Package ‘HydroPortailStats’

January 20, 2025

Type Package

Title 'HydroPortail' Statistical Functions

Version 1.1.0

Description
Statistical functions used in the French 'HydroPortail' <https://hydro.eaufrance.fr/>.
This includes functions to estimate distributions, quantile curves and uncertainties, along with various other utilities.
Technical details are available (in French) in Renard (2016) <https://hal.inrae.fr/hal-02605318>.

License GPL-3

Encoding UTF-8

LazyData true

URL https://github.com/benRenard/HydroPortailStats

BugReports https://github.com/benRenard/HydroPortailStats/issues

Depends R (>= 3.5.0)

Imports stats, graphics, evd, mvtnorm, numDeriv

RoxygenNote 7.2.3

Suggests knitr, rmarkdown

NeedsCompilation no

Author Benjamin Renard [aut, cre, cph]
(<https://orcid.org/0000-0001-8447-5430>),
INRAE [fnd],
Ministère de la Transition Ecologique - SCHAPI [fnd],
European Commission [fnd] (This project has received funding from the European Union’s Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 835496)

Maintainer Benjamin Renard <benjamin.renard@inrae.fr>

Repository CRAN

Date/Publication 2024-08-19 13:50:02 UTC

Contents

`distInfo` *Information on available distributions*

Description

A named list containing information (parameters, contraints, notes, warnings, etc.) for all available univariate distributions.

Usage

```
distInfo
```

Format

A named list where each element is itself a list containing:

parName parameters short names

parLongName parameters long names

parSymbol parameters typical symbols

constraints constraints on parameters

url link to more information

note notes

warning warnings: read carefully since this highlights in particular differences with "standard" parameterizations found in e.g. Wikipedia or R.

`Generate` *Random numbers generator*

Description

Generate random realizations from a distribution

Usage

```
Generate(dist, par, n = 1)
```

Arguments

`dist`	character, distribution name
`par`	numeric vector, parameter vector
`n`	integer, number of values to generate

Value

The generated values as a numeric vector.

Examples

```
Generate('Normal',c(0,1),10)
Generate('GEV',c(100,25,-0.2),10)
Generate('GEV',c(100,25,0.2),10)
Generate('Poisson',0.75,10)
```

GenerateWithinBounds *Constrained random numbers generator*

Description

Generate random realizations from a distribution, constraining these realizations to stay within bounds.

Usage

```
GenerateWithinBounds(dist, par, n = 1, lowerBound = -Inf, higherBound = Inf)
```

Arguments

dist	character, distribution name
par	numeric vector, parameter vector
n	integer, number of values to generate
lowerBound	Numeric, lower bound
higherBound	Numeric, higher bound, should be strictly larger than the lower bound

Value

The generated values as a numeric vector.

Examples

```
set.seed(123456)
y0=GenerateWithinBounds(dist='GEV',par=c(0,1,-0.2),n=1000)
y1=GenerateWithinBounds(dist='GEV',par=c(0,1,-0.2),n=1000,lowerBound=0,higherBound=5)
plot(y0);points(y1,col='red')
```

GetCdf *Cumulative Distribution Function (cdf)*

Description

Evaluates the cdf of a distribution

Usage

```
GetCdf(y, dist, par)
```

Arguments

`y`	numeric, value at which the cdf is evaluated
`dist`	character, distribution name
`par`	numeric vector, parameter vector

Value

The cdf as a numeric.

Examples

```
GetCdf(0,'Normal',c(0,1))
GetCdf(200,'GEV',c(100,25,-0.2))
GetCdf(200,'GEV',c(100,25,0.2))
GetCdf(3,'Poisson',0.75)
```

`GetEmpFreq` *Empirical nonexceedance frequency*

Description

Computes the empirical nonexceedance frequency of the ith sorted value amongst n

Usage

```
GetEmpFreq(i, n, formula = "Hazen")
```

Arguments

`i`	integer or integer vector, observation rank(s)
`n`	integer, number of observations
`formula`	character, formula, available: 'Hazen', 'Standard', 'MinusOne', 'Weibull', 'Benard', 'Cunnane', 'Beard', 'Blom', 'Gringorten', 'Landwehr', 'Tukey'.

Value

The nonexceedance frequency.

Examples

```
GetEmpFreq(i=1:10,n=10)
GetEmpFreq(i=1:10,n=10,formula='Standard')
GetEmpFreq(i=1:10,n=10,formula='MinusOne')
GetEmpFreq(i=1:10,n=10,formula='Cunnane')
```

`GetEstimate_BAY` *Bayesian estimation of a distribution*

Description

Returns MCMC samples from the posterior distribution.

Usage

```
GetEstimate_BAY(
  y,
  dist,
  prior,
  par0,
  mult = 0.1,
  eps = 0.1,
  batch.length = 100,
  batch.n = 100,
  moverate.min = 0.1,
  moverate.max = 0.5,
  mult.down = 0.9,
  mult.up = 1.1
)
```

Arguments

`y`	numeric vector, data
`dist`	character, distribution name
`prior`	list of lists, prior distributions. For each parameter to be estimated, the prior is a list of the form pr=list(dist=..., par=...). See example below.
`par0`	numeric vector, initial parameter guess. You may use GetEstimate_ROUGH().
`mult`	numeric, initial jump standard deviations are set to mult * abs(par0)
`eps`	numeric, where par0 is zero, initial jump standard deviations are set to eps (to avoid jumps of size zero)
`batch.length`	integer, MCMC parameter: length of each non-adaptive batch
`batch.n`	integer, MCMC parameter: number of batches (= adaptation period). Total number of simulations is nsim=batch.n*batch.length
`moverate.min`	numeric in (0;1), MCMC parameter: lower bound for the desired move rate interval
`moverate.max`	numeric in (0;1), MCMC parameter: upper bound for the desired move rate interval
`mult.down`	numeric in (0;1), MCMC parameter: multiplication factor used to decrease jump size when move rate is too low.
`mult.up`	numeric (>1, avoid 1/mult.down), MCMC parameter: multiplication factor used to increase jump size when move rate is too high.

