Package 'mvPot'

April 9, 2018

Type Package

Title Multivariate Peaks-over-Threshold Modelling for Spatial Extreme Events

Version 0.1.4

Date 2018-04-09

Description Tools for high-dimensional peaks-over-threshold inference and simulation of spatial extremal processes.

License GPL-2

Imports MASS, evd, numbers, gmp

RoxygenNote 6.0.1

URL http://github.com/r-fndv/mvPot

NeedsCompilation yes

Author Raphael de Fondeville [aut, cre],
Leo Belzile [aut],
Emeric Thibaud [ctb]

Maintainer Raphael de Fondeville <raphael.de-fondeville@epfl.ch>

Repository CRAN

Date/Publication 2018-04-09 19:02:54 UTC

R topics documented:

`mvPot-package` *Multivariate Peaks-over-Threshold Modelling for Extreme Events Analysis*

Description

The mvPot package provides functions to perform high-dimensional peaks-over-threshold inference of spatial processes such as the Brown–Resnick. Parallel implementation for censored likelihood allows up to 500 locations, whereas the gradient score can handle thousands of locations. The package also includes simulations algorithms for the Brown-Resnick max-stable process as well as its associated Pareto process.

Details

Package:	mvPot
Type:	Package
Version:	0.1.0
Date:	2016-05-26
License:	GPL2

The mvPot package provides functions to perform high-dimensional peaks-over-threshold inference of spatial processes such as the Brown–Resnick.

`spectralLikelihood` relies on the spectral likelihood as developed by Engelke et al. (2015). This methods is fast to compute, however it is not robust with regard to non-extreme components.

`censoredLikelihood` (Wadsworth and Tawn, 2013) is a likelihood function for exceedances with at least one component exceeding a threshold and where low components, i.e., components under their threshold,. This approach is robust and performs best but requires heavy computations. The implementation in this package makes use of quasi-Monte Carlo estimation and thus can handle 100 locations in a reasonable time and up to 500 when parallelized.

`scoreEstimation` is a faster alternative to the `censoredLikelihood`, which is more robust than `spectralLikelihood`. This method can also be used with any kind of differentiable risk functional (Fondeville and Davison, 2016). Here the algorithm is limited only by matrix inversion and thus thousands of locations can be used.

`simulBrownResnick` is an exact algorithm for simulation of Brown-Resnick max-stable processes as described in Dombry et al. (2015).

`simulPareto` allows for simulation of Pareto processes associated to log-Gaussian random functions.

Author(s)

Raphael de Fondeville

Maintainer: Raphael de Fondeville <raphael.de-fondeville@epfl.ch>

References

Fondeville, R. de and Davison A. (2016). High-dimensional Peaks-over-threshold Inference for Brown-Resnick Processes. Submitted.

Engelke, S. et al. (2015). Estimation of Huesler-Reiss Distributions and Brown-Resnick Processes. Journal of the Royal Statistical Society: Series B, 77(1):239-265

Wadsworth, J.L. and Tawn, J.A. (2013). Efficient Inference for Spatial Extreme Value Processes Associated to Log-Gaussian Random Function. Biometrika, 101(1):1-15.

Dombry, C., Engelke S., and Oesting, M. (2016). Exact Simulation of Max-stable processes. Biometrika, To appear.

Genz, A. and Bretz, F. (2009). Computations of Multivariate Normal and t Probabilities, volume 105. Springer, Dordrecht.

Genz, A. (2013). QSILATMVNV http://www.math.wsu.edu/faculty/genz/software/software.html

