 200
x <- runif(n)
fun <- function(x) sin(3 * pi * x)
y <- fun(x) + rnorm(n, 0, sqrt(0.4))
knots <- quantile(x, prob = 1:(n/4) / (n/4 + 1))
gcv(y, x, knots = knots, degree = 3)
```

`Generalized Extreme Value Distribution`
The Generalized Extreme Value Distribution

Description

Density, distribution function, quantile function and random generation for the GP distribution with location equal to 'loc', scale equal to 'scale' and shape equal to 'shape'.

Usage

```
rgev(n, loc = 0, scale = 1, shape = 0)
pgev(q, loc = 0, scale = 1, shape = 0, lower.tail = TRUE)
qgev(p, loc = 0, scale = 1, shape = 0, lower.tail = TRUE)
dgev(x, loc = 0, scale = 1, shape = 0, log = FALSE)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations.
`loc`	vector of the location parameters.
`scale`	vector of the scale parameters.
`shape`	a numeric of the shape parameter.
`lower.tail`	logical; if TRUE (default), probabilities are $\Pr[X \leq x]$, otherwise, $\Pr[X > x]$.
`log`	logical; if TRUE, probabilities p are given as log(p).

Value

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

The GEV distribution function for loc = u, scale = σ and shape = ξ is

$$G(x) = \exp\left[-\left\{1 + \xi \frac{x-u}{\sigma}\right\}^{-1/\xi}\right]$$

for $1 + \xi(x-u)/\sigma > 0$ and $x > u$, where $\sigma > 0$. If $\xi = 0$, the distribution is defined by continuity corresponding to the Gumbel distribution.

Examples

```
dgev(0.1)
rgev(100, 1, 2, 0.2)
qgev(seq(0.1, 0.9, 0.1), 1, 0.5, -0.2)
pgev(12.6, 2, 0.5, 0.1)
```

`Generalized Pareto Distribution`
The Generalized Pareto Distribution

Description

Density, distribution function, quantile function and random generation for the GP distribution with location equal to 'loc', scale equal to 'scale' and shape equal to 'shape'.

Usage

```
rgpd(n, loc = 0, scale = 1, shape = 0)
pgpd(q, loc = 0, scale = 1, shape = 0, lower.tail = TRUE, lambda = 0)
qgpd(p, loc = 0, scale = 1, shape = 0, lower.tail = TRUE, lambda = 0)
dgpd(x, loc = 0, scale = 1, shape = 0, log = FALSE)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations.
`loc`	vector of the location parameters.
`scale`	vector of the scale parameters.
`shape`	a numeric of the shape parameter.
`lower.tail`	logical; if TRUE (default), probabilities are $\Pr[X \leq x]$, otherwise, $\Pr[X > x]$.
`log`	logical; if TRUE, probabilities p are given as log(p).
`lambda`	a single probability - see the "value" section.

Value

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

The GP distribution function for loc = u, scale = σ and shape = ξ is

$$G(x) = 1 - \left[1 + \frac{\xi(x-u)}{\sigma}\right]^{-1/\xi}$$

for $1 + \xi(x-u)/\sigma > 0$ and $x > u$, where $\sigma > 0$. If $\xi = 0$, the distribution is defined by continuity corresponding to the exponential distribution.

By definition, the GP distribution models exceedances above a threshold. In particular, the G function is a suited candidate to model

$$\Pr\left[X \geq x | X > u\right] = 1 - G(x)$$

for u large enough.

However, it may be usefull to model the "non conditional" quantiles, that is the ones related to $\Pr[X \leq x]$. Using the conditional probability definition, one have :

$$\Pr\left[X \geq x\right] = (1-\lambda)\left(1 + \xi\frac{x-u}{\sigma}\right)^{-1/\xi}$$

where $\lambda = \Pr[X \leq u]$.

When $\lambda = 0$, the "conditional" distribution is equivalent to the "non conditional" distribution.

Examples

```
dgpd(0.1)
rgpd(100, 1, 2, 0.2)
qgpd(seq(0.1, 0.9, 0.1), 1, 0.5, -0.2)
pgpd(12.6, 2, 0.5, 0.1)
##for non conditional quantiles
qgpd(seq(0.9, 0.99, 0.01), 1, 0.5, -0.2, lambda = 0.9)
pgpd(2.6, 2, 2.5, 0.25, lambda = 0.5)
```

gev2frech	*Transforms GEV data to unit Frechet ones and vice versa*

Description

Transforms GEV data to unit Frechet ones and vice versa

Usage

```
gev2frech(x, loc, scale, shape, emp = FALSE)
frech2gev(x, loc, scale, shape)
```

Arguments

`x` The data to be transformed to unit Frechet or ordinary GEV margins

`loc, scale, shape` The location, scale and shape parameters of the GEV.

`emp` Logical. If `TRUE`, data are transformed to unit Frechet margins using the empirical CDF.

Details

If Y is a random variable with a GEV distribution with location μ, scale σ and shape ξ. Then,

$$Z = \left[1 + \xi \frac{Y - \mu}{\sigma}\right]_+^{1/\xi}$$

is unit Frechet distributed - where $x_+ = max(0, x)$.

If Z is a unit Frechet random variable. Then,

$$Y = \mu + \sigma \frac{Z_+^\xi - 1}{\xi}$$

is unit GEV distributed with location, scale and shape parameters equal to μ, σ and ξ respectively.

Value

Returns a numeric vector with the same length of `x`

Author(s)

Mathieu Ribatet

Examples

```
x <- c(2.2975896, 1.6448808, 1.3323833, -0.4464904, 2.2737603,
    -0.2581876, 9.5184398, -0.5899699, 0.4974283, -0.8152157)
y <- gev2frech(x, 1, 2, .2)
y
frech2gev(y, 1, 2, .2)
```

`lmadogram` *Computes the lambda-madogram*

Description

Computes the lambda-madogram for max-stable processes.

Usage

