Package: Renext (via r-universe)

August 24, 2024

Type Package

Title Renewal Method for Extreme Values Extrapolation

Version 3.1-4

Date 2023-08-29

Author Yves Deville <deville.yves@alpestat.com>, Lise Bardet <lise.bardet@irsn.fr>

Maintainer Yann Richet <yann.richet@irsn.fr>

URL https://github.com/IRSN/Renext, https://irsn.github.io/Renext/

Depends R (>= 2.8.0), stats, graphics, evd

Imports numDeriv, splines, methods

Suggests MASS, ismev, XML

Description Peaks Over Threshold (POT) or 'methode du renouvellement'.
The distribution for the excesses can be chosen, and
heterogeneous data (including historical data or block data)
can be used in a Maximum-Likelihood framework.

License GPL (>= 2)

LazyData yes

Repository https://irsn.r-universe.dev

RemoteUrl https://github.com/irsn/renext

RemoteRef HEAD

RemoteSha d1df900464ef9894bfbf52145012b317d9d84a8d

Contents

`Renext-package` *Renewal Method for Extreme Values Extrapolation*

Description

This package proposes fits and diagnostics for the so-called *méthode du renouvellement*, an alternative to other "Peaks Over Threshold" (POT) methods. The *méthode du renouvellement* generalises the classical POT by allowing the excesses over the threshold to follow a probability distribution which can differ from the Generalised Pareto Distribution (GPD). Weibull or gamma excesses are sometimes preferred to GPD excesses. The special case of exponential excesses (which falls in the three families: GPD, Weibull and gamma) has a special interest since it allows exact inference for the (scalar) parameter and for the quantiles form OT data (only).

The package allows the joint use of possibly three kinds of data or information. The first kind is *classical excesses*, or *"OT data"*. It can be completed with two kinds of data resulting from a temporal aggregation, as is often the case for *historical data*. Both types are optional, and concern periods or *blocks* that must not overlap nor cross the OT period.

- *MAX data* correspond to the case where one knows the r largest observations over each block. The number r may vary across blocks. This kind of data is often called 'r largest', or "r Largest Order Statistics" (r LOS).
- *OTS data* (for OT Supplementary data) correspond to the case where one knows for each block b all the observations that exceeded a threshold u_b which is greater (usually much greater) than the main threshold u. The number r_b of such observations can be zero, in which case we may say that u_b is an *unobserved level*. A threshold u_b is sometimes called a *perception threshold*.

Historical data are often available in hydrology (e.g. for river flood discharges, for sea-levels or sea surges) and can concern large periods such as past centuries. An unobserved level can typically be related to a material benchmark.

Maximum likelihood estimation is made possible in this context of heterogeneous data. Inference is based on the asymptotic normality of parameter vector estimate and on linearisation ("delta method") for quantiles or parameter functions.

The package allows the use of "marked-process observations" data (datetime of event and level) where an interevent analysis can be useful. It also allows the event dates to be unknown and replaced by a much broader *block* indication, e.g. a year number. The key point is then that the "effective duration" (total duration of observation periods) is known. Event counts for blocks can be used to check the assumption of Poisson-distributed events.

The package development was initiated, directed and financed by the french *Institut de Radioprotection et de Sûreté Nucléaire* (IRSN). The package is a non-academic tool designed for applied analysis on case studies and investigations or comparisons on classical probabilistic models.

Details

The DESCRIPTION file: This package was not yet installed at build time.

Index: This package was not yet installed at build time.

This package contains a function `Renouv` to fit "renouvellement" models.

Author(s)

Yves Deville <deville.yves@alpestat.com>, Lise Bardet <lise.bardet@irsn.fr>

Maintainer: Yann Richet <yann.richet@irsn.fr>

References

- Miquel J. (1984) *Guide pratique d'estimation des probabilités de crues*, Eyrolles (coll. EDF DER).
- Coles S. (2001) *Introduction to Statistical Modelling of Extremes Values*, Springer.
- Embrechts P., Klüppelberg C. and Mikosch T. (1997) *Modelling Extremal Events for Insurance and Finance*. Springer.

See Also

The CRAN packages **evd**, **ismev**, **extRemes**, **POT**.

Examples

```
## 'Garonne' data set
summary(Garonne)
plot(Garonne)

## Weibull excesses
fG <- Renouv(x = Garonne,
             threshold = 3000,
             distname.y = "weibull",
             main = "Weibull fit for 'Garonne'")

coef(fG)
vcov(fG)
summary(fG)
```

```
logLik(fG)
## Re-plot if needed
plot(fG)

## Classical 'predict' method with usual formal args
predict(fG, newdata = c(100, 150, 200), level = c(0.8, 0.9))
```

`anova.Renouv` *Compute an analysis of deviance table for two nested Renouv objects*

Description

Compute an analysis of deviance table for two nested Renouv objects

Usage

```
   ## S3 method for class 'Renouv'
anova(object, object1, trace = 1L, ...)
```

Arguments

`object`	A Renouv model as fitted with `Renouv`.
`object1`	A Renouv object such that `object` is nested in `object1`.
`trace`	Level of verbosity. The value `0` prints nothing.
`...`	Not used yet.