Examples

```
#Define semi-variogram function
vario <- function(h, alpha = 1.5){
    norm(h,type = "2")^alpha
}

#Define locations
loc <- expand.grid(1:4, 1:4)

#Simulate data
obs <- simulPareto(1000, loc, vario)

#Evaluate risk functional
sums <- sapply(obs, sum)

#Define weighting function
weigthFun <- function(x, u){
 x * (1 - exp(-(sum(x) / u - 1)))
}

#Define partial derivative of weighting function
dWeigthFun <- function(x, u){
 (1 - exp(-(sum(x) / u - 1))) + (x / u) * exp( - (sum(x) / u - 1))
}


#Select exceedances
threshold <- quantile(sums, 0.9)
exceedances <- obs[sums > threshold]

#Define objective function
objectiveFunction = function(parameter, exceedances, loc, vario, weigthFun, dWeigthFun, threshold){

 #Define semi-variogram for the corresponding parameters
 varioModel <- function(h){
  vario(h, parameter[1])
 }

 #Compute score
 scoreEstimation(exceedances, loc, varioModel, weigthFun, dWeigthFun, u = threshold)
}

#Estimate the parameter by optimization of the objective function
est <- optim(par = c(1.5),
             fn = objectiveFunction,
             exceedances = exceedances,
             loc = loc,
             vario = vario,
             weigthFun = weigthFun,
             dWeigthFun = dWeigthFun,
             threshold = threshold,
             control = list(maxit = 100, trace = 1),
             lower = c(0.01),
             upper = c(1.99),
             method = "L-BFGS-B")
```

`censoredLikelihoodBR` *Censored log-likelihood function for the Brown–Resnick model.*

Description

Compute the peaks-over-threshold censored negative log-likelihood function for the Brown–Resnick model.

Usage

```
censoredLikelihoodBR(obs, loc, vario, u, p = 499L, vec = NULL,
  nCores = 1L, cl = NULL)

censoredLikelihood(obs, loc, vario, u, p = 499L, vec = NULL, nCores = 1L,
  cl = NULL)
```

Arguments

`obs`	List of vectors for which at least one component exceeds a high threshold.
`loc`	Matrix of coordinates as given by `expand.grid()`.
`vario`	Semi-variogram function taking a vector of coordinates as input.
`u`	Vector of thresholds for censoring components.
`p`	Number of samples used for quasi-Monte Carlo estimation. Must be a prime number.
`vec`	Generating vector for the quasi-Monte Carlo procedure. For a given `p` and dimensionality, can be computed using `genVecQMC`.

`nCores`	Number of cores used for the computation
`cl`	Cluster instance as created by `makeCluster` of the `parallel` package.

Details

The function computes the censored log-likelihood function based on the representation developed by Wadsworth et al. (2014) and Engelke et al. (2015). Margins must have been standardized, for instance to unit Frechet.

Value

Evaluation of the negative censored log-likelihood function for the set of observations `obs` and semi-variogram `vario`.

Author(s)

Raphael de Fondeville

References

Wadsworth, J. L. and J. A. Tawn (2014). Efficient Inference for Spatial Extreme Value Processes Associated to Log-Gaussian Random Function. Biometrika, 101(1):1-15.

Asadi, P., Davison A. C. and S. Engelke (2015). Extremes on River Networks. Annals of Applied Statistics, 9(4), 2023-2050.

Examples

```
#Define semi-variogram function
vario <- function(h){
   0.5 * norm(h, type = "2")^1.5
}

#Define locations
loc <- expand.grid(1:4, 1:4)

#Simulate data
obs <- simulPareto(1000, loc, vario)

#Evaluate risk functional
maxima <- sapply(obs, max)
thres <- quantile(maxima, 0.9)

#Select exceedances
exceedances <- obs[maxima > thres]

#Compute generating vector
p <- 499
latticeRule <- genVecQMC(p, (nrow(loc) - 1))
primeP <- latticeRule$primeP
vec <- latticeRule$genVec
```

```
#Compute log-likelihood function
censoredLikelihoodBR(exceedances, loc, vario, rep(thres, nrow(loc)), primeP, vec)
```

---

censoredLikelihoodXS *Censored log-likelihood function of the extremal Student model*

Description

Compute the peaks-over-threshold censored negative log-likelihood function for the extremal Student model.