```
lmadogram(data, coord, n.bins, xlab, ylab, zlab, n.lambda = 11, marge =
"emp", col = terrain.colors(50, alpha = 0.5), theta = 90, phi = 20,
border = NA, ...)
```

Arguments

`data`	A matrix representing the data. Each column corresponds to one location.
`coord`	A matrix that gives the coordinates of each location. Each row corresponds to one location.
`n.bins`	The number of bins to be used. If missing, pairwise lambda-madogram estimates will be computed.
`xlab,ylab,zlab`	The x-axis, y-axis and z-axis labels. May be missing.
`n.lambda`	Integer giving the number of lambda values.
`marge`	Character string. If 'emp', probabilities of non exceedances are estimated using the empirical CDF. If 'mle' (default), maximum likelihood estimates are used.
`col`	The colors used to emphasize the gradient of the lambda-madogram.
`theta,phi,border`	Options to be passed to the `persp` function.
`...`	Additional options to be passed to the `persp` function.

Details

Let $Z(x)$ be a stationary process. The λ-madogram is defined as follows:

$$\nu_\lambda(h) = \frac{1}{2}\mathrm{E}\left[|F^\lambda(Z(x+h)) - F^{1-\lambda}(Z(x))|\right]$$

Value

A graphic and (invisibly) a matrix with the lag distances, the λ-madogram estimate.

Author(s)

Mathieu Ribatet

References

Naveau, P., Guillou, A., Cooley, D. and Diebolt, J. (2009) Modelling Pairwise Dependence of Maxima in Space. To appear in *Biometrika*.

See Also

madogram, fmadogram

Examples

```
n.site <- 50
locations <- matrix(runif(2*n.site, 0, 10), ncol = 2)
colnames(locations) <- c("lon", "lat")

##Simulate a max-stable process - with unit Frechet margins
data <- rmaxstab(40, locations, cov.mod = "whitmat", sill = 1, range = 1,
smooth = 2)

##Compute the lambda-madogram
lmadogram(data, locations, n.bins = 80)
```

---

logLik — *Extracts Log-Likelihood*

Description

Extract the pairwise log-likelihood for objects of class "maxstab"

Usage

```
## S3 method for class 'maxstab':
logLik(object, ...)
```

Arguments

`object`	An object of class "maxstab". Most often this will be the output of the fitmaxstab function.
`...`	Other arguments to be passed to the logLik function.

Value

Standard `logLik` object (see logLik) except that this is the pairwise log-likelihood!

Author(s)

Mathieu Ribatet

See Also

logLik

Examples

```
##Define the coordinates of each location
n.site <- 30
locations <- matrix(5 + runif(2*n.site, 0, 10), ncol = 2)

##Simulate a max-stable process - with unit Frechet margins
data <- rmaxstab(30, locations, cov.mod = "whitmat", sill = 1, range = 3,
smooth = 0.5)
fit <- fitmaxstab(data, locations, "whitmat")
logLik(fit)
```

---

lsmaxstab — *Estimates the spatial dependence parameter of a max-stable process by minimizing least squares.*

Description

Estimates the spatial dependence parameter of a max-stable process by minimizing least squares.

Usage

```
lsmaxstab(data, coord, cov.mod = "gauss", marge = "emp", control =
list(), iso = FALSE, ..., weighted = TRUE)
```

Arguments

`data`	A matrix representing the data. Each column corresponds to one location.
`coord`	A matrix that gives the coordinates of each location. Each row corresponds to one location.
`cov.mod`	Character string specifying the max-stable process considered. Must be one of "gauss" (Smith's model), "whitmat", "cauchy", "powexp", "bessel", "caugen" for the Schlather model with the corresponding correlation function.
`marge`	Character string specifying how margins are transformed to unit Frechet. Must be one of "emp", "frech" or "mle" - see function `fitextcoeff`.
`control`	The control arguments to be passed to the `optim` function.
`iso`	Logical. If `TRUE`, isotropy is supposed. Otherwise (default), anisotropy is allowed. Currently this is only useful for the Smith model.
`...`	Optional arguments.
`weighted`	Logical. Should weighted least squares be used? See Details.

Details

The fitting procedure is based on weighted least squares. More precisely, the fitting criteria is to minimize:

$$\sum_{i,j} \left(\frac{\tilde{\theta}_{i,j} - \hat{\theta}_{i,j}}{s_{i,j}} \right)^2$$

where $\tilde{\theta}_{i,j}$ is a non parametric estimate of the extremal coefficient related to location `i` and `j`, $\hat{\theta}_{i,j}$ is the fitted extremal coefficient derived from the maxstable model considered and $s_{i,j}$ are the standard errors related to the estimates $\tilde{\theta}_{i,j}$.

Value

An object of class maxstab.

Author(s)

Mathieu Ribatet

References

Smith, R. L. (1990) Max-stable processes and spatial extremes. Unpublished manuscript.

See Also

`fitcovariance`, `fitmaxstab`, `fitextcoeff`

Examples

```
n.site <- 50
n.obs <- 100
locations <- matrix(runif(2*n.site, 0, 40), ncol = 2)
colnames(locations) <- c("lon", "lat")

## Simulate a max-stable process - with unit Frechet margins
data <- rmaxstab(50, locations, cov.mod = "gauss", cov11 = 200, cov12 =
0, cov22 = 200)

lsmaxstab(data, locations, "gauss")

##Force an isotropic model and do not use weights
lsmaxstab(data, locations, "gauss", iso = TRUE, weighted = FALSE)
```

madogram *Computes madograms*

Description

Computes the madogram for max-stable processes.

Usage

```
madogram(data, coord, fitted, n.bins, gev.param = c(0, 1, 0), which =
c("mado", "ext"), xlab, ylab, col = c(1, 2), angles = NULL, marge =
"emp", add = FALSE, ...)
```

Arguments

`data`	A matrix representing the data. Each column corresponds to one location.
`coord`	A matrix that gives the coordinates of each location. Each row corresponds to one location.
`fitted`	An object of class maxstab - usually the output of the `fitmaxstab` function. May be missing.
`n.bins`	The number of bins to be used. If missing, pairwise madogram estimates will be computed.
`gev.param`	Numeric vector of length 3 specifying the location, scale and shape parameters for the GEV.
`which`	A character vector of maximum size 2. It specifies if the madogram and/or the extremal coefficient functions have to be plotted.
`xlab,ylab`	The x-axis and y-axis labels. May be missing. Note that `ylab` must have the same length has `which`.
`col`	The colors used for the points and optionnaly for the fitted curve.
`angles`	A numeric vector. A partition of the interval $(-\pi, \pi)$ to help detecting anisotropy.
`marge`	Character string. If 'emp', the observation are first transformed to the unit Frechet scale by using the empirical CDF. If 'mle' (default), maximum likelihood estimates are used.
`add`	Logical. If `TRUE`, the plot is added to the current figure; otherwhise (default) a new plot is computed.
`...`	Additional options to be passed to the `plot` function.

Details

Let $Z(x)$ be a stationary process. The madogram is defined as follows:

$$\nu(h) = \frac{1}{2}\mathrm{E}\left[|Z(x+h) - Z(x)|\right]$$

If now $Z(x)$ is a stationary max-stable random field with GEV marginals. Provided the GEV shape parameter ξ is such that $\xi < 1$. The extremal coefficient $\theta(h)$ satisfies:

$$\theta(h) = \begin{cases} u_\beta \left(\mu + \frac{\nu(h)}{\Gamma(1-\xi)}\right), & \xi \neq 0 \\ \exp\left(\frac{\nu(h)}{\sigma}\right), & \xi = 0 \end{cases}$$

where $\Gamma(\cdot)$ is the gamma function and u_β is defined as follows:

$$u_\beta(u) = \left(1 + \xi \frac{u-\mu}{\sigma}\right)_+^{1/\xi}$$

and $\beta = (\mu, \sigma, \xi)$ i.e the vector of the GEV parameters.

Value

A graphic and (invisibly) a matrix with the lag distances, the madogram and extremal coefficient estimates.

Author(s)

Mathieu Ribatet

References

Cooley, D., Naveau, P. and Poncet, P. (2006) Variograms for spatial max-stable random fields. *Dependence in Probability and Statistics*, 373–390.

See Also

fmadogram, lmadogram

Examples