Details

Of special interest is the case when the distribution of the excesses used in `object` is exponential while `object1` uses a two-parameters alternative in the GPD family. We know then that the convergence to the asymptotic distribution is slow, and a numerical approximation of the true distribution of the test statistic is used when possible, i.e. when the objects do not use MAX or OTS data and the number of exceedances is between 8 and 500.

Value

An object of class `"anova"` inheriting from class `"data.frame"`.

Note

The deviance of the models can not be interpreted: only the difference of the deviance is meaningful.

See Also

`anova`, `LRExp.test`.

Examples

```
## test using historical data
fit1Exp <- Renouv(Garonne,  distname.y = "exponential", plot = FALSE)
fit1GPD <- Renouv(Garonne, distname.y = "GPD", plot = FALSE)
anova(fit1Exp, fit1GPD)

## test without using historical data
x <- Garonne$OTdata$Flow
dur <- Garonne$OTinfo$effDuration

fit2Exp <- Renouv(x,  threshold = 2700,  effDuration = dur,
                  distname.y = "exponential", plot = FALSE)
fit2GPD <- Renouv(x, threshold = 2700, effDuration = dur,
                  distname.y = "GPD", plot = FALSE)
anova(fit2Exp, fit2GPD)
```

`barplotRenouv` *Barplot for Renouv "Over Threshold" counts*

Description

Barplot for "Over Threshold" counts in time blocks (usually years)

Usage

```
barplotRenouv(data,
              blockname = colnames(data)[1],
              varname = colnames(data)[2],
              threshold = quantile(data[, varname], 0.2),
              na.block = NULL,
              plot = TRUE,
              main = NULL, xlab = NULL, ylab = NULL,
              mono = FALSE,
              prob.theo = 0.999,
              ...)
```

Arguments

`data`	A dataframe object containing the variables.
`blockname`	Name of the "block" variable (column in `data`). This variable should contain integers, or be of class "factor", but with integer values such as year numbers.
`varname`	Name of the variable (e.g. `"Surge"`).
`threshold`	Only obs for which the variable exceeds `threshold` will be taken into account.
`na.block`	Values of blocks containing missing values. See the Details section.
`plot`	If `FALSE` tests are computed without producing any plot.
`main`	Character for main title or `NULL` in which case a default main title is used.

`xlab`	Character for x axis label or `NULL` in which case a default lab is used.
`ylab`	Character for y axis or `NULL` in which case a default lab is used.
`mono`	If `FALSE` barplot will have colors, else greyscale will be used.
`prob.theo`	The total theoretical probability corresponding to the plotted (theoretical) bars.
`...`	Further args to be passed to `barplot`.

Details

Blocks described in the `na.block` are omitted in the determination of counts. The object given in the `na.block` is coerced to character and the same is done for values of `block` before comparing them to the `na.block` values. If `block` variable is of class factor with levels representing years (e.g. 1980, 1981, etc.) missing blocks can be specified either as `c("1980", "1981")` or as numeric `c(1980, 1981)`.

For the chi-square test, counts for neighbouring frequency classes are grouped in order to reach a minimum frequency of `5` in each group. E.g. if we expect respectively `1.0`, `3.8` and `7.0` blocks with frequency `0`, `1` and `2` for events, the three counts are grouped in one group with frequency `1.0+3.8+7.0=11.8`. Note that this strategy of grouping is not unique and is likely to weaken the power of the test. Before grouping, the higher class theoretical probability is computed as the probability to obtain a count equal to or greater than the max value.

Value

A list with the following objects.

`freq`	frequency table (matrix) giving observed and theoretical (Poisson) frequencies as well as a group number for the chi-square test.
`overdispersion`	the overdispersion coefficient (variance/mean ratio).
`disp.test`	a list giving results of the (over)dispersion test. See the reference Yagouti and al. in the **References** section.
`chisq.test`	a list giving results for the chis-square test of goodness-of-fit to the Poisson distribution.
`tests`	a matrix with the two tests displayed in two rows.

For both tests, the statistic follows a chi-square distribution under the null hypothesis . The list of results contains the statistic `statistic`, the number of degrees of freedom `df` and the p-value `p.value`.

Note

The two tests: (over-)dispersion and chi-square have *one-sided* (upper tail) p-value. In other words, we do not intend to reject when statistics take "abnormally small" values, but only when abnormally large values are met.

Author(s)

Yves Deville

References

See Yagouti A., Abi-Zeid I., Ouarda, T.B.M.J. and B. Bobée (2001), Revue de processus ponctuels et synthèse de tests statistiques pour le choix d'un type de processus *Revue des Sciences de l'Eau*, **1**, pp. 323-361.

See Also

`plot.Rendata`

Examples

```
## na.block influence for Brest data
opar <- par(mfrow = c(2, 2))

bp1 <- barplotRenouv(data = Brest.years, threshold = 30,
         main = "missing periods ignored")
bp2 <- barplotRenouv(data = Brest.years, threshold = 30,
         na.block = 1992, main = "1992 missing")
bp3 <- barplotRenouv(data = Brest.years, threshold = 30,
         na.block = 1991:1993, main ="1991:1993 missing")
bp4 <- barplotRenouv(data = Brest.years, threshold = 30,
         na.block = Brest.years.missing, main = "all missing periods")

par(opar)

## threshold influence
opar <- par(mfrow = c(2,2))

thresh <- c(30, 35, 40, 50)

for (i in 1:length(thresh)) {
  bp  <- barplotRenouv(data = Brest.years, threshold = thresh[i],
                   na.block = Brest.years.missing,
                   main = paste("threshold =", thresh[i], "cm at Brest"))
}
par(opar)
```