Value

A list with the following components:

x	numeric matrix nsim*length(par0), MCMC simulations
fx	numeric vector, corresponding values f(x)

Examples

```
y=c(9.2,9.5,11.4,9.5,9.4,9.6,10.5,11.1,10.5,10.4)
prior1=list(dist='FlatPrior',par=NULL)
prior2=list(dist='LogNormal',par=c(1,1))
prior3=list(dist='Normal',par=c(0,0.25))
prior=list(prior1,prior2,prior3)
par0=GetEstimate_ROUGH(y,'GEV')$par
mcmc=GetEstimate_BAY(y,'GEV',prior,par0,batch.length=50,batch.n=50)
graphicalpar=par(mfrow=c(2,3))
plot(mcmc$x[,1],type='l'); plot(mcmc$x[,2],type='l'); plot(mcmc$x[,3],type='l')
hist(mcmc$x[,1]); hist(mcmc$x[,2]); hist(mcmc$x[,3])
par(graphicalpar)
```

GetEstimate_HBay *Bayesian estimation using historical data*

Description

Bayesian estimation of a GEV or Gumbel distribution based on a mixed sample containing point (i.e. perfectly known) or interval (i.e. known to be within bounds) data. Systematic errors induced by rating curve errors can also be accounted for. Returns MCMC samples from the posterior distribution.

Usage

```
GetEstimate_HBay(
  y,
  dist,
  prior,
  SystErrorIndex = rep(0, NROW(y)),
  SystErrorPrior = list(),
 par0 = GetEstimate_ROUGH(0.5 * (y[, 1] + y[, 2])[is.finite(y[, 1] + y[, 2])], dist)$par,
  SystError0 = rep(1, length(SystErrorPrior)),
  mult = 0.1,
  eps = 0.1,
  batch.length = 100,
  batch.n = 100,
  moverate.min = 0.1,
  moverate.max = 0.5,
  mult.down = 0.9,
  mult.up = 1.1
)
```

Arguments

`y`	numeric 2-column matrix, data. The first column gives the lower bound, the second column gives the upper bound. Where y[i,1]==y[i,2], the value is assumed perfectly known (up to systematic errors, see below). Where y[i,1]<y[i,2], the value is assumed to be in the interval [y[i,1];y[i,2]] -Inf and +Inf are allowed for data being only right- or left-censored (i.e. values known to be smaller than or larger than some threshold).
`dist`	character, distribution name. Only distributions 'GEV' and 'Gumbel' are supported.
`prior`	list of lists, prior distributions. For each parameter to be estimated, the prior is a list of the form pr=list(dist=..., par=...). See example below.
`SystErrorIndex`	integer vector, length NROW(y). Index of systematic errors. Rows where SystErrorIndex==k are all affected by the same multiplicative error gamma_k, typically induced by the kth rating curve. SystErrorIndex==0 means no systematic error. Should only contain integer values between 0 and N_{systematic errors}.
`SystErrorPrior`	list of lists, prior distribution for each systematic error. For instance for a systematic error in the range +/- 20%, you may use a Uniform between 0.8 and 1.2, or a triangular distribution with the same bounds and peak at 1.
`par0`	numeric vector, initial parameter guess.
`SystError0`	numeric vector, initial guess for systematic errors. Typically a vector of 1.
`mult`	numeric, initial jump standard deviations are set to mult * abs(par0)
`eps`	numeric, where par0 is zero, initial jump standard deviations are set to eps (to avoid jumps of size zero)
`batch.length`	integer, MCMC parameter: length of each non-adaptive batch
`batch.n`	integer, MCMC parameter: number of batches (= adaptation period). Total number of simulations is nsim=batch.n*batch.length
`moverate.min`	numeric in (0;1), MCMC parameter: lower bound for the desired move rate interval
`moverate.max`	numeric in (0;1), MCMC parameter: upper bound for the desired move rate interval
`mult.down`	numeric in (0;1), MCMC parameter: multiplication factor used to decrease jump size when move rate is too low.
`mult.up`	numeric (>1, avoid 1/mult.down), MCMC parameter: multiplication factor used to increase jump size when move rate is too high.