```
n.site <- 15
locations <- matrix(runif(2*n.site, 0, 10), ncol = 2)
colnames(locations) <- c("lon", "lat")

##Simulate a max-stable process - with unit Frechet margins
data <- rmaxstab(40, locations, cov.mod = "whitmat", sill = 1, range = 1,
smooth = 2)

##Compute the madogram
madogram(data, locations)

##Compare the madogram with a fitted max-stable model
fitted <- fitmaxstab(data, locations, "whitmat", sill = 1)
madogram(fitted = fitted, which = "ext")
```

map *Produces a 2D map from a fitted max-stable process*

Description

Produces a 2D map from a fitted max-stable process.

Usage

```
map(fitted, x, y, covariates = NULL, param = "quant", ret.per = 100, col
= terrain.colors(64), plot.contour = TRUE, ...)
```

Arguments

`fitted`	An object of class `maxstab`. Most often, it will be the output of the function `fitmaxstab`.
`x,y`	Numeric vector that gives the coordinates of the grid.
`covariates`	An array specifying the covariates at each grid point defined by `x` and `y`. If `NULL`, no covariate is needed. See the example to see how to build it.
`param`	A character string. Must be one of "loc", "scale", "shape" or "quant" for a map of the location, scale, shape parameters or for a map of a specified quantile.
`ret.per`	A numeric giving the return period for which the quantile map is plotted. It is only required if `param = "quant"`.
`col`	A list of colors such as that generated by 'rainbow', 'heat.colors', 'topo.colors', 'terrain.colors' or similar functions.
`plot.contour`	Logical. If `TRUE` (default), contour lines are added to the plot.
`...`	Several arguments to be passed to the `link{image}` function.

Value

A plot. Additionally, a list with the details for plotting the map is returned invisibly.

Author(s)

Mathieu Ribatet

See Also

`condmap`, `filled.contour`, `heatmap`, `heat.colors`, `topo.colors`, `terrain.colors`, `rainbow`

Examples

```
##We run an artifical example using the volcano data set as a study
##region
dim <- dim(volcano)
n.x <- dim[1]
n.y <- dim[2]

x <- 10 * 1:n.x
y <- 10 * 1:n.y

n.site <- 15
idx.x <- sample(n.x, n.site)
idx.y <- sample(n.y, n.site)
locations <- cbind(lon = x[idx.x], lat = y[idx.y])
alt <- diag(volcano[idx.x, idx.y])

##Simulate a max-stable process - with unit Frechet margins
data <- rmaxstab(40, locations, cov.mod = "whitmat", sill = 1, range = 750,
smooth = 1)

##Now define the spatial model for the GEV parameters
param.loc <- -10 - 0.04 * locations[,1] + alt / 5
param.scale <- 5 - locations[,2] / 30 + alt / 4
param.shape <- rep(.2, n.site)

##Transform the unit Frechet margins to GEV
for (i in 1:n.site)
  data[,i] <- frech2gev(data[,i], param.loc[i], param.scale[i],
param.shape[i])

##Define a model for the GEV margins to be fitted
##shape ~ 1 stands for the GEV shape parameter is constant
##over the region
loc.form <- loc ~ lon + alt
scale.form <- scale ~ lat + alt
shape.form <- shape ~ 1

##  1- Fit a max-stable process
schlather <- fitmaxstab(data, locations, "whitmat", loc.form, scale.form,
                        shape.form, marg.cov = cbind(alt = alt), sill =
1, std.err.type = "none")

##  2- Produce a map of the pointwise 50-year return level

##Here we have only one covariate i.e. alt
n.cov <- 1
covariates <- array(volcano, dim = c(n.x, n.y, n.cov), dimnames =
list(NULL, NULL, "alt"))

par(mfrow = c(1,2))
image(x, y, volcano, col = terrain.colors(64), main = "Elevation map")
map(schlather, x, y, covariates, ret.per = 50, plot.contour = FALSE,
```

```
main = "50-year return level")
```

`margin fits` *Fits univariate extreme value distributions to data*

Description

These functions fit the generalised extreme value and generalised Pareto distribution to data using maximum likelihood.

Usage

```
gevmle(x, ..., method = "Nelder")
gpdmle(x, threshold, ..., method = "Nelder")
```

Arguments

`x`	Numeric vector of observations
`...`	Optional arguments to be passed to the `optim` function.
`threshold`	Numeric. The threshold value.
`method`	The numerical optimisation method to be used.

Details

These two functions are "extremely light" functions to fit the GEV/GPD. These functions are mainly useful to compute starting values for the Schlather and Smith model - see `fitmaxstab`.

If more refined (univariate) analysis have to be performed, users should use more specialised packages - e.g. POT, evd, ismev, ...

Value

A vector for the estimated parameters of the GEV/GPD.

Author(s)

Mathieu Ribatet

Examples

```
## 1 - GEV fit
x <- rep(NA, 100)
for (i in 1:100)
  x[i] <- max(rnorm(365))

gevmle(x)

## 2- GPD fit
x <- rnorm(10000)
```

```
##we need to fix a threshold
u <- quantile(x, 0.99)
gpdmle(x, u)
```

---

modeldef — *Define a model for the spatial behaviour of the GEV parameters*

Description

This function defines the model for the spatial behaviour of the GEV parameter.

Usage

```
modeldef(data, formula)
```

Arguments

`data`	A matrix representing the data. Each column corresponds to one location.
`formula`	A R formula. See details for further details.

Value

need to be documented

Author(s)

Mathieu Ribatet

See Also

`formula`

Examples