`Brest` *Surge heights at Brest*

Description

Surge heights near high tide at Brest tide gauge station (France), detailed version

Usage

```
Brest
```

Format

The format is: List of 5

- `$info` : List of 6
  - `$name` : chr `"Brest"`
  - `$shortLab` : chr `"Surge Heights at Brest (France)"`
  - `$longLab` : chr `"Surge Heights near high tide, Brest (France)"`
  - `$varName` : chr `"Surge"`
  - `$varShortLab` : chr `"Surge"`
  - `$varUnit` : chr `"cm"`
- `$describe` : chr "High tide sea surge over 30 cm at Brest (France)..."
- `$OTinfo` : List of 4
  - `$start` : chr POSIXct[1:1], format: `"1846-01-01"`
  - `$end` : chr POSIXct[1:1], format: `"2009-01-01"`
  - `$effDuration` : num `148`
  - `$threshold` : num `30`
- `$OTdata` : 'data.frame': 1289 obs. of 2 variables:
  - `$date` : POSIXct[1:1289], format: `"1846-01-14" "1846-01-21"` ...
  - `$Surge` : num [1:1289] `36 60 46 40 33` ...
- `$OTmissing` : 'data.frame': 43 obs. of 3 variables:
  - `$start` : POSIXct[1:43], format: `"1846-01-01" "1847-01-01"` ...
  - `$end` : POSIXct[1:43], format: `"1846-01-04" "1847-01-21"` ...
  - `$comment` : chr [1:43] `"" "" "" ""` ...

  - `attr(*, "class")= chr "Rendata"`

Details

Data are provided as a list.

- `info` gives general information about the data
- `OTinfo` gives general information about the Over the Threshold part of data. The effective duration (`effDuration` element) is the total duration for the periods with effective measurements.
- `OTdata` give OT measurements
- `OTmissing` gives start and end of the missing periods for OT measurements.

Data come from hourly sea levels measured and predicted by the french *Service Hydrogéographique et Océanographique de la Marine* (SHOM). Observed sea levels are available as *REFMAR* data at the url `https://data.shom.fr/`. Data were processed (declustered) by IRSN in order to provide a series of independent surge heights at high tide. Surge height at high tide is defined as the difference between the observed and the predicted maximal sea levels near high tide. A correction was applied to account for trend in the sea-level over the observation period.

The effective duration given in years is defined up to a small fraction of year due to leap years and leap seconds.

Source

https://data.shom.fr/

Examples

```
str(Brest)
Brest$OTinfo$start
plot(Brest)
```

`Brest.years` *Surge heights at Brest partial data*

Description

Surge heights at Brest (France)

Usage

```
Brest.years
```

Format

A data frame with 954 observations on the following 2 variables.

`year` Year e.g; 1980

`Surge` Surge heights above the threshold of 30 cm.

Details

These data are a simplified version of `Brest`. For each surge event only the year is retained as timestamp. Years with missing periods are available as a vector `Brest.years.missing`.

This dataset is useful for testing since similar data are sometimes met in the analyses.

Examples

```
names(Brest.years)
```

`Brest.years.missing` *Years with missing periods in 'Brest.year' dataset*

Description

Years with missing periods in the 'Brest.years' dataset

Usage

```
Brest.years.missing
```

Format

The format is: int [1:49] 1846 1847 1852 1857 1858 1859 1860 1861 1862 1863 ...

Details

Vector of years containing missing periods in the `Brest.years` dataset. This years should be ignored when computing yearly statistics such as event rates, since time records are lost.

Examples

```
print(Brest.years.missing)
```

`CV2` *Squared Coefficient of Variation*

Description

Squared Coefficient of Variation.

Usage

```
CV2(x)
```

Arguments

`x`	Numeric vector or matrix.

Details

Compute the squared Coefficient of Variation of one or several samples provided as a numeric vector or matrix.

Value

Numeric vector of the squared coefficients of variation.

Note

The squared coefficient of variation is the ratio $S^2/\bar{X}^2$ where $\bar{X}$ and S^2 are the sample mean and the sample variance. The variance is computed using the sample size n as denominator, rather than the usual $n-1$.

Examples

```
n <- 30; nSamp <- 500
X <- matrix(rexp(n * nSamp), nrow= nSamp, ncol = n)
W <- CV2(X)
plot(density(W), main = "CV2 of exponential samples")
```

`CV2.test` *CV2 test of exponentiality*

Description

Test of exponentiality based on the squared coefficient of variation.

Usage

```
CV2.test(x, method = c("num", "sim", "asymp"), nSamp = 15000)
```

Arguments

`x`	Numeric vector giving the sample.
`method`	Method used to compute the p-value. Can be `"asymp"`, `"num"` or `"sim"` as in `LRExp.test`.
`nSamp`	Number of samples used to compute the p-value when `method` is `"sim"`.

Details

The distribution of CV^2 is that of *Greenwood's statistic* up to normalising constants. It approximately normal with expectation 1 and standard deviation $2/\sqrt{n}$ for a large sample size `n`. Yet the convergence to the normal is known to be *very slow*.

Value

A list of test results.