Usage

```
censoredLikelihoodXS(obs, loc, corrFun, nu, u, p = 499L, vec = NULL,
  nCores = 1L, cl = NULL, likelihood = c("mgp", "poisson", "binom"),
  ntot = length(obs), censored = TRUE)
```

Arguments

`obs`	List of vectors for which at least one component exceeds a high threshold.
`loc`	Matrix of coordinates as given by `expand.grid()`.
`corrFun`	correlation function taking a vector of coordinates as input.
`nu`	degrees of freedom of the Student process
`u`	Vector of thresholds for censoring components.
`p`	Number of samples used for quasi-Monte Carlo estimation. Must be a prime number.
`vec`	Generating vector for the quasi-Monte Carlo procedure. For a given `p` and dimensionality, can be computed using `genVecQMC`.
`nCores`	Number of cores used for the computation
`cl`	Cluster instance as created by `makeCluster` of the `parallel` package.
`likelihood`	string specifying the contribution. Either `"mgp"` for multivariate generalized Pareto, `"poisson"` for a Poisson contribution for the observations falling below or `"binom"` for a binomial contribution.
`ntot`	integer number of observations below and above the threshold, to be used with Poisson or binomial likelihood
`censored`	boolean indicating whether to censor observations lying below the threshold. Default to `TRUE`

Details

The function computes the censored log-likelihood function based on the representation developed by Ribatet (2013); see also Thibaud and Opitz (2015). Margins must have been standardized, for instance to unit Frechet.

Value

Evaluation of the censored log-likelihood function for the set of observations `obs` and correlation function `corrFun`.

Negative censored log-likelihood function for the set of observations `obs` and correlation function `corrFun`, with `attributes` `exponentMeasure`.

Author(s)

Leo Belzile

References

Thibaud, E. and T. Opitz (2015). Efficient inference and simulation for elliptical Pareto processes. Biometrika, 102(4), 855-870.

Ribatet, M. (2013). Spatial extremes: max-stable processes at work. JSFS, 154(2), 156-177.

Examples

```
#Define correlation function
corrFun <- function(h, alpha = 1, lambda = 1){
   exp(-norm(h, type = "2")^alpha/lambda)
}

#Define locations
loc <- expand.grid(1:4, 1:4)

#Compute generating vector
p <- 499L
latticeRule <- genVecQMC(p, (nrow(loc) - 1))
primeP <- latticeRule$primeP
vec <- latticeRule$genVec

#Simulate data
Sigma <- exp(-as.matrix(dist(loc))^0.8)
obs <- rExtremalStudentParetoProcess(n = 1000, nu = 2, Sigma = Sigma)
obs <- split(obs, row(obs))

#Evaluate risk functional
maxima <- sapply(obs, max)
thres <- quantile(maxima, 0.9)

#Select exceedances
exceedances <- obs[maxima > thres]

#Compute log-likelihood function
eval <- censoredLikelihoodXS(exceedances, loc, corrFun, nu = 2, rep(thres, nrow(loc)), primeP, vec)
```

genVecQMC *Generating vectors for lattice rules*

Description

Compute an efficient generating vector for quasi-Monte Carlo estimation.

Usage

```
genVecQMC(p, d, bt = rep(1, d), gm = c(1, (4/5)^(0:(d - 2))))
```

Arguments

`p`	number of samples to use in the quasi-Monte Carlo procedure.
`d`	Dimension of the multivariate integral to estimate.
`bt`	Tuning parameter for finding the vector. See D. Nuyens and R. Cools (2004) for more details.
`gm`	Tuning parameter for finding the vector. See D. Nuyens and R. Cools (2004) for more details.

Details

The function computes a generating vector for efficient multivariate integral estimation based on D. Nuyens and R. Cools (2004). If `p` is not a prime, the nearest smaller prime is used instead.

Value

`primeP`, the highest prime number smaller than `p` and `genVec`, a `d`-dimensional generating vector defining an efficient lattice rule for `primeP` samples.

Examples

```
#Define the number of sample.
p <- 500

#Choose a dimension
d <- 300

#Compute the generating vector
latticeRule <- genVecQMC(p,d)

print(latticeRule$primeP)
print(latticeRule$genVec)
```

`mvtNormQuasiMonteCarlo` *Multivariate normal distribution function*

Description

Estimate the multivariate distribution function with quasi-Monte Carlo method.