```
## 1- A design matrix from a classical linear model
n.site <- 5
coord <- matrix(rnorm(2*n.site, sd = sqrt(.2)), ncol = 2)
colnames(coord) <- c("lon", "lat")
loc.form <- loc ~ lat + I(lon^2)
modeldef(coord, loc.form)

## 2- A design and penalization matrix from a penalized smoothin spline
x <- sort(runif(10, -2, 10))
n.knots <- 3
knots <- quantile(x, prob = 1:n.knots / (n.knots + 2))
modeldef(x, y ~ rb(x, knots = knots, degree = 3, penalty = 1))
```

predict.maxstab *Prediction of the max-stable marginal parameters*

Description

This function predicts the marginal GEV parameters from a fitted max-stable process.

Usage

```
## S3 method for class 'maxstab':
predict(object, newdata, ret.per = NULL, std.err =
TRUE, ...)
```

Arguments

object	An object of class “maxstab”. Most often, it will be the output of the function `fitmaxstab`.
newdata	An optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used.
ret.per	Numeric vector giving the return periods for which return levels are computed. If `NULL` (default), no return levels are computed.
std.err	Logical. If `TRUE` (default), standard erros will be computed if possible.
...	further arguments passed to or from other methods.

Value

’predict.maxstab’ produces a vector of predictions or a matrix of predictions.

Author(s)

Mathieu Ribatet

See Also

`predict`

Examples

```
## 1- Simulate a max-stable random field
n.site <- 35
locations <- matrix(runif(2*n.site, 0, 10), ncol = 2)
colnames(locations) <- c("lon", "lat")

data <- rmaxstab(50, locations, cov.mod = "whitmat", sill = 1, range = 30,
smooth = 0.5)

## 2- Transformation to non unit Frechet margins
```

```
param.loc <- -10 + 2 * locations[,2]
param.scale <- 5 + 2 * locations[,1]
param.shape <- rep(0.2, n.site)

for (i in 1:n.site)
  data[,i] <- frech2gev(data[,i], param.loc[i], param.scale[i],
param.shape[i])

## 3- Fit a max-stable process with the following model for
##    the GEV parameters
form.loc <- loc ~ lat
form.scale <- scale ~ lon
form.shape <- shape ~ 1

schlather <- fitmaxstab(data, locations, "whitmat", loc.form = form.loc,
                        scale.form = form.scale, shape.form =
                        form.shape)

## 4- GEV parameters estimates at each locations or at ungauged locations
predict(schlather)
ungauged <- data.frame(lon = runif(10, 0, 10), lat = runif(10, 0, 10))
predict(schlather, ungauged)
```

`predict.pspline` *Prediction of smoothing spline with radial basis functions*

Description

This function predicts the response from a fitted smoothing spline.

Usage

```
## S3 method for class 'pspline':
predict(object, new.data, ...)
```

Arguments

`object`	An object of class “pspline”. Most often, it will be the output of the function `rbpspline`.
`new.data`	An optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used.
`...`	further arguments passed to or from other methods.

Value

’predict.pspline’ produces a vector of predictions or a matrix of predictions.

Author(s)

Mathieu Ribatet

See Also

predict

Examples

```
## 1- Define a function to approximate
fun <- function(x)
  sin(3 * pi * x)

## 2- Compute the response from fun - and adding a noise
x <- seq(0, 1, length = 200)
y <- fun(x) + rnorm(200, 0, sqrt(0.4))

## 2- Fit a penalized smoothing spline
n.knots <- 30
knots <- quantile(x, prob = 1:n.knots / (n.knots + 2))
fitted <- rbpspline(y, x, knots, degree = 3)

## 3- Prediction from the fitted spline
plot(x, y, pch = 2, col = "red")
plot(fun, from = 0, to = 1, add = TRUE)
pred <- predict(fitted)
lines(pred[,1], pred[,2], col = "blue", pch = 3)
```

---

predict.spatgev *Prediction of the GEV parameters*

Description

This function predicts the marginal GEV parameters from a fitted ”spatial GEV” model.

Usage

```
## S3 method for class 'spatgev':
predict(object, newdata, ret.per = NULL, ...)
```

Arguments

object	An object of class spatgev”. Most often, it will be the output of the function fitspatgev.
newdata	An optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used.
ret.per	Numeric vector giving the return periods for which return levels are computed. If NULL (default), no return levels are computed.
...	further arguments passed to or from other methods.

Value

'predict.spatgev' produces a vector of predictions or a matrix of predictions.

Author(s)

Mathieu Ribatet

See Also

`predict`

Examples

```
## 1- Simulate a max-stable random field
n.site <- 35
locations <- matrix(runif(2*n.site, 0, 10), ncol = 2)
colnames(locations) <- c("lon", "lat")

data <- rmaxstab(50, locations, cov.mod = "whitmat", sill = 1, range = 3,
smooth = 0.5)
## 2- Transformation to non unit Frechet margins
param.loc <- -10 + 2 * locations[,2]
param.scale <- 5 + 2 * locations[,1]
param.shape <- rep(0.2, n.site)

for (i in 1:n.site)
  data[,i] <- frech2gev(data[,i], param.loc[i], param.scale[i],
param.shape[i])

## 3- Fit a ''spatial GEV'' mdoel to data with the following models for
##    the GEV parameters
form.loc <- loc ~ lat
form.scale <- scale ~ lon
form.shape <- shape ~ 1

fitted <- fitspatgev(data, locations, form.loc, form.scale, form.shape)

## 4- GEV parameters estimates at each locations or at ungauged locations
predict(fitted)
ungauged <- data.frame(lon = runif(10, 0, 10), lat = runif(10, 0, 10))
predict(fitted, ungauged)
```

`print.maxstab` *Printing objects of class "maxstab"*

Description

A method for printing object of class "maxstab".

Usage

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

Arguments

x	An object of class "maxstab". Most often, x is the output of the fitmaxstab function.
digits	The number of digits to be printed.
...	Other options to be passed to the print function.

Value

Print several information on screen.

Author(s)

Mathieu Ribatet

Examples

```
##Define the coordinates of each location
n.site <- 30
locations <- matrix(5 + rnorm(2*n.site, sd = sqrt(2)), ncol = 2)

##Simulate a max-stable process - with unit Frechet margins
data <- rmaxstab(30, locations, cov.mod = "whitmat", sill = 1, range = 3,
smooth = 0.5)

fit <- fitmaxstab(data, locations, "whitmat")
fit
```

---

print.pspline *Printing objects of class "pspline"*

Description

A method for printing object of class "pspline".

Usage

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

Arguments

`x`	An object of class “pspline”. Most often, `x` is the output of the `rbpspline` function.
`...`	Other options to be passed to the `print` function.

Value

Print several information on screen.

Author(s)

Mathieu Ribatet

Examples