Value

A list with the following components:

`x`	numeric matrix nsim * (length(par0)+length(SystError0)), MCMC simulations
`fx`	numeric vector, corresponding values f(x)

Examples

```
set.seed(98765)
n=50;n_censored=30
y0=Generate('GEV',c(100,50,-0.2),n)
y=cbind(y0,y0)
# Mimics censoring between 0 and 300 for first n_censored years
y[1:n_censored,1][y0[1:n_censored]<300]=0
y[1:n_censored,2][y0[1:n_censored]<300]=300
plot(y[,1]);points(y[,2])
# Systematic errors
SystErrorIndex=c(rep(1,n_censored),rep(2,n-n_censored))
SystErrorPrior=list(list(dist="Triangle",par=c(1,0.7,1.3)),
                    list(dist="Triangle",par=c(1,0.95,1.05)))
# Priors on GEV parameters
prior=list(list(dist="FlatPrior",par=NULL),
           list(dist="FlatPrior",par=NULL),
           list(dist="Normal",par=c(0,0.25)))
# Go!
mcmc=GetEstimate_HBay(y=y,dist='GEV',prior=prior,
                      SystErrorIndex=SystErrorIndex,
                      SystErrorPrior=SystErrorPrior,
                      # The values below aim at making this example fast to run.
                      # In practice, it is recommended to use the default values
                      # (batch.length=100,batch.n=100) or larger.
                      batch.length=25,batch.n=20)
graphicalpar=par(mfrow=c(2,3))
for(i in 1:5){hist(mcmc$x[,i])}
par(graphicalpar)
```

`GetEstimate_HYDRO2` *Hydro2 estimate of a distribution*

Description

Returns an estimate of a distribution as it was computed in the old HYDRO2 software. Only available for distributions 'Normal', 'LogNormal', and 'Gumbel'.

Usage

```
GetEstimate_HYDRO2(y, dist)
```

Arguments

`y`	numeric vector, data
`dist`	character, distribution name

Value

A list with the following components:

par	numeric vector, estimated parameter vector.
obj	numeric, objective fonction (NA for this estimate)
ok	logical, did computation succeed?
err	integer, error code (0 if ok)
message	error message

Examples

```
y=c(9.2,9.5,11.4,9.5,9.4,9.6,10.5,11.1,10.5,10.4)
GetEstimate_HYDRO2(y,'Normal')
GetEstimate_HYDRO2(y,'LogNormal')
GetEstimate_HYDRO2(y,'Gumbel')
GetEstimate_HYDRO2(y,'GEV')
GetEstimate_HYDRO2(y,'Poisson')
```

GetEstimate_LMOM *L-Moment estimate of a distribution*

Description

Returns an estimate of a distribution using the method of L-moments. Note that for some distributions, this is not strictly speaking the L-moment estimate: For LogNormal and LogPearsonIII, the L-moment estimate of log(data) is used.

Usage

```
GetEstimate_LMOM(y, dist)
```

Arguments

y	numeric vector, data
dist	character, distribution name

Value

A list with the following components:

par	numeric vector, estimated parameter vector.
obj	numeric, objective fonction (NA for this estimate)
ok	logical, did computation succeed?
err	integer, error code (0 if ok)
message	error message

Examples

```
y=c(9.2,9.5,11.4,9.5,9.4,9.6,10.5,11.1,10.5,10.4)
GetEstimate_LMOM(y,'Normal')
GetEstimate_LMOM(y,'LogNormal')
GetEstimate_LMOM(y,'Gumbel')
GetEstimate_LMOM(y,'GEV')
GetEstimate_LMOM(y,'Poisson')
```

GetEstimate_ML *Maximum-likelihood estimate of a distribution*

Description

Returns an estimate of a distribution using the method of maximum likelihood.

Usage

```
GetEstimate_ML(
  y,
  dist,
  par0 = NULL,
  method = optim_method_def,
  lower = -Inf,
  upper = Inf
)
```

Arguments

y	numeric vector, data
dist	character, distribution name
par0	numeric vector, initial parameter guess. You may use GetEstimate_ROUGH().
method	character, method used to maximize likelihood, see ?optim
lower	numeric vector, lower bounds, see ?optim
upper	numeric vector, upper bounds, see ?optim

Value

A list with the following components:

par	numeric vector, estimated parameter vector.
obj	numeric, objective fonction (maximum log-likelihood)
ok	logical, did computation succeed?
err	integer, error code (0 if ok)
message	error message

Examples

```
y=c(9.2,9.5,11.4,9.5,9.4,9.6,10.5,11.1,10.5,10.4)
GetEstimate_ML(y,'Normal')
GetEstimate_ML(y,'LogNormal')
GetEstimate_ML(y,'Gumbel')
GetEstimate_ML(y,'Gumbel',par0=GetEstimate_ROUGH(y,'Gumbel')$par)
GetEstimate_ML(y,'GEV',par0=GetEstimate_ROUGH(y,'GEV')$par)
GetEstimate_ML(y,'Poisson')
```

`GetEstimate_MOM` *Moment estimate of a distribution*

Description

Returns an estimate of a distribution using the method of moments. Note that for some distributions, this is not strictly speaking the moment estimate. For LogPearsonIII for instance, the moment estimate of log(data) is used. Also for GPD3, the threshold is estimated as min(data).

Usage

```
GetEstimate_MOM(y, dist)
```

Arguments

`y`	numeric vector, data
`dist`	character, distribution name

Value

A list with the following components:

`par`	numeric vector, estimated parameter vector.
`obj`	numeric, objective fonction (NA for this estimate)
`ok`	logical, did computation succeed?
`err`	integer, error code (0 if ok)
`message`	error message

Examples

```
y=c(9.2,9.5,11.4,9.5,9.4,9.6,10.5,11.1,10.5,10.4)
GetEstimate_MOM(y,'Normal')
GetEstimate_MOM(y,'LogNormal')
GetEstimate_MOM(y,'Gumbel')
GetEstimate_MOM(y,'GEV')
GetEstimate_MOM(y,'Poisson')
```

GetEstimate_ROUGH *Rough estimate of a distribution*

Description

Returns a rough first-guess estimate of a distribution. This estimate may be poor but it solely aims at being used as a starting point for more advanced estimation approaches (e.g. max-likelihood or Bayesian). It is therefore chosen as an easy-to-compute explicit formula, robust and error-proof.