`statistic, p.value`	The test statistic, i.e. the squared coefficient of variation CV^2 and the p-value.
`df`	The sample size.
`method`	Description of the test method.

Note

This test is sometimes referred to as *Wilk's exponentiality test* or as *WE1 test*. It works quite well for a Lomax alternative (i.e. GPD with shape $\xi > 0$), and hence can be compared to Jackson's test and the Likelihood-Ratio (LR) test of exponentiality. However, this test has lower power that of the two others while having a comparable computation cost due to the evaluation of the Greenwood's statistic distribution.

Author(s)

Yves Deville

References

S. Ascher (1990) "A Survey of Tests for Exponentiality" *Commun. Statist. Theory Methods*, 19(5), pp. 1811-1525.

See Also

The function `CV2` that computes the statistic and `LRExp.test` or `Jackson.test` for functions implementing comparable tests or exponentiality with the same arguments.

Examples

```
n <- 30; nSamp <- 500
X <- matrix(rexp(n * nSamp), nrow = nSamp, ncol = n)
pVals <- apply(X, 1, function(x) CV2.test(x)$p.value)
plot(pVals)  ## should be uniform on (0, 1)
```

`Dunkerque` *Surge heights at Dunkerque*

Description

Surge heights near high tide at Dunkerque tide gauge station (France)

Usage

```
Dunkerque
```

Format

The format is: List of 7

- `$info` : List of 6
  - `$name` : chr `"Dunkerque"`
  - `$shortLab` : chr `"Surge Heights at Dunkerque (France)"`
  - `$longLab` : chr `"Surge Heights near high tide, Dunkerque (France)"`

  - $varName : chr "Surge"
  - $varShortLab : chr "Surge"
  - $varUnit : chr "cm"
- $describe : chr "High tide sea surge over 30 cm at Dunkerque... "
- $OTinfo : List of 4
  - $start : POSIXct[1:1], format: "1956-01-01"
  - $end : POSIXct[1:1], format: "2009-01-01"
  - $effDuration: num 38.8
  - $threshold : num "30"
- $OTdata : 'data.frame': 740 obs. of 3 variables:
  - $date : POSIXct[1:740], format: "1956-11-27" "1956-12-03" ...
  - $Surge : num [1:740] 67.9 30.9 51.8 30.8 39.8 ...
  - $comment : chr [1:740] "" "" "" "" ...
- $OTmissing : 'data.frame': 83 obs. of 3 variables:
  - $start : POSIXct[1:83], format: "1956-01-01" "1956-08-08" ...
  - $end : POSIXct[1:83], format: "1956-06-07" "1956-11-03" ...
  - $comment: chr [1:83] "" "" "" "" ...
- $MAXinfo : 'data.frame' : 1 obs. of 3 variables:
  - $start : POSIXct[1:1], format: "1706-01-01"
  - $end : POSIXct[1:1], format: "1956-01-01"
  - $duration : num 250
- $MAXdata :'data.frame': 1 obs. of 4 variables:
  - $block : int 1
  - $date : POSIXct[1:1], format: "1953-02-01"
  - $Surge : num 213
  - $comment : chr "1"

  - attr(*, "class")= chr "Rendata"

Details

See Brest and Garonne datasets with the same list structure.

An 'historical' surge of 213 cm was observed on 1953-02-01 and is considered by experts as having a return period of 250 years.

Examples

```
Dunkerque$info
plot(Dunkerque)
```

EM.mixexp — *Expectation-Maximisation for a mixture of exponential distributions*

Description

Experimental function for Expectation-Maximisation (EM) estimation

Usage

```
EM.mixexp(x, m = 2)
```

Arguments

`x`	Sample vector with values `>0`.
`m`	Number of mixture components.

Details

The EM algorithm is very simple for exponential mixtures (as well as for many other mixture models).

According to a general feature of EM, this iterative method leads to successive estimates with increasing likelihood but which may converge to a local maximum of the likelihood.

Value

List with

`estimate`	Estimated values as a named vector.
`logL`	Vector giving the log-likelihood for successive iterations.
`Alpha`	Matrix with `m` columns giving probability weights for successive iterations. Row with number `it` contains the `m` probabilities at iteration `it`.
`Theta`	Matrix with `m` columns giving the estimates of the `m` expectations for the successive iterations

Note

The estimation is done for expectation (inverse rates) but the `estimate` vector in the result contains rates for compatibility reasons (e.g with exponential).

Author(s)

Yves Deville

See Also

`mom.mixexp2` and `ini.mixexp2` for "cheap" estimators when `m = 2`.

Examples

```
set.seed(1234)
x <- rmixexp2(n = 100, prob1 = 0.5, rate2 = 4)
EM.mixexp(x) -> res
res$estimate
matplot(res$Theta, type = "l", lwd = 2,
        xlab = "iteration", ylab = "theta",
        main = "exponential inverse rates")
```

---

expplot *Classical "exponential distribution" plot*

Description

Plot a vector using "exponential distribution" scales

Usage

```
expplot(x,
        plot.pos = "exp",
        rate = NULL,
        labels = NULL,
        mono = TRUE,
        ...)
```

Arguments

`x`	The vector to be plotted.
`plot.pos`	Plotting position for points: either "exp" for *expected* ranks or "med" for a *median* rank approximation (see **Details** below).
`rate`	Rate parameter for one or several "exponential distribution" lines to be plotted
`labels`	Text to display in legend when "exponential distribution" lines are specified
`mono`	Monochrome graph?
`...`	Arguments to be passed to `plot`.