Usage

```
mvtNormQuasiMonteCarlo(p, upperBound, cov, genVec)
```

Arguments

`p`	Number of samples used for quasi-Monte Carlo estimation. Must be a prime number.
`upperBound`	Vector of probabilities, i.e., the upper bound of the integral.
`cov`	Covariance matrix of the multivariate normal distribution. Must be positive semi-definite. WARNING: for performance in high-dimensions, no check is performed on the matrix. It is the user responsibility to ensure that this property is verified.
`genVec`	Generating vector for the quasi-Monte Carlo procedure. Can be computed using `genVecQMC`.

Details

The function uses a quasi-Monte Carlo procedure based on randomly shifted lattice rules to estimate the distribution function a multivariate normal distribution as described in Genz, A. and Bretz, F.(2009).

Value

An estimate of the distribution function along with empirical Monte Carlo error.

References

Genz, A. and Bretz, F. (2009). Computations of Multivariate Normal and t Probabilities, volume 105. Springer, Dordrecht.

Genz, A. (2013). QSILATMVNV http://www.math.wsu.edu/faculty/genz/software/software.html

Examples

```
#Define locations
loc <- expand.grid(1:4, 1:4)
ref <- sample.int(16, 1)

#Compute variogram matrix
variogramMatrix <- ((sqrt((outer(loc[,1],loc[,1],"-"))^2 +
(outer(loc[,2],loc[,2],"-"))^2)) / 2)^(1.5)

#Define an upper boud
upperBound <- variogramMatrix[-ref,ref]

#Compute covariance matrix
cov <-  (variogramMatrix[-ref,ref]%*%t(matrix(1, (nrow(loc) - 1), 1)) +
t(variogramMatrix[ref,-ref]%*%t(matrix(1, (nrow(loc) - 1), 1))) -
variogramMatrix[-ref,-ref])

#Compute generating vector
p <- 499
latticeRule <- genVecQMC(p, (nrow(loc) - 1))

#Estimate the multivariate distribution function
mvtNormQuasiMonteCarlo(latticeRule$primeP, upperBound, cov, latticeRule$genVec)
```

`mvTProbQuasiMonteCarlo` *Multivariate t distribution function*

Description

Estimate the multivariate t distribution function with quasi-Monte Carlo method.

Usage

```
mvTProbQuasiMonteCarlo(p, upperBound, cov, nu, genVec)
```

Arguments

`p`	Number of samples used for quasi-Monte Carlo estimation. Must be a prime number.
`upperBound`	Vector of probabilities, i.e., the upper bound of the integral.
`cov`	Covariance matrix of the multivariate normal distribution. Must be positive semi-definite. WARNING: for performance in high-dimensions, no check is performed on the matrix. It is the user responsibility to ensure that this property is verified.
`nu`	Degrees of freedom of the t distribution.
`genVec`	Generating vector for the quasi-Monte Carlo procedure. Can be computed using `genVecQMC`.

Details

The function uses a quasi-Monte Carlo procedure based on randomly shifted lattice rules to estimate the distribution function a multivariate normal distribution as described in Genz, A. and Bretz, F.(2009).

Value

An estimate of the distribution function along with empirical Monte Carlo error.

Author(s)

Raphael de Fondeville

References

Genz, A. and Bretz, F. (2009). Computations of Multivariate Normal and t Probabilities, volume 105. Springer, Dordrecht.

Genz, A. (2013). QSILATMVTV http://www.math.wsu.edu/faculty/genz/software/software.html

Examples

```
#Define locations
loc <- expand.grid(1:4, 1:4)
ref <- sample.int(16, 1)

#Define degrees of freedom
nu <- 3

#Compute variogram matrix
variogramMatrix <- ((sqrt((outer(loc[,1],loc[,1],"-"))^2 +
(outer(loc[,2],loc[,2],"-"))^2)) / 2)^(1.5)

#Define an upper boud
upperBound <- variogramMatrix[-ref,ref]

#Compute covariance matrix
cov <-  (variogramMatrix[-ref,ref]%*%t(matrix(1, (nrow(loc) - 1), 1)) +
t(variogramMatrix[ref,-ref]%*%t(matrix(1, (nrow(loc) - 1), 1))) -
variogramMatrix[-ref,-ref])

#Compute generating vector
p <- 499
latticeRule <- genVecQMC(p, (nrow(loc) - 1))

#Estimate the multivariate distribution function
mvTProbQuasiMonteCarlo(latticeRule$primeP, upperBound, cov, nu, latticeRule$genVec)
```