Usage

```
GetEstimate_ROUGH(y, dist)
```

Arguments

y	numeric vector, data
dist	character, distribution name

Value

A list with the following components:

par	numeric vector, estimated parameter vector.
obj	numeric, objective fonction (NA for this estimate)
ok	logical, did computation succeed?
err	integer, error code (0 if ok)
message	error message

Examples

```
y=c(9.2,9.5,11.4,9.5,9.4,9.6,10.5,11.1,10.5,10.4)
GetEstimate_ROUGH(y,'Normal')
GetEstimate_ROUGH(y,'LogNormal')
GetEstimate_ROUGH(y,'Gumbel')
GetEstimate_ROUGH(y,'GEV')
GetEstimate_ROUGH(y,'Poisson')
```

GetParFeas *Parameter feasibility*

Description

Evaluates whether a parameter vector is feasible (for instance, are scale parameters >0 ?)

Usage

```
GetParFeas(dist, par)
```

Arguments

dist	character, distribution name
par	numeric vector, parameter vector

Value

A logical.

Examples

```
# Feasible
GetParFeas('Normal',c(0,1))
# Not feasible because second parameter (standard deviation) is negative
GetParFeas('Normal',c(0,-1))
```

GetParName *Parameter names.*

Description

Returns the names of the parameters of a distribution, in French (default) or English.

Usage

```
GetParName(dist, lang = "fr")
```

Arguments

dist	character, distribution name
lang	character, language ('en' or 'fr')

Value

A character vector.

Examples

```
GetParName('Normal')
GetParName('GEV')
GetParName('GEV',lang='en')
```

GetParNumber *Number of parameters.*

Description

Returns the number of parameters of a distribution.

Usage

```
GetParNumber(dist)
```

Arguments

dist	character, distribution name

Value

An integer.

Examples

```
GetParNumber('Normal')
GetParNumber('GEV')
```

GetPdf *Probability Density Function (pdf)*

Description

Evaluates the pdf of a distribution

Usage

```
GetPdf(y, dist, par, log = FALSE)
```

Arguments

y	numeric, value at which the pdf is evaluated
dist	character, distribution name
par	numeric vector, parameter vector
log	logical, returns log-pdf if TRUE

Value

The pdf or the log-pdf as a numeric.

Examples

```
GetPdf(0,'Normal',c(0,1))
GetPdf(200,'GEV',c(100,25,-0.2))
GetPdf(200,'GEV',c(100,25,0.2))
GetPdf(3,'Poisson',0.75)
```

GetQfromT   *Get quantile from return period*

Description

Compute the T-quantile from the results of Hydro3_Estimation()

Usage

```
GetQfromT(RP, H3, options = options_def)
```

Arguments

`RP`	numeric, return period
`H3`	list, resulting from a call to Hydro3_Estimation()
`options`	list, see ?Hydro3_Estimation

Value

A list with the following components:

`q`	numeric, quantile
`IC`	numeric vector, uncertainty interval

Examples

```
y=stats::rnorm(50)
H3=Hydro3_Estimation(y,'Normal')
GetQfromT(100,H3)
```

GetQuantile — *Quantile Function*

Description

Evaluates the quantiles of a distribution

Usage

```
GetQuantile(p, dist, par)
```

Arguments

`p`	numeric in (0;1), nonexceedance probability
`dist`	character, distribution name
`par`	numeric vector, parameter vector

Value

The p-quantile as a numeric.

Examples

```
GetQuantile(0.99,'Normal',c(0,1))
GetQuantile(0.99,'GEV',c(100,25,-0.2))
GetQuantile(0.99,'GEV',c(100,25,0.2))
GetQuantile(0.99,'Poisson',0.75)
```

GetReducedVariate — *Reduced variate*

Description

Returns the 'reduced variate' that is used in some quantile plots (see e.g. quantile curve on Gumbel paper)

Usage

```
GetReducedVariate(p, dist)
```

Arguments

`p`	numeric in (0;1), nonexceedance probability
`dist`	character, distribution name

Value

The reduced variate with nonexceedance probability p.

Examples

```
GetReducedVariate(0.99,'Normal')
GetReducedVariate(0.99,'Gumbel')
GetReducedVariate(0.99,'GEV')
GetReducedVariate(0.99,'Poisson')
```

GetTfromQ *Get return period from value*

Description

Compute the return period associated with a value from the results of Hydro3_Estimation()

Usage

```
GetTfromQ(q, H3, options = options_def)
```

Arguments

q	numeric, value
H3	list, resulting from a call to Hydro3_Estimation()
options	list, see ?Hydro3_Estimation

Value

A list with the following components:

RP	numeric, return period
IC	numeric vector, uncertainty interval

Examples

```
y=stats::rnorm(50)
H3=Hydro3_Estimation(y,'Normal')
GetTfromQ(3,H3)
```

GetUncertainty_ML *Maximum-likelihood estimation of uncertainty*

Description

Returns an estimate of the uncertainty around the maximum-likelihood estimate, in the form of a covariance matrix and some simulations from the corresponding Gaussian distribution.

Usage

```
GetUncertainty_ML(y, dist, par, nsim = nsim_def)
```

Arguments

y	numeric vector, data
dist	character, distribution name
par	numeric vector, estimated parameter (using GetEstimate_ML()).
nsim	integer, number of simulated parameter replicates.