Details

This plot shows $-\log[1 - F(x)]$ against x where $F(x)$ at point i is taken as $i/(n + 1)$ if `plot.pos` is `"exp"`, or as the "median rank" approximation $(i - 0.3)/(n + 0.4)$ if `plot.pos` is `"med"`.

If the data in `x` is a sample from an exponential distribution, the points should be roughly aligned. However the largest order statistics have high sampling dispersion.

Note

The log scale for y is emulated via the construction of suitable graduations. So be careful when adding graphical material (points, etc) to this graph with functions of the "add to plot" family (`points`, `lines`, ...).

The ML estimate of the `rate` parameter is the inverse of the sample mean.

Author(s)

Yves Deville

See Also

The `weibplot` function for a classical "Weibull" plot. The `interevt` is useful to compute interevents (or "interarrivals") that should follow an exponential distribution in the homogeneous Poisson process context.

Examples

```
x <- rexp(200)
expplot(x, rate = 1/mean(x), labels = "fitted")
```

`fgamma` *ML estimation of the Gamma distribution*

Description

Fast Maximum Likelihood estimation of the Gamma distribution.

Usage

```
fgamma(x, check.loglik = FALSE)
```

Arguments

`x`	Sample vector to be fitted. Should contain only positive non-NA values.
`check.loglik`	If `TRUE`, the log-likelihood is recomputed using `dgamma` function with `log = TRUE`. The result is returned as a list element.

Details

The likelihood is concentrated with respect to the scale parameter. The concentrated log-likelihood is a strictly concave function of the shape parameter which can easily maximised numerically.

Value

A list with the following elements

`estimate`	Parameter ML estimates.
`sd`	Vector of (asymptotic) standard deviations for the estimates.
`loglik`	The maximised log-likelihood.
`check.loglik`	The checked log-likelihood.
`cov`	The (asymptotic) covariance matrix computed from theoretical or observed information matrix.
`info`	The information matrix.

Note

The distribution is fitted by using the `scale` parameter rather than `rate` (inverse of `scale`).

Author(s)

Yves Deville

See Also

`GammaDist` in the **stats** package.

Examples

```
set.seed(9876)
alpha <- 0.06
beta <- rexp(1)
n <- 30
x <- rgamma(n, shape = alpha, scale = beta)
fit <- fgamma(x, check.loglik = TRUE)

## compare with MASS results
if (require(MASS)) {
   fit.MASS <- fitdistr(x, densfun = "gamma")
   rate <- 1 / fit$estimate["scale"]
   est <- c(fit$estimate, rate = rate)
   der <- rate * rate ## derivative of rate w.r.t scale
   sdest <- c(fit$sd, rate = der * fit$sd["scale"])
   tab <- rbind(sprintf(" %10.8f ", est),
                sprintf("(%10.8f)", sdest))
   colnames(tab) <- c("shape", "scale", "rate")
   rownames(tab) <- c("est", "sd")
   noquote(tab)
}
```

`fGEV.MAX`	*Fit a GEV distribution from block maxima or r largest order statistics using an aggregated Renewal POT process*

Description

Fit a GEV distribution from block maxima or r largest order statistics using an aggregated Renewal POT process.

Usage

```
fGEV.MAX(MAX.data, MAX.effDuration,
         MAX = NULL,
         control = list(maxit = 300, fnscale = -1),
         scaleData = TRUE,
         cov = TRUE,
         info.observed = TRUE,
         trace = 0)
```

Arguments

`MAX.data`	A list of block maxima or r largest statistics as in `Renouv`.
`MAX.effDuration`	A vector of durations as in `Renouv`. *The durations must be identical* in order to have a common GEV distribution for the maxima.
`MAX`	A compact representation of the needed data as a list. This is typically created by using the (non exported) `Renext:::makeMAXdata` function.
`control`	List to control the optimisation in `optim`.
`scaleData`	Logical. If `TRUE`, the data in `MAX.data` are scaled before being used in the likelihood. The scaling operation is carried on the excesses (observations minus threshold).
`cov`	Logical. If `TRUE` the standard deviation and the covariance matrix of the estimates are computed and returned as `sd` and `cov` elements of the list. However if the estimated shape parameter is < -0.5 the two elements are filled with `NA` because the regularity conditions can not be thought of as valid.
`info.observed`	Logical. If `TRUE` the covariance is computed from the *observed* information matrix. If `FALSE`, the *expected* information matrix is used instead. This is only possible for block maxima data, i.e. when all the blocks contain only one observation. The computation relies on the formula given by Prescott and Walden. Note that the default value differs from that of the functions `fGPD`, `flomax` and `fmaxlo`, for historical reasons.
`trace`	Integer level of verbosity during execution. With the value `0`, nothing is printed.

Details

The data are assumed to provide maxima or r largest statistics arising from an aggregated renewal POT model with unknown event rate λ and unknown two-parameter Generalised Pareto Distribution for the excesses. A threshold u is fixed below the given data and the three unknown parameters `lambda`, `scale` and `shape` of the POT model are found by maximising the likelihood. Then a vector of the three parameters for the GEV distribution is computed by transformation. The covariance matrix and standard deviations are computed as well using the jacobian matrix of the transformation.