```
n <- 200
x <- runif(n)
fun <- function(x) sin(3 * pi * x)
y <- fun(x) + rnorm(n, 0, sqrt(0.4))
knots <- quantile(x, prob = 1:31 / 32)
fitted <- rbpspline(y, x, knots = knots, degree = 3)
fitted
```

`print.spatgev` *Printing objects of class ”spatgev”*

Description

A method for printing object of class ”spatgev”.

Usage

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

Arguments

`x`	An object of class ”spatgev”. Most often, `x` is the output of the `fitspatgev` function.
`digits`	The number of digits to be printed.
`...`	Other options to be passed to the `print` function.

Value

Print several information on screen.

Author(s)

Mathieu Ribatet

Examples

```
## 1- Simulate a max-stable random field
n.site <- 35
locations <- matrix(runif(2*n.site, 0, 10), ncol = 2)
colnames(locations) <- c("lon", "lat")

data <- rmaxstab(50, locations, cov.mod = "whitmat", sill = 1, range = 3,
smooth = 0.5)

## 2- Transformation to non unit Frechet margins
param.loc <- -10 + 2 * locations[,2]
param.scale <- 5 + 2 * locations[,1]
param.shape <- rep(0.2, n.site)

for (i in 1:n.site)
  data[,i] <- frech2gev(data[,i], param.loc[i], param.scale[i],
param.shape[i])

## 3- Fit a ''spatial GEV'' mdoel to data with the following models for
##    the GEV parameters
form.loc <- loc ~ lat
form.scale <- scale ~ lon
form.shape <- shape ~ 1

fitspatgev(data, locations, form.loc, form.scale, form.shape)
```

`profile` *Method for profiling fitted max-stable objects*

Description

Computes profile traces for fitted max-stable models.

Usage

```
## S3 method for class 'maxstab':
profile(fitted, param, range, n = 10, plot = TRUE,
conf = 0.90, method = "RJ", square = "chol", ...)
```

Arguments

`fitted`	An object of class “maxstab”. Most often, it will be the output of the function `fitmaxstab`.
`param`	A character string giving the model parameter that are to be profiled.

`range`	The range for the profiled model parameter that must be explored.
`n`	Integer. The number of profiled model parameter that must be considered.
`plot`	Logical. If `TRUE` (default), the profile trace is plotted.
`conf`	Numeric giving the confidence interval level.
`method`	Character string. Must be one of "CB", "RJ" or "none" for the Chandler and Bate or the Rotnitzky and Jewell approaches respectively. The "none" method simply plots the profile of the log-composite likelihood. See details.
`square`	The choice for the matrix square root. This is only useful for the 'CB' method. Must be one of 'chol' (Cholesky) or 'svd' (Singular Value Decomposition).
`...`	Extra options that must be passed to the `plot` function.

Details

The Rotnitzky and Jewell approach consists in adjusting the distribution of the likelihood ratio statistics - which under misspecification is no longer χ^2 distributed.

The Chandler and Bate approach adjusts the composite likelihood itself is such a way that the usual asymptotic χ^2 null distribution is preserved. Note that in the current code, we use the singular value decomposition for the computation of matrix square roots to preserve asymmetry in the profile composite likelihood.

Value

A matrix. The first column corresponds to the values for which the profiled model parameter is fixed. The second column gives the value of the pairwise log-likelihood. The remaining columns contain the constrained maximum likelihood estimates for the remaining model parameters.

Warnings

This function can be **really** time consuming!

Author(s)

Mathieu Ribatet

References

Chandler, R. E. and Bate, S. (2007) Inference for clustered data using the independence loglikelihood *Biometrika*, **94**, 167–183.

Rotnitzky, A. and Jewell, N. (1990) Hypothesis testing of regression parameters in semiparametric generalized linear models for cluster correlated data. *Biometrika* **77**, 485–97.

Examples

```
## Not run:
##Define the coordinates of each location
n.site <- 30
locations <- matrix(rnorm(2*n.site, sd = sqrt(.2)), ncol = 2)
colnames(locations) <- c("lon", "lat")
```

```
##Simulate a max-stable process - with unit Frechet margins
data <- rmaxstab(50, locations, cov.mod = "gauss", cov11 = 100, cov12 =
25, cov22 = 220)

##Fit a max-stable process
##  1- using the Smith's model
fitted <- fitmaxstab(data, locations, "gauss", fit.marge = FALSE)

##Plot the profile pairwise log-likelihood for the ''cov11'' parameter
profile(fitted, "cov11", range = c(20, 180))

## End(Not run)
```

`profile2d` *Method for profiling (in 2d) fitted max-stable objects*

Description

Computes profile surfaces for fitted max-stable models.

Usage

```
## S3 method for class 'maxstab':
profile2d(fitted, params, ranges, n = 10, plot = TRUE,
...)
```

Arguments

`fitted`	An object of class "maxstab". Most often, it will be the output of the function `fitmaxstab`.
`params`	A character vector giving the two model parameters that are to be profiled.
`ranges`	A matrix corresponding to the ranges for the profiled model parameters that must be explored. Each row corresponds to one model parameter range.
`n`	Integer. The number of profiled model parameter that must be considered.
`plot`	Logical. If `TRUE` (default), the profile surface is plotted.
`...`	Extra options that must be passed to the `plot` function.

Value

A list with two arguments: `coord` and `llik`. `coord` is a matrix representing the grid where the profiled model parameters are fixed. `llik` the corresponding pairwise log-likelihood.

Warnings

This function can be **really** time consuming!

Author(s)

Mathieu Ribatet

Examples

```
## Not run:
##Define the coordinates of each location
n.site <- 30
locations <- matrix(rnorm(2*n.site, sd = sqrt(.2)), ncol = 2)
colnames(locations) <- c("lon", "lat")

##Simulate a max-stable process - with unit Frechet margins
data <- rmaxstab(30, locations, cov.mod = "whitmat", sill = 1, range = 30,
smooth = 0.5)

##Now define the spatial model for the GEV parameters
param.loc <- -10 + 2 * locations[,2]
param.scale <- 5 + 2 * locations[,1] + locations[,2]^2
param.shape <- rep(0.2, n.site)

##Transform the unit Frechet margins to GEV
for (i in 1:n.site)
  data[,i] <- frech2gev(data[,i], param.loc[i], param.scale[i],
param.shape[i])