Value

A list with the following components:

cov	numeric matrix npar*npar, covariance matrix.
sim	numeric matrix nsim*npar, simulated parameter replicates.
ok	logical, did computation succeed?
err	integer, error code (0 if ok)
message	error message

Examples

```
y=c(9.2,9.5,11.4,9.5,9.4,9.6,10.5,11.1,10.5,10.4)
estim=GetEstimate_ML(y,'Gumbel',par0=GetEstimate_ROUGH(y,'Gumbel')$par)
GetUncertainty_ML(y,'Gumbel',par=estim$par)
```

HBay_Plot *HBay plot*

Description

Plot summarizing the results of Hydro3_HBay()

Usage

```
HBay_Plot(H3, curve_color = "black")
```

Arguments

H3	list, resulting from a call to Hydro3_HBay()
curve_color	color, color used for quantile curve

Value

nothing (just creates a plot)

Examples

```
set.seed(98765)
n=50;n_censored=30
y0=Generate('GEV',c(100,50,-0.2),n)
y=cbind(y0,y0)
# Mimics censoring between 0 and 300 for first n_censored years
y[1:n_censored,1][y0[1:n_censored]<300]=0
y[1:n_censored,2][y0[1:n_censored]<300]=300
plot(y[,1]);points(y[,2])
# Systematic errors
SystErrorIndex=c(rep(1,n_censored),rep(2,n-n_censored))
SystErrorPrior=list(list(dist="Triangle",par=c(1,0.7,1.3)),
                    list(dist="Triangle",par=c(1,0.95,1.05)))
# Priors on GEV parameters
prior=list(list(dist="FlatPrior",par=NULL),
           list(dist="FlatPrior",par=NULL),
           list(dist="Normal",par=c(0,0.25)))
# Handle MCMC options
# The values below aim at making this example fast to run.
# In practice, it is recommended to use the default values
# (batch.length=100,batch.n=100) or larger.
mcmcoptions=mcmcoptions_def
mcmcoptions$batch.length=25
mcmcoptions$batch.n=20
# Go!
H3=Hydro3_HBay(y=y,dist='GEV',prior=prior,
               SystErrorIndex=SystErrorIndex,
               SystErrorPrior=SystErrorPrior,
```

```
                mcmcoptions=mcmcoptions)
# HBay plot
HBay_Plot(H3)
```

Hydro3_Estimation *Hydro3 estimation*

Description

Main estimation function used in the HydroPortail. In short, this function estimates a distribution and the associated uncertainty, and returns all needed information to display and plot the results (parameter estimates, quantile curves, etc.)

Usage

```
Hydro3_Estimation(
  y,
  dist,
  Emeth = Emeth_def,
  Umeth = Umeth_def,
  options = options_def,
  mcmcoptions = mcmcoptions_def,
  prior = GetDefaultPrior(GetParNumber(dist)),
  do.KS = TRUE,
  do.MK = TRUE,
  do.Pettitt = TRUE
)
```

Arguments

`y`	numeric vector, data.
`dist`	character, distribution name. See dataset distInfo for a description of available distributions. In particular, type names(distInfo) for the list of available distributions, and distInfo[['GEV']] for more information on a particular distribution (here, GEV).
`Emeth`	character, estimation method. Default is 'LMOM' (L-Moments), available: 'MOM' (Moments), 'ML' (Maximum Likelihood), 'BAY' (Bayesian).
`Umeth`	character, uncertainty quantification method. Default is 'PBOOT' (Parametric bootstrap), available: 'BOOT' (Bootstrap, not recommended), 'NONE', 'ML' (only usable when Emeth='ML' as well), and 'BAY' (the only usable method when Emeth='BAY').
`options`	list, options, see details below.
`mcmcoptions`	list, MCMC options, see details below.
`prior`	list, prior distributions, only used when Emeth='BAY'. See ?GetEstimate_BAY for details.
`do.KS, do.MK, do.Pettitt`	logical, perform KS/MK/Pettitt tests?

Details

The argument 'options' allows controlling various properties of the analysis and results. It is a list with the following components:

- FreqFormula, character, formula for computing nonexceedance frequency, see ?GetEmpFreq.
- pgrid, numeric vector, probabilities defining the x values where pdf f(x) and cdf F(x) are computed. These x values are quantiles from the estimated distribution with probabilities pgrid.
- Tgrid, numeric vector, return periods where quantile function q(T) is computed.
- IClevel, numeric, level of uncertainty interval.
- p2T, numeric, conversion factor between nonexceedance probability p and return period T. p=1-1/(p2T*T). In general p2T=1 but for a peak-over-threshold approach leading to say 3 events per year on average, p2T=3.
- invertT, logical, when invertT=TRUE, LARGE return periods correspond to SMALL data values. This is typically used for low-flow statistics.
- splitZeros, logical, when splitZeros=TRUE zero and negative values are removed from the data y before estimating the distribution,and are used to estimate the probability of zeros p0. This is typically used for low-flow statistics to estimate the probability of zero streamflow.
- lang, chanracter, language ('fr' or 'en').
- nsim, integer, number of replicated parameters representing uncertainty.