The maximisation is for the log-likelihood with the rate `lambda` concentrated out, so it is a two-parameter optimisation.

Value

A list

`estimate`	Named vector of the estimated parameters for the GEV distribution of maxima.
`opt`	Result of the optimisation.
`loglik`	Identical to `opt$value`. This is the maximised log-likelihood for the renewal POT model.
`sd`	Standard deviation of the estimates (approximation based on the ML theory).
`cov`	Covariance matrix of the estimates (approximation based on the ML theory).

Caution

Whatever be the data, the log-likelihood is infinite (hence maximal) for any vector of GEV parameters with shape < -1 and postive scale. Hence the log-likelihood should be maximised with a constraint on the shape, while an *unconstrained* optimisation is used here. In practice, for numerical reasons, the estimate usually remains inside the `shape > -1` region. *An estimation leading to shape* < -1 *must be considered as meaningless*. An estimation with shape < -0.5 should be considered with care.

Note

This function could get more arguments in the future.

Author(s)

Yves Deville

References

The *Renext Computing Details* document.

Prescott P. and Walden A.T. (1980) Maximum Likelihood Estimation of the Parameters of the Generalized Extreme-Value Distribution. *Biometrika* **67**(3), 723-724.

See Also

`Renouv`.

Examples

```
##====================================================================
## block maxima: simulated data and comparison with  the 'fgev'
## function from the 'evd' package
##====================================================================
set.seed(1234)
u <- 10
nBlocks <- 30
nSim <- 100   ## number of samples
Par <- array(NA, dim = c(nSim, 3, 2),
             dimnames = list(NULL, c("loc", "scale", "shape"), c("MAX", "evd")))
LL <- array(NA, dim = c(nSim, 2),
            dimnames = list(NULL, c("MAX", "evd")))

for (i in 1:nSim) {
  rd <- rRendata(threshold = u,
                 effDuration = 1,
                 lambda = 12,
                 MAX.effDuration = rep(1, nBlocks),
                 MAX.r = rep(1, nBlocks),
                 distname.y = "exp", par.y = c(rate = 1 / 100))

  MAX <- Renext:::makeMAXdata(rd)
  fit.MAX <- fGEV.MAX(MAX = MAX)
  fit.evd <- fgev(x = unlist(MAX$data))
  Par[i, , "MAX"] <- fit.MAX$estimate
  Par[i, , "evd"] <- fit.evd$estimate
  LL[i, "MAX"] <- fit.MAX$loglik
  LL[i, "evd"] <- logLik(fit.evd)
}

##====================================================================
## r largest: use 'ismev::rlarg.fit' on the venice data set.
## NB 'venice' is taken from the 'evd' package here.
##====================================================================
## Not run:
require(ismev);
fit1 <- ismev::rlarg.fit(venice)

## transform data: each row is a block
MAX.data <- as.list(as.data.frame(t(venice)))
## remove the NA imposed by the rectangular matrix format
MAX.data <- lapply(MAX.data, function(x) x[!is.na(x)])
MAX.effDuration <- rep(1, length(MAX.data))

fit2 <- fGEV.MAX(MAX.data = MAX.data,
                 MAX.effDuration = MAX.effDuration)

## estimates
est <- cbind(ismev = fit1$mle, RenextLab = fit2$estimate)
print(est)
# covariance
```

```
covs <- array(dim = c(2, 3, 3),
              dimnames = list(c("ismev", "RenextLab"),
                colnames(fit2$cov), colnames(fit2$cov)))

covs["ismev", , ] <- fit1$cov
covs["RenextLab", , ] <- fit2$cov
print(covs)

## End(Not run)
```

---

fGPD *Fit a two-parameters Generalised Pareto Distribution from a sample*

Description

Fit a two-parameters Generalised Pareto Distribution from a sample.

Usage

```
fGPD(x,
     info.observed = TRUE,
     shapeMin = -0.8,
     dCV = 1e-04,
     cov = TRUE,
     trace = 0)
```

Arguments

`x`	The sample data vector.
`info.observed`	Logical. The observed information matrix is used when `TRUE` and the expected information matrix is used when `FALSE`.
`shapeMin`	Lower bound on the shape parameter. This must be `>= -1.0` since otherwise the ML estimate is obtained with the `scale` parameter equal to `max(x)`.
`dCV`	Half-length of a small interval centered on 1.0. When the Coefficient of Variation (CV) falls in this interval, an exponential distribution is fitted, see **Details**.
`cov`	Logical. If `FALSE`, a minimal estimation is performed with no covariance matrix or derivative returned. This can be useful when a large number of ML estimations are required, e.g. to sample from a likelihood ratio.
`trace`	Integer level of verbosity. The value `0` prints nothing.

Details

This function mainly relies on the `flomax` and `fmaxlo` functions. When `CV` is larger than `1.0` + `dCV`, a Lomax distribution is fitted by Maximum Likelihood using a concentrated log-likelihood. When instead `CV` is smaller than `1.0 - dCV`, a maxlo distribution is fitted. Finally, when `CV -1.0` has absolute value <= dCV, an exponential distribution is fitted. In all cases, the result is translated into a parameter vector for the GPD.

Note that when CV is close to 1.0, fitting a Lomax or a maxlo distribution can lead to problems because the estimated values of the shape and scale parameter are large, and because the concentrated log-likelihood is a flat function of the scale parameter.