##Define a model for the GEV margins to be fitted
##shape ~ 1 stands for the GEV shape parameter is constant
##over the region
loc.form <- loc ~ lat
scale.form <- scale ~ lon + (lat^2)
shape.form <- shape ~ 1

##Fit a max-stable process
##  1- using the Schlather representation
fitted <- fitmaxstab(data, locations, "whitmat", loc.form, scale.form,
                     shape.form)

##Plot the profile pairwise log-likelihood for the smooth parameter
ranges <- rbind(c(9,11), c(.3, .8))
profile2d(fitted, c("range", "smooth"), ranges = ranges)

## End(Not run)
```

---

qqextcoeff — *QQ-plot for the extremal coefficient*

Description

This function compares the extremal coefficients estimated from a fitted max-stable process to the ones estimated semi-parametrically.

Usage

```
qqextcoeff(fitted, estim = "ST", marge = "emp", xlab = "Semi-Empirical",
ylab = "Model", ...)
```

Arguments

`fitted`	An object of class `maxstab`. Most often, this will be the output of `fitmaxstab`, `fitcovmat` or `fitcovariance`.
`estim,marge`	The `estim` and `marge` options to be passed to the `fitextcoeff` function.
`xlab,ylab`	The x and y-axis labels.
`...`	Optional arguments to be passed to the `plot` function.

Value

A QQ-plot.

Author(s)

Mathieu Ribatet

References

Schlather, M. (2002) Models for Stationary Max-Stable Random Fields. *Extremes* **5**:1, 33–44.

Schlather, M. and Tawn, J. A. (2003) A dependence measure for multivariate and spatial extreme values: Properties and inference. Biometrika **90**(1):139–156.

Smith, R. L. (1990) Max-stable processes and spatial extremes. Unpublished manuscript.

See Also

`fitmaxstab`, `fitextcoeff`

Examples

```
##Define the coordinate of each location
n.site <- 30
locations <- matrix(runif(2*n.site, 0, 10), ncol = 2)
colnames(locations) <- c("lon", "lat")

##Simulate a max-stable process - with unit Frechet margins
data <- rmaxstab(50, locations, cov.mod = "gauss", cov11 = 10, cov12 =
5, cov22 = 22)

fitted <- fitmaxstab(data, locations, "gauss")
qqextcoeff(fitted)
```

`qqgev` *QQ-plot for the GEV parameters*

Description

This function compares the GEV parameters estimated separately for each location to the ones estimated from a fitted spatial model.

Usage

```
qqgev(fitted, xlab, ylab, ...)
```

Arguments

`fitted`	An object of class `maxstab` or `spatgev`. Most often, this will be the output of `fitmaxstab`, `fitcovmat`, `fitcovariance` or `fitspatgev`.
`xlab,ylab`	The x and y-axis labels. May be missing.
`...`	Optional arguments to be passed to the `plot` function.

Value

A QQ-plot.

Author(s)

Mathieu Ribatet

References

Schlather, M. (2002) Models for Stationary Max-Stable Random Fields. *Extremes* **5**:1, 33–44.

Schlather, M. and Tawn, J. A. (2003) A dependence measure for multivariate and spatial extreme values: Properties and inference. Biometrika **90**(1):139–156.

Smith, R. L. (1990) Max-stable processes and spatial extremes. Unpublished manuscript.

See Also

`qqextcoeff`

Examples

```
##Define the coordinate of each location
n.site <- 30
locations <- matrix(runif(2*n.site, 0, 10), ncol = 2)
colnames(locations) <- c("lon", "lat")

##Simulate a max-stable process - with unit Frechet margins
data <- rmaxstab(50, locations, cov.mod = "gauss", cov11 = 100, cov12 =
```

```
25, cov22 = 220)

##Now define the spatial model for the GEV parameters
param.loc <- -10 + 2 * locations[,2]
param.scale <- 5 + 2 * locations[,1] + locations[,2]^2
param.shape <- rep(0.2, n.site)

##Transform the unit Frechet margins to GEV
for (i in 1:n.site)
  data[,i] <- frech2gev(data[,i], param.loc[i], param.scale[i], param.shape[i])

##Define a model for the GEV margins to be fitted
##shape ~ 1 stands for the GEV shape parameter is constant
##over the region
loc.form <- loc ~ lat
scale.form <- scale ~ lon + I(lat^2)
shape.form <- shape ~ 1

fitted <- fitspatgev(data, locations, loc.form = loc.form, scale.form =
scale.form, shape.form = shape.form)

qqgev(fitted)
```

`rainfall` *Summer annual maxima precipitation in Switzerland*

Description

Maximum daily rainfall amounts over the years 1962–2008 occuring during June–August at 79 sites in Switzerland.

Usage

```
data(rainfall)
```

Format

This data set contains two R objects: 'rain' and 'coord'. 'rain' is a 47 by 79 matrix giving the amount of rain in centimeters, each column correspond to one locations. 'coord' is a 79 by 3 matrix giving the longitude, latitude and the elevation for each station, all of them being in meters.

Author(s)

Mathieu Ribatet

Examples

```
data(rainfall)
op <- par(mfrow = c(1,2),pty = "s", mar = c(0,0,0,0))
swiss(city = TRUE)
idx.site <- c(1, 10, 20)
points(coord[-idx.site,])
points(coord[idx.site,], pch = 15, col = 2:4)

par(mar = c(2,4,0,0))
plot(1962:2008, rain[,1], type = "b", xlab = "Year", ylab =
"Precipitation (cm)", ylim = c(0, 120), col = 2)
lines(1962:2008, rain[,10], col = 3, type = "b")
lines(1962:2008, rain[,20], col = 4, type = "b")
par(op)
```

`rb` *Creates a model using penalized smoothing splines*

Description

Creates a model using penalized smoothing splines using radial basis functions

Usage

```
rb(..., knots, degree, penalty)
```

Arguments

`...`	The explicative variables for which the spline is based on.
`knots`	The coordinates of knots. See section details.
`degree`	Numeric. The degree of the spline.
`penalty`	Numeric. The penalty coefficient.

Details

If one explicative variable is given in "...", the `knots` should be a numeric vector. Otherwise, `knots` should be a matrix with the same number of column and covariates.

Value

A list giving all the required information to fit a penalized smoothing spline:

`dsgn.mat`	The design matrix.
`pen.mat`	The penalization matrix.
`degree`	The degree of the smoothing spline.
`penalty`	The penalty of the smoothing spline.
`knots`	The knots of the smoothing spline.
`data`	The explicative variables (e.g. covariates).
`call`	How was the `rb` function was called?

Warning

This function is not supposed to be called directly. `rb` is supposed to be embedded in a R formula.

Author(s)

Mathieu Ribatet

See Also

`fitmaxstab`

Examples