The argument 'mcmcoptions' is only used when Emeth='BAY' and is a list controlling MCMC properties:

- mult, numeric, see ?Metropolis_OAAT_adaptive
- eps, numeric, see ?Metropolis_OAAT_adaptive
- batch.length, integer, see ?Metropolis_OAAT_adaptive
- batch.n, integer, see ?Metropolis_OAAT_adaptive
- moverate.min, numeric, see ?Metropolis_OAAT_adaptive
- moverate.max, numeric, see ?Metropolis_OAAT_adaptive
- mult.down, numeric, see ?Metropolis_OAAT_adaptive
- mult.up, numeric, see ?Metropolis_OAAT_adaptive
- burn, numeric, burn-in factor, e.g. if burn=0.2 the first 20 percents of MCMC samples are discarded
- slim, integer, sliming factor, e.g. if slim=5 only one MCMC sample every 5 is kept (after burn-in)

Value

A list with the following components:

`dist`	character, estimated distribution.
`ok`	logical, did estimation succeed?
`err`	integer, error code (0 if ok).

message	error message.
empirical	data frame, sorted data and empirical estimates (nonexceedance frequency, return period and reduced variate)
pcdf	data frame, estimated pdf and cdf
quantile	data frame, estimated quantiles and uncertainty intervals
par	data frame, estimated parameters and uncertainty intervals
KS	list, result of the Kolmogorov-Smirnov test, see ?KS
MK	list, result of the Mann-Kendall test, see ?MK
Pettitt	list, result of the Pettitt test, see ?Pettitt
u	list, parameter uncertainty in the form of a covariance matrix ($cov) and simulated parameter replicates ($sim). Also contains error-handling flags $ok, $err and $message.

Examples

```
y=stats::rnorm(50)
H3=Hydro3_Estimation(y,'Normal')
H3=Hydro3_Estimation(y,'GEV',Emeth='ML',Umeth='ML')
```

Hydro3_HBay *Bayesian estimation using historical data*

Description

Bayesian estimation of a GEV or Gumbel distribution based on a mixed sample containing point (i.e. perfectly known) or interval (i.e. known to be within bounds) data. Systematic errors induced by rating curve errors can also be accounted for. Returns an Hydro3 object

Usage

```
Hydro3_HBay(
  y,
  dist,
  prior = GetDefaultPrior(GetParNumber(dist)),
  SystErrorIndex = rep(0, NROW(y)),
  SystErrorPrior = list(),
  options = options_def,
  mcmcoptions = mcmcoptions_def,
  do.KS = TRUE,
  do.MK = TRUE,
  do.Pettitt = TRUE
)
```

Arguments

`y` numeric 2-column matrix, data. The first column gives the lower bound, the second column gives the upper bound. Where y[i,1]==y[i,2], the value is assumed perfectly known (up to systematic errors, see below). Where y[i,1]<y[i,2], the value is assumed to be in the interval [y[i,1];y[i,2]] -Inf and +Inf are allowed for data being only right- or left-censored (i.e. values known to be smaller than or larger than some threshold).

`dist` character, distribution name. Only distributions 'GEV' and 'Gumbel' are supported.

`prior` list of lists, prior distributions. For each parameter to be estimated, the prior is a list of the form pr=list(dist=..., par=...). See example below.

`SystErrorIndex` integer vector, length NROW(y). Index of systematic errors. Rows where SystErrorIndex==k are all affected by the same multiplicative error gamma_k, typically induced by the kth rating curve. SystErrorIndex==0 means no systematic error. Should only contain integer values between 0 and N_{systematic errors}.

`SystErrorPrior` list of lists, prior distribution for each systematic error. For instance for a systematic error in the range +/- 20%, you may use a Uniform between 0.8 and 1.2, or a triangular distribution with the same bounds and peak at 1.

`options` list, options, see details below.

`mcmcoptions` list, MCMC options, see details below.

`do.KS`, `do.MK`, `do.Pettitt` logical, perform KS/MK/Pettitt tests?

Details

The argument 'options' allows controlling various properties of the analysis and results. It is a list with the following components:

- FreqFormula, character, formula for computing nonexceedance frequency, see ?GetEmpFreq.
- pgrid, numeric vector, probabilities defining the x values where pdf f(x) and cdf F(x) are computed. These x values are quantiles from the estimated distribution with probabilities pgrid.
- Tgrid, numeric vector, return periods where quantile function q(T) is computed.
- IClevel, numeric, level of uncertainty interval.
- p2T, numeric, conversion factor between nonexceedance probability p and return period T. p=1-1/(p2T*T). Here p2T=1 in general since GEV/Gumbel are applied to annual maxima in general.
- invertT, logical, when invertT=TRUE, LARGE return periods correspond to SMALL data values. This is typically used for low-flow statistics. Unused here.
- splitZeros, logical, when splitZeros=TRUE zero and negative values are removed from the data y before estimating the distribution,and are used to estimate the probability of zeros p0. This is typically used for low-flow statistics to estimate the probability of zero streamflow. Unused here.
- lang, chanracter, language ('fr' or 'en').

- nsim, integer, number of replicated parameters representing uncertainty. Unused here (derives from mcmc options)

The argument 'mcmcoptions' is a list controlling MCMC properties:

- mult, numeric, see ?Metropolis_OAAT_adaptive
- eps, numeric, see ?Metropolis_OAAT_adaptive
- batch.length, integer, see ?Metropolis_OAAT_adaptive
- batch.n, integer, see ?Metropolis_OAAT_adaptive
- moverate.min, numeric, see ?Metropolis_OAAT_adaptive
- moverate.max, numeric, see ?Metropolis_OAAT_adaptive
- mult.down, numeric, see ?Metropolis_OAAT_adaptive
- mult.up, numeric, see ?Metropolis_OAAT_adaptive
- burn, numeric, burn-in factor, e.g. if burn=0.2 the first 20 percents of MCMC samples are discarded
- slim, integer, sliming factor, e.g. if slim=5 only one MCMC sample every 5 is kept (after burn-in)