Value

A list

estimate	Vector containing the estimated values of the unknown parameters.
CV	The coefficient of variation of x computed using length(x) as denominator in the variance estimation.
logLik, dlogLik	The maximised value of the log-likelihood and the vector of its first order derivatives, which should be close to zero.
sd, cov	Vector of approximated standard deviations and covariance matrix for the estimated parameters. These are based on the inversion of expected information matrix.
sd, cov	Vector of standard deviations and covariance matrix for the estimates if the cov formal is TRUE.
cvg	Logical. Was convergence reached? This logical flag is set to TRUE and remains for compatibility reasons.

Note

It may happen that the estimated shape parameter is < -0.5, in which case the expected information matrix can not be computed, nor does the covariance matrix and standard deviations. In this case, the cov and sd objects contain NA values. This problem can arise also when the shape parameter is greater than but close to the value -0.5. Even when info.observed is TRUE, the information matrix, covariance and standard deviations are set to NA.

When the true (unknown) value is is < -0.5, the regularity conditions required in the ML approximated inference do not hold.

The default value of info.observed was set to TRUE from version 3.0-1 because standard deviations obtained with this choice are usually better.

Author(s)

Yves Deville

References

See the *Renext Computing Details* document.

See Also

The fmaxlo and flomax functions.

Examples

```
## Not run:
set.seed(123456)
n <- 500
ns <- 1000
xi <- runif(ns, min = -0.5, max = 0.5)
X <- matrix(nrow = n, ncol = ns)

for (i in 1:length(xi)) {
  Xi <- rgpd(n, scale = 1, shape = xi[i])
  X[ , i] <- Xi
  res1 <- fGPD(Xi)
  res2 <- try(fpot(Xi, threshold = 0.0))
  if (inherits(res2, "try-error")) {
    cat(res2, "\n")
    break
  }
  logLik1 <- res1$loglik; logLik2 <- logLik(res2)
  if (abs(logLik1 - logLik2) > 0.001) {
    cat(sprintf("i = %d, xi = %7.4f\n", i, xi[i]))
    mat <- rbind(c(res1$estimate[1:2], logLik = logLik1),
                 c(res2$estimate[1:2], logLik = logLik2))
    rownames(mat) <- c("fGPD", "fpot")
    print(mat)
  }
}

## End(Not run)
```

flomax *ML estimation of the Lomax distribution*

Description

Fast Maximum Likelihood estimation of the Lomax distribution.

Usage

```
flomax(x,
       info.observed = TRUE,
       plot = FALSE,
       scaleData = TRUE,
       cov = TRUE)
```

Arguments

`x`	Sample vector to be fitted. Should contain only positive non-NA values.
`info.observed`	Should the observed information matrix be used or the expected one be used?

`plot`	Logical. If `TRUE`, a plot will be produced showing the derivative of the concentrated log-likelihood, function of the shape parameter.
`scaleData`	Logical. If `TRUE` observations in `x` (which are positive) are divided by their mean value. The results are in theory not affected by this transformation, but scaling the data could improve the estimation in some cases. The log-likelihood plots are shown using the scaled values so the returned estimate of the scale parameter is not the the abscissa of the maximum shown on the plot.
`cov`	Logical. If `FALSE`, a minimal estimation is performed with no covariance matrix or derivative returned. This can be useful when a large number of ML estimations are required, e.g. to sample from a likelihood ratio.

Details

The likelihood is concentrated with respect to the shape parameter. This function is increasing for small values of the scale parameter β. For large β, the derivative of the concentrated log-likelihood tends to zero, and its sign is that of $(1 - \text{CV}^2)$ where CV is the coefficient of variation, computed using n as denominator in the formula for the standard deviation.

The ML estimate does not exist when the sample has a coefficient of variation CV less than 1 and it may fail to be found when CV is greater than yet close to 1.

Value

A list with the following elements

`estimate`	Parameter ML estimates.
`sd`	Vector of (asymptotic) standard deviations for the estimates.
`loglik`	The maximised log likelihood.
`dloglik`	Gradient of the log-likelihood at the optimum. Its two elements should normally be close to zero.
`cov`	The (asymptotic) covariance matrix computed from theoretical or observed information matrix.
`info`	The information matrix.

Note

The estimates are biased for small or medium sized sample. The bias is positive for the shape parameter, thus the estimated shape tends to be larger than the true unknown value.

Fitting a Lomax distribution to an exponential sample might lead to a divergence since the exponential is the limit of a Lomax distribution with large shape and large scale with constant ratio shape/scale. Fitting this distribution to a sample having a coefficient of variation smaller than 1 is not allowed since it should lead to divergence of the estimation.

The default value of `info.observed` was set to `TRUE` from version `3.0-1` because standard deviations obtained with this choice are usually better.

Author(s)

Yves Deville

References

J. del Castillo and J. Daoudi (2009) "Estimation of the Generalized Pareto Distribution", *Statist. Probab. Lett.* 79(5), pp. 684-688.

D.E. Giles, H. Feng & R.T. Godwin (2013) "On the Bias of the Maximum Likelihood Estimator for the Two-Parameter Lomax Distribution" *Comm. Statist. Theory Methods*. Vol. 42, n. 11, pp. 1934-1950.

N. Johnson, S. Kotz and N. Balakrishnan *Continuous Univariate Distributions* vol. 1, Wiley 1994.

See Also

`Lomax` for the Lomax distribution.