rExtremalStudentParetoProcess

Simulation of extremal Student generalized Pareto vectors

Description

The algorithm is described in section 4 of Thibaud and Opitz. It uses the Cholesky decomposition of the matrix `Sigma` to generate samples on the unit sphere and an accept-reject algorithm to decide which samples to retain. If `normalize = TRUE` (the default), the vector is scaled by the exponent measure κ so that the maximum of the sample is greater than κ.

Usage

```
rExtremalStudentParetoProcess(n, Sigma, nu, normalize = FALSE,
  matchol = NULL)
```

Arguments

`n`	sample size
`Sigma`	correlation matrix
`nu`	degrees of freedom parameter
`normalize`	logical; should unit Pareto samples above κ be returned?
`matchol`	Cholesky matrix $\mathbf{A}$ such that $\mathbf{A}\mathbf{A}^\top = \boldsymbol{\Sigma}$. Corresponds to `t(chol(Sigma))`. Default to `NULL`, in which case the Cholesky root is computed within the function.

Author(s)

Emeric Thibaud, Leo Belzile

References

Thibaud, E. and T. Opitz (2015). Efficient inference and simulation for elliptical Pareto processes. Biometrika, 102(4), 855-870.

Examples

```
loc <- expand.grid(1:4, 1:4)
Sigma <- exp(-as.matrix(dist(loc))^1.5)
rExtremalStudentParetoProcess(1000, Sigma, nu = 2)
```

`scoreEstimation` *Gradient score function for the Brown–Resnick model.*

Description

Compute the peaks-over-threshold gradient score function for the Brown–Resnick model.

Usage

```
scoreEstimation(obs, loc, vario, weightFun = NULL, dWeightFun = NULL,
  nCores = 1L, cl = NULL, ...)
```

Arguments

`obs`	List of vectors exceeding an R-threshold, see de Fondeville and Davison (2016) for more details.
`loc`	Matrix of coordinates as given by `expand.grid()`.
`vario`	Semi-variogram function taking a vector of coordinates as input.
`weightFun`	Function of weights.
`dWeightFun`	Partial derivative function of `weightFun`.
`nCores`	Number of cores used for the computation
`cl`	Cluster instance as created by `makeCluster` of the `parallel` package.
`...`	Parameters for `weightFun` and `dWeightFun`.

Details

The function computes the gradient score based on the representation developed by Wadsworth et al. (2014). Margins must have been standardized. The weighting function must be differentiable and verify some properties for consistency, see de Fondeville and Davison (2016) for more details.

Value

Evaluation of the gradient score function for the set of observations `obs` and semi-variogram `vario`.

Author(s)

Raphael de Fondeville

References

de Fondeville, R. and A. C. Davison (2017). High-dimensional Peaks-over-threshold Inference for Brown-Resnick Processes. Submitted.

Examples

```
#Define variogram function
vario <- function(h){
   1 / 2 * norm(h,type = "2")^1.5
}

#Define locations
loc <- expand.grid(1:4, 1:4)

#Simulate data
obs <- simulPareto(1000, loc, vario)

#Evaluate risk functional
sums <- sapply(obs, sum)

#Define weighting function
weightFun <- function(x, u){
 x * (1 - exp(-(sum(x / u) - 1)))
}

#Define partial derivative of weighting function
dWeightFun <- function(x, u){
(1 - exp(-(sum(x / u) - 1))) + (x / u) * exp( - (sum(x / u) - 1))
}

#Select exceedances
threshold <- quantile(sums, 0.9)
exceedances <- obs[sums > threshold]

#Evaluate gradient score function
scoreEstimation(exceedances, loc, vario, weightFun = weightFun, dWeightFun, u = threshold)
```

`simulBrownResnick` *Simulation of Brown–Resnick random vectors*

Description

`simulBrownResnick` provides `n` replicates of a Brown–Resnick max-stable process with semi-variogram `vario` at locations `loc`.