Value

A list with the following components:

`dist`	character, estimated distribution.
`ok`	logical, did estimation succeed?
`err`	integer, error code (0 if ok).
`message`	error message.
`empirical`	data frame, sorted data and empirical estimates (nonexceedance frequency, return period and reduced variate). NOTE: interval data are replaced by a value randomly sampled from a GEV constrained in this interval. See ?GenerateWithinBounds.
`pcdf`	data frame, estimated pdf and cdf
`quantile`	data frame, estimated quantiles and uncertainty intervals
`par`	data frame, estimated parameters and uncertainty intervals
`KS`	list, result of the Kolmogorov-Smirnov test, see ?KS. NOTE: interval data are replaced by a value randomly sampled from a GEV constrained in this interval. See ?HBay_simGEV.
`MK`	list, result of the Mann-Kendall test, see ?MK. Same note as KS test.
`Pettitt`	list, result of the Pettitt test, see ?Pettitt. Same note as KS test.
`u`	list, parameter uncertainty in the form of a covariance matrix ($cov) and simulated parameter replicates ($sim). Also contains error-handling flags $ok, $err and $message.

Examples

```
set.seed(98765)
n=50;n_censored=30
y0=Generate('GEV',c(100,50,-0.2),n)
y=cbind(y0,y0)
# Mimics censoring between 0 and 300 for first n_censored years
y[1:n_censored,1][y0[1:n_censored]<300]=0
y[1:n_censored,2][y0[1:n_censored]<300]=300
plot(y[,1]);points(y[,2])
# Systematic errors
SystErrorIndex=c(rep(1,n_censored),rep(2,n-n_censored))
SystErrorPrior=list(list(dist="Triangle",par=c(1,0.7,1.3)),
                    list(dist="Triangle",par=c(1,0.95,1.05)))
# Priors on GEV parameters
prior=list(list(dist="FlatPrior",par=NULL),
           list(dist="FlatPrior",par=NULL),
           list(dist="Normal",par=c(0,0.25)))
# Handle MCMC options
# The values below aim at making this example fast to run.
# In practice, it is recommended to use the default values
# (batch.length=100,batch.n=100) or larger.
mcmcoptions=mcmcoptions_def
mcmcoptions$batch.length=25
mcmcoptions$batch.n=20
# Go!
H3=Hydro3_HBay(y=y,dist='GEV',prior=prior,
               SystErrorIndex=SystErrorIndex,
               SystErrorPrior=SystErrorPrior,
               mcmcoptions=mcmcoptions)
Hydro3_Plot(H3)
```

Hydro3_Plot *Hydro3 plot*

Description

Plot summarizing the results of Hydro3_Estimation()

Usage

```
Hydro3_Plot(
  H3,
  useU = FALSE,
  lwd = 2,
  cex.lab = 2,
  cex.axis = 1.3,
  pch = 19,
  col = "red"
)
```

Arguments

`H3`	list, resulting from a call to Hydro3_Estimation()
`useU`	logical, use reduced variate u rather than return period T in plots?
`lwd`, `cex.lab`, `cex.axis`, `pch`	numeric, graphical parameters, see ?graphics::par
`col`	character, graphical parameter (points color)

Value

nothing (just creates a plot)

Examples

```
y=stats::rnorm(50)
H3=Hydro3_Estimation(y,'Normal')
Hydro3_Plot(H3)
```

`Import_HBayConfig` *Import HBay Configuration folder*

Description

Imports configuration data as specified with HBay executable. Returns NULL if configuration folder is not found

Usage

```
Import_HBayConfig(path)
```

Arguments

`path`	character, path to configuration folder.

Value

A list with the following components (see ?Hydro3_HBay for details):

`y`	numeric matrix, data.
`dist`	character, distribution name.
`prior`	list of lists, prior distributions.
`SystErrorIndex`	integer vector, index of systematic errors.
`SystErrorPrior`	list of lists, prior distribution for each systematic error.
`options`	list, inference options.
`mcmcoptions`	list, MCMC options.
`year`	numeric vector, years.

Examples

```
config=Import_HBayConfig('path/to/config')
if(!is.null(config)){
  H3=Hydro3_HBay(y=config$y,dist=config$dist,prior=config$prior,
                 SystErrorIndex=config$SystErrorIndex,
                 SystErrorPrior=config$SystErrorPrior,
                 options=config$options,
                 mcmcoptions=config$mcmcoptions)
  Hydro3_Plot(H3)
}
```

KS *Kolmogorov-Smirnov Test*

Description

Applies a one-sample Kolmogorov-Smirnov test (see ?stats::ks.test)

Usage

```
KS(y, dist, par)
```

Arguments

y	numeric vector, data
dist	character, distribution name
par	numeric vector, parameter vector

Value

A list with the following components:

pval	numeric, p-value of the test
stat	numeric, test statistics
xtra	numeric, xtra information: empty for this test

Examples

```
y=stats::rnorm(20)
KS(y,'Normal',c(0,1))
KS(y,'Normal',c(1,1))
KS(y,'Gumbel',c(0,1))
```

`mcmcoptions_def` *Default MCMC options*

Description

A named list containing the default MCMC options. See ?Hydro3_Estimation for more details.

Usage

```
mcmcoptions_def
```

Format

An object of class `list` of length 10.

`Metropolis_OAAT` *One-At-A-Time Metropolis sampler*

Description

Performs nsim iterations of the OAAT Metropolis sampler (simulated vector is updated one component at a time). a.k.a block Metropolis sampler with blocks of length one. Sometimes also called 'Metropolis-within-Gibbs'.