```
n.site <- 30
locations <- matrix(runif(2*n.site, 0, 10), ncol = 2)
colnames(locations) <- c("lon", "lat")
knots <- quantile(locations[,2], 1:5/6)

form <- y ~ rb(lat, knots = knots, degree = 3, penalty = .5)
```

---

rbpspline *Fits a penalized spline with radial basis functions to data*

Description

Fits a penalized spline with radial basis functions to data.

Usage

```
rbpspline(y, x, knots, degree, penalty = "gcv", ...)
```

Arguments

`y`	The response vector.
`x`	A vector/matrix giving the values of the predictor variable(s). If `x` is a matrix, each row corresponds to one observation.
`knots`	A vector givint the coordinates of the knots.
`degree`	The degree of the penalized smoothing spline.
`penalty`	A numeric giving the penalty coefficient for the penalization term. Alternatively, it could be either 'cv' or 'gcv' to choose the `penalty` using the (generalized) cross-validation criterion.
`...`	Additional options to be passed to the `cv` or `gcv` function.

Details

The penalized spline with radial basis is defined by:

$$f(x) = \beta_0 + \beta_1 x + \ldots + \beta_{m-1} x^{m-1} + \sum_{k=0}^{K-1} \beta_{m+k} ||x - \kappa_k||^{2m-1}$$

where β_i are the coefficients to be estimated, κ_i are the coordinates of the i-th knot and $m = \frac{d+1}{2}$ where d corresponds to the `degree` of the spline.

The fitting criterion is to minimize

$$||y - X\beta||^2 + \lambda^{2m-1} \beta^T K \beta$$

where λ is the penalty coefficient and K the penalty matrix.

Value

An object of class `pspline`.

Author(s)

Mathieu Ribatet

References

Ruppert, D. Wand, M.P. and Carrol, R.J. (2003) *Semiparametric Regression* Cambridge Series in Statistical and Probabilistic Mathematics.

See Also

`cv`, `gcv`

Examples

```
n <- 200
x <- runif(n)
fun <- function(x) sin(3 * pi * x)
y <- fun(x) + rnorm(n, 0, sqrt(0.4))
knots <- quantile(x, prob = 1:(n/4) / (n/4 + 1))
fitted <- rbpspline(y, x, knots = knots, degree = 3)
fitted

plot(x, y)
lines(fitted, col = 2)
```

rgp *Gaussian Random Fields Simulation*

Description

This functions generates gaussian random fields.

Usage

```
rgp(n, coord, cov.mod = "powexp", mean = 0, nugget = 0, sill = 1, range
= 1, smooth = 1, grid = FALSE, control = list())
```

Arguments

`n`	Integer. The number of replications.
`coord`	The locations coordinates for which the gaussian process is observed.
`cov.mod`	Character string. The covariance model used. Must be one of "whitmat", "cauchy", "powexp" of "cauchy". See the `covariance` function.
`mean`	Numeric. The mean of the gaussian random field.
`nugget`	Numeric. The nugget of the gaussian random field.
`sill`	Numeric. The sill parameter in the covariance function.
`range`	Numeric. The range parameter in the covariance function.
`smooth`	Numeric. The smooth parameter in the covariance function.
`grid`	Logical. Does `coord` defines a grid?
`control`	A named list with arguments 'nlines' (number of lines of the TBM simulation) and 'method' the name of the simulation method - must be one of 'exact' or 'tbm'. If 'method' is `NULL` (default), the function tries to find the most appropriate simulation technique. If 'nlines' is `NULL` it is set to 1000.

Value

A matrix or an array containing the random field replicates.

Author(s)

Mathieu Ribatet

See Also

`link{rmaxstab}`

Examples

```
x <- y <- seq(0, 20, length = 100)
coord <- cbind(x, y)
gp <- rgp(1, coord, cov.mod = "whitmat", grid = TRUE)
filled.contour(x, y, gp[,,1], color.palette = terrain.colors)
```

`rmaxstab` *Simulation of Max-Stable Random Fields*

Description

This functions generates realisation from a max-stable random field.

Usage

```
rmaxstab(n, coord, cov.mod = "gauss", grid = FALSE, control =
list(), ...)
```

Arguments

`n`	Integer. The number of observations.
`coord`	A vector or matrix that gives the coordinates of each location. Each row corresponds to one location - if any.
`cov.mod`	A character string that gives the max-stable model. This must be one of "gauss" for the Smith model or "whitmat", "cauchy", "powexp" and "bessel" for the Schlather model with the given correlation family.
`grid`	Logical. Does the coordinates represent grid points?
`control`	A named list with arguments 'nlines' (number of lines of the TBM simulation), 'method' the name of the simulation method - must be one of 'exact' or 'tbm', and 'uBound' the uniform upper bound. See details.
`...`	The parameters of the max-stable model. See details.

Details

Users must supply the parameters for the max-stable model. For the Schlather model, users should supply the "sill", "range" and "smooth" parameter values. For the Smith model, if `coord` is univariate you must specify `var`, otherwise users should supply the covariance parameters i.e. parameters with names such as `cov11`, `cov12`, ...

Here are the details for each allowed components of 'control'. If 'method' is `NULL` (default), the function tries to find the most appropriate simulation technique. If 'nlines' is `NULL`, it is set to 1000. If 'uBound' is `NULL`, it is set to reasonable values - for example 3.5 for the Schalther model.

Value

A matrix containing observations from the required max-stable model. Each column represents one stations. If `grid = TRUE`, the function returns an array of dimension nrow(coord) x nrow(coord) x n.

Author(s)

Mathieu Ribatet

References

Schlather, M. (2002) Models for Stationary Max-Stable Random Fields. *Extremes* **5**:1, 33–44.

Smith, R. L. (1990) Max-stable processes and spatial extremes. Unpublished manuscript.

See Also

`fitmaxstab`

Examples

```
## 1. Smith's model
set.seed(8)
x <- y <- seq(0, 10, length = 200)
coord <- cbind(x, x)
data <- rmaxstab(1, coord, cov11 = 9/8, cov12 = 0, cov22 = 9/8, grid = TRUE)
##We change to unit Gumbel margins for visibility
filled.contour(x, y, log(data[,,1]), color.palette = terrain.colors)