Usage

```
simulBrownResnick(n, loc, vario, nCores = 1, cl = NULL)
```

Arguments

`n`	Number of replicates desired.
`loc`	Matrix of coordinates as given by `expand.grid()`.
`vario`	Semi-variogram function.

`nCores`	Number of cores needed for the computation
`cl`	Cluster instance as created by `makeCluster` of the `parallel` package. Make sure the random number generator has been properly initialized with `clusterSetRNGStream()`.

Details

The algorithm used here is based on the spectral representation of the Brown–Resnick model as described in Dombry et al. (2015). It provides `n` exact simulations on the unit Frechet scale and requires, in average, for each max-stable vector, the simulation of d Pareto processes, where d is the number of locations.

Value

List of `n` random vectors drawn from a max-stable Brown–Resnick process with semi-variogram `vario` at location `loc`.

Examples

```
#Define semi-variogram function
vario <- function(h){
   1 / 2 * norm(h,type = "2")^1.5
}

#Define locations
loc <- expand.grid(1:4, 1:4)

#Simulate data
obs <- simulBrownResnick(10, loc, vario)
```

`simulPareto` *Simulate Pareto random vectors*

Description

`simulPareto` provides `n` replicates of the multivariate Pareto distribution associated to log-Gaussian random function with semi-variogram `vario`.

Usage

```
simulPareto(n, loc, vario, nCores = 1, cl = NULL)
```

Arguments

`n`	Number of replicates desired.
`loc`	Matrix of coordinates as given by `expand.grid()`.
`vario`	Semi-variogram function.
`nCores`	Number of cores used for the computation
`cl`	Cluster instance as created by `makeCluster` of the `parallel` package. Make sure the random number generator has been properly initialized with `clusterSetRNGStream()`.

Details

The algorithm used here is based on the spectral representation of the Brown–Resnick model as described in Dombry et al. (2015). It provides `n` replicates conditioned that `mean(x) > 1` on the unit Frechet scale.

Value

List of `n` random vectors drawn from a multivariate Pareto distribution with semi-variogram `vario`.

Examples

```
#Define variogram function
vario <- function(h){
   1 / 2 * norm(h,type = "2")^1.5
}

#Define locations
loc <- expand.grid(1:4, 1:4)

#Simulate data
obs <- simulPareto(100, loc, vario)
```

`spectralLikelihood` *Spectral log-likelihood function*

Description

Compute the spectral log-likelihood function for Brown–Resnick model with peaks-over-threshold.

Usage

```
spectralLikelihood(obs, loc, vario, nCores = 1L, cl = NULL)
```

Arguments

`obs`	List of observations vectors for which `sum(x)` exceeds a high threshold.
`loc`	Matrix of coordinates as given by `expand.grid()`.
`vario`	Semi-variogram function taking a vector of coordinates as input.
`nCores`	Number of cores used for the computation
`cl`	Cluster instance as created by `makeCluster` of the `parallel` package.

Details

The function compute the log-likelihood function based on the spectral representation developed by Engelke et al. (2015). This simplified expression is obtained by conditioning on the event '`sum(x)` exceeds a high threshold `u > 1`'. Margins must have been standardized.

Value

Evaluation of the spectral likelihood function for the set of observations `obs` and semi-variogram `vario`.

References

Engelke, S. et al. (2015). Estimation of Huesler-Reiss Distributions and Brown-Resnick Processes. Journal of the Royal Statistical Society: Series B, 77(1):239-265

Examples

```
#Define semi-variogram function
vario <- function(h){
   1 / 2 * norm(h,type = "2")^1.5
}

#Define locations
loc <- expand.grid(1:4, 1:4)

#Simulate data
obs <- simulPareto(1000, loc, vario)

#Evaluate risk functional
sums <- sapply(obs, sum)

#Select exceedances
exceedances <- obs[sums > quantile(sums, 0.9)]

#Evaluate the spectral function
spectralLikelihood(exceedances, loc, vario)
```

Index