Usage

```
Metropolis_OAAT(f, x0, nsim, sdjump, ...)
```

Arguments

`f`	function, log-pdf of the target distribution
`x0`	numeric vector, starting point
`nsim`	integer, number of simulations
`sdjump`	numeric vector, standard deviation of the Gaussian jump for each component
`...`	other arguments passed to f

Value

A list with the following components:

`x`	numeric matrix nsim*length(x0), MCMC simulations
`fx`	numeric vector, corresponding values f(x)
`moverate`	numeric vector, move rate associated with each component

Examples

```
# Bivariate target distribution: beta(0.8,0.4) X exp(1)
f=function(x){stats::dbeta(x[1],0.8,0.4,log=TRUE)+stats::dexp(x[2],log=TRUE)}
x0=c(0.5,2)
sdjump=c(0.5,1)
mcmc=Metropolis_OAAT(f,x0,1000,sdjump)
graphicalpar=par(mfrow=c(1,3))
plot(mcmc$x);hist(mcmc$x[,1]); hist(mcmc$x[,2])
par(graphicalpar)
```

`Metropolis_OAAT_adaptive` *Adaptive One-At-A-Time Metropolis sampler*

Description

Performs nsim iterations of the Adaptive version of the OAAT Metropolis sampler (see ?Metropolis_OAAT). Adaptation is performed by monitoring move rates every batch.length iterations, and increasing / decreasing the jump standard deviation if the move rate is not within specified bounds.

Usage

```
Metropolis_OAAT_adaptive(
  f,
  x0,
  sdjump,
  ...,
  batch.length = 100,
  batch.n = 100,
  moverate.min = 0.1,
  moverate.max = 0.5,
  mult.down = 0.9,
  mult.up = 1.1
)
```

Arguments

`f`	function, log-pdf of the target distribution
`x0`	numeric vector, starting point
`sdjump`	numeric vector, initial standard deviation of the Gaussian jump for each component
`...`	other arguments passed to f
`batch.length`	integer, length of each non-adaptive batch
`batch.n`	integer, number of batches (= adaptation period). Total number of simulations is nsim=batch.n*batch.length

moverate.min	numeric in (0;1), lower bound for the desired move rate interval
moverate.max	numeric in (0;1), upper bound for the desired move rate interval
mult.down	numeric in (0;1), multiplication factor used to decrease jump size when move rate is too low.
mult.up	numeric (>1, avoid 1/mult.down) multiplication factor used to increase jump size when move rate is too high.

Value

A list with the following components:

x	numeric matrix nsim*length(x0), MCMC simulations
fx	numeric vector, corresponding values f(x)

Examples

```
# Bivariate target distribution: beta(0.8,0.4) X exp(1)
f=function(x){stats::dbeta(x[1],0.8,0.4,log=TRUE)+stats::dexp(x[2],log=TRUE)}
x0=c(0.5,2)
sdjump=c(0.5,1)
mcmc=Metropolis_OAAT_adaptive(f,x0,sdjump)
graphicalpar=par(mfrow=c(1,3))
plot(mcmc$x);hist(mcmc$x[,1]); hist(mcmc$x[,2])
par(graphicalpar)
```

Metropolis_OAAT_jump *One-At-A-Time Metropolis sampler*

Description

Performs a single iteration of the OAAT Metropolis sampler (simulated vector is updated one component at a time). a.k.a block Metropolis sampler with blocks of length one. Sometimes also called 'Metropolis-within-Gibbs'.

Usage

```
Metropolis_OAAT_jump(f, x0, fx0, sdjump, ...)
```

Arguments

f	function, log-pdf of the target distribution
x0	numeric vector, starting point
fx0	numeric, f(x0)
sdjump	numeric vector, standard deviation of the Gaussian jump for each component
...	other arguments passed to f

Value

A list with the following components:

x	numeric vector, updated point after the iteration
fx	numeric, updated value f(x)
move	logical vector, TRUE for components of the vector x that changed

Examples

```
# Bivariate target distribution: beta(2,10) X exp(1)
f=function(x){stats::dbeta(x[1],2,10,log=TRUE)+stats::dexp(x[2],log=TRUE)}
x0=c(0.5,0.5)
fx0=f(x0)
sdjump=c(0.1,0.1)
Metropolis_OAAT_jump(f,x0,fx0,sdjump)
```

MK *Mann-Kemdall Test*

Description

Applies the Mann-Kendall trend test

Usage

```
MK(y)
```

Arguments

y	numeric vector, data

Value

A list with the following components:

pval	numeric, p-value of the test
stat	numeric, test statistics
xtra	numeric, xtra information: empty for this test

Examples

```
y=stats::rnorm(50)
MK(y)
y=y+0.1*(1:length(y))
MK(y)
```

`options_def` *Default estimation options*

Description

A named list containing the default estimation options. See ?Hydro3_Estimation for more details.

Usage

```
options_def
```

Format

An object of class `list` of length 9.

`Pettitt` *Pettitt Test*

Description

Applies the Pettitt step-change test

Usage

```
Pettitt(y)
```

Arguments

`y`	numeric vector, data

Value

A list with the following components:

`pval`	numeric, p-value of the test
`stat`	numeric, test statistics
`xtra`	numeric, xtra information: position of the step change

Examples

```
y=stats::rnorm(50)
Pettitt(y)
y[26:50]=y[26:50]+2
Pettitt(y)
```

Index