## 2. Schlather's model
coord <- matrix(runif(100, 0, 15), ncol = 2)
data <- rmaxstab(100, coord, cov.mod = "whitmat", sill = 1, range = 10,
smooth = 1)
```

`SpatialExtremes` *Analysis of Spatial Extremes*

Description

The package **SpatialExtremes** aims to provide tools for the analysis of spatial extremes. Currently, the package uses the max-stable processes framework for the modelling of spatial extremes.

Max-stable processes are the extension of the extreme value theory to random fields. Consequently, they are good candidate to the analysis of spatial extremes. The strategy used in this package is to fit max-stable processes to data using composite likelihood.

In the future, the package will allow for non-stationarity as well as other approaches to model spatial extremes; namely latent variable and copula based approaches.

A package vignette has been writen to help new users. It can be viewed, from the R console, by invoking `vignette("SpatialExtremesGuide")`.

Details

The package provides the following main tools:

1. `rgp`: simulates gaussian random fields,
2. `rmaxstab`: simulates max-stable random fields,
3. `fitspatgev`: fits a spatial GEV model to data,
4. `fitmaxstab`: fits max-stable processes to data,

5. predict: allows predictions for fitted max-stable processes,
6. map, condmap: plot a map for GEV parameter as well as return levels - or conditional return levels
7. anova, TIC: help users in model selection,
8. madogram: are (kind of) variograms devoted to extremes,
9. fitextcoeff: estimates semi-parametrically the extremal coefficient,
10. extcoeff: plots the evolution of the extremal coefficient from a fitted max-stable process,
11. rbpspline: fits a penalized spline with radial basis function,
12. gev2frech, frech2gev: transform GEV (resp. Frechet) observation to unit Frechet (resp. GEV) ones
13. gevmle, gpdmle: fit the GEV/GPD distributions to data,
14. distance: computes the distance between each pair of locations,
15. profile, profile2d: computes the profile composite likelihood,
16. covariance: computes the covariance function.

Acknowledgement

The development of the package has been financially supported by the Competence Center Environment and Sustainability (CCES) and more precisely within the EXTREMES project (http://www.cces.ethz.ch/projects/hazri/EXTREMES).

Author(s)

Mathieu Ribatet

---

swiss *Map of the Switzerland.*

Description

Plot a map of Switzerland and optionnaly some cities.

Usage

```
swiss(city = FALSE, add = FALSE, axes = FALSE, km = TRUE, xlab = "",
ylab = "", ...)
```

Arguments

city	Logical. It TRUE, some city are displayed. Default is to omit.
add	Logical. Should the map be added to an existing plot?
axes	Logical. Should the axis be displayed?
km	Logical. If TRUE (default) the longitude and latitude are expressed in kilometers. Otherwise it is in meters.
xlab,ylab	The x and y-axis labels.
...	Optional arguments to be passed to the plot function.

Value

A graphic window.

Author(s)

Dominik Schaub

Examples

```
swiss()
```

---

swissalt *Elevation in Switzerland*

Description

Data required for plotting a Switzerland map with elevation.

Usage

```
data(swissalt)
```

Format

This data set contains three R objects. 'alt.mat' is a 192 by 115 matrix giving the elevation at the grid points defined by 'lon.vec' and 'lat.vec'.

Author(s)

Mathieu Ribatet

Examples

```
data(swissalt)

layout(matrix(c(1,2), 1), width = c(3.5,1))
mar <- rep(0, 4)
op <- par(mar = mar)
breaks <- seq(0, 2000, by = 250)
image(lon.vec, lat.vec, alt.mat, col = terrain.colors(length(breaks) - 1),
      xaxt = "n", yaxt = "n", bty = "n", xlab = "", ylab = "", xlim =
c(480, 840), ylim = c(58, 300))
swiss(add = TRUE, city = TRUE)

##Heat bar
mar <- c(3, 3, 3, 4)
par(las = 1, mar = mar)

plot.new()
```

```
plot.window(xlim = c(0, 1), ylim = range(breaks), xaxs = "i",
            yaxs = "i")
rect(0, breaks[-length(breaks)], 1, breaks[-1], col = terrain.colors(length(breaks) - 1),
     border = NA)
axis(4, at = breaks[-length(breaks)])
box()
title("Elevation\n(meters)")
par(op)
```

TIC *Takeuchi's information criterion*

Description

Computes a the Takeuchi's information criterion which is equivalent to the AIC when the model is miss-specified.

Usage

```
TIC(object, ..., k = 2)
```

Arguments

`object`	An object of class `maxstab` or `spatgev`. Often, it will be the output of the `fitmaxstab` or `fitspatgev` function.
`...`	Additional objects of class `maxstab` or `spatgev` for which TIC should be computed.
`k`	Numeric. The penalty per parameter to be used. The case k = 2 (default) correspond to the classical TIC and $k = \log n$, n number of observations, is the robust version of the BIC.

Details

TIC is like AIC so that when comparing models one wants to get the lowest TIC score.

Value

Numeric.

Author(s)

Mathieu Ribatet

References

Gao, X. and Song, P. X.-K. (2009) Composite likelihood Bayesian information criteria for model selection in high dimensional data. Preprint.

Sakamoto, Y., Ishiguro, M., and Kitagawa G. (1986) Akaike Information Criterion Statistics. D. Reidel Publishing Company.

Varin, C. and Vidoni, P. (2005) A note on composite likelihood inference and model selection. Biometrika **92**(3):519–528.

See Also

fitmaxstab, AIC

Examples

```
##Define the coordinate of each location
n.site <- 50
locations <- matrix(runif(2*n.site, 0, 100), ncol = 2)
colnames(locations) <- c("lon", "lat")

##Simulate a max-stable process - with unit Frechet margins
data <- rmaxstab(40, locations, cov.mod = "whitmat", sill = 0.8, range =
30, smooth = 0.5)

M0 <- fitmaxstab(data, locations, "powexp", std.err.type = "score",
                 fit.marge = FALSE)
M1 <- fitmaxstab(data, locations, "cauchy", std.err.type = "score",
                 fit.marge = FALSE)

TIC(M0, M1)
TIC(M0, M1, k = log(nrow(data)))
```

---

vdc *Van der Corput Sequence*

Description

This function generates the three dimensional Van der Corput sequence on the half unit Sphere of R^3 - eventually randomly rotated.

Usage

```
vdc(n, rand.rot = FALSE)
```

Arguments

n	Integer. The length of the sequence.
rand.rot	Logical. Should the sequence be randomly rotated?

Value

A matrix giving the coordinates of the points in the half unit sphere.

Author(s)

Mathieu Ribatet

References

Freulon, X., de Fouquet, C., 1991. Remarques sur la pratique des bandes tournantes a trois dimensions. Cahiers de geostatistique, Fascicule 1, Centre de Geostatistique, Ecole des Mines de Paris, Fontainebleau, pp. 101–117.

Examples

```
vdc(10)
```

Index