Examples

```
## generate sample
set.seed(1234)
n <- 200
alpha <- 2 + rexp(1)
beta <- 1 + rexp(1)
x <- rlomax(n, scale = beta, shape = alpha)
res <- flomax(x, plot = TRUE)

## compare with a GPD with shape 'xi' and scale 'sigma'
xi <- 1 / alpha; sigma <- beta * xi
res.evd <- evd::fpot(x, threshold = 0, model = "gpd")
xi.evd <- res.evd$estimate["shape"]
sigma.evd <- res.evd$estimate["scale"]
beta.evd <- sigma.evd / xi.evd
alpha.evd <- 1 / xi.evd
cbind(Renext = res$estimate, evd = c(alpha = alpha.evd, beta = beta.evd))
```

`fmaxlo` *ML estimation of a 'maxlo' distribution*

Description

Fast Maximum Likelihood estimation of a 'maxlo' distribution.

Usage

```
fmaxlo(x,
       shapeMin = 1.25,
       info.observed = TRUE,
       plot = FALSE,
       scaleData = TRUE,
       cov = TRUE)
```

Arguments

`x`	Sample vector to be fitted. Should contain only positive non-NA values.
`shapeMin`	Lower bound on the shape parameter. This must be `>= 1.0` since otherwise the ML estimate is obtained with the `scale` parameter equal to `max(x)`.
`info.observed`	Should the observed information matrix be used or the expected one be used?
`plot`	Logical. If `TRUE`, a plot will be produced showing the derivative of the concentrated log-likelihood, function of the shape parameter. The derivative function shown is that of the log-likelihood for the unconstrained maximisation; it is not used in the estimation.
`scaleData`	Logical. If `TRUE` observations in `x` (which are positive) are divided by their mean value. The results are in theory not affected by this transformation, but scaling the data could improve the estimation in some cases. The log-likelihood plots are shown using the scaled values so the returned estimate of the scale parameter is not the the abscissa of the maximum shown on the plot.
`cov`	Logical. If `FALSE`, a minimal estimation is performed with no covariance matrix or derivative returned. This can be useful when a large number of ML estimations are required, e.g. to sample from a likelihood ratio.

Details

The 'maxlo' likelihood is concentrated with respect to the shape parameter, thus the function to be maximised has only one one scalar argument: the scale parameter β. For large scale β, the derivative of the concentrated log-likelihood tends to zero, and its sign is that of $(\mathrm{CV}^2 - 1)$ where CV is the coefficient of variation, computed using n as denominator in the formula for the standard deviation.

The ML estimate does not exist when the sample has a coefficient of variation `CV` greater than `1.0` and it may fail to be found when `CV` is smaller than yet close to `1.0`.

The expected information matrix can be obtained by noticing that when the r.v. Y follows the 'maxlo' distribution with shape α and scale β the r.v $V := 1/(1-Y/\beta)$ follows a Pareto distribution with minimum 1 and and shape parameter α. The information matrix involves the second order moment of V.

The default value of `info.observed` was set to `TRUE` from version `3.0-1` because standard deviations obtained with this choice are usually better.

Value

A list with the following elements

`estimate`	Parameter ML estimates.
`sd`	Vector of (asymptotic) standard deviations for the estimates.
`loglik`	The maximised log-likelihood.
`dloglik`	Gradient of the log-likelihood at the optimum. Its two elements should normally be close to zero.
`cov`	The (asymptotic) covariance matrix computed from theoretical or observed information matrix.
`info`	The information matrix.

Note

The name of the distribution hence also that of the fitting function are still experimental and might be changed.

Author(s)

Yves Deville

See Also

`Maxlo` for the description of the distribution.

Examples

```
## generate sample
set.seed(1234)
n <- 200
alpha <- 2 + rexp(1)
beta <- 1 + rexp(1)
x <- rmaxlo(n, scale = beta, shape = alpha)
res <- fmaxlo(x, plot = TRUE)

## compare with a GPD with shape 'xi' and scale 'sigma'
xi <- -1 / alpha; sigma <- -beta * xi
res.evd <- evd::fpot(x, threshold = 0, model = "gpd")
xi.evd <- res.evd$estimate["shape"]
sigma.evd <- res.evd$estimate["scale"]
beta.evd <- -sigma.evd / xi.evd
alpha.evd <- -1 / xi.evd
cbind(Renext = res$estimate, evd = c(alpha = alpha.evd, beta = beta.evd))
```

`fweibull` *ML estimation of classical Weibull distribution*

Description

Fast Maximum Likelihood estimation of the classical two parameters Weibull distribution.

Usage

```
fweibull(x, info.observed = TRUE, scaleData = TRUE, cov = TRUE,
          check.loglik = FALSE)
```

Arguments

`x`	Sample vector to be fitted. Should contain only positive non-NA values.
`info.observed`	Should the observed information matrix be used or the expected one be used?
`scaleData`	Should the data be scaled before estimation? If `TRUE`, the observations in `x` (which are positive) are divided by their mean value. The results are in theory not affected by this transformation, but scaling the data could improve the estimation in some cases.
`cov`	Should the covariance of estimates be computed?
`check.loglik`	If `TRUE`, the log-likelihood is recomputed using `dweibull` function with `log = TRUE`. The result is returned as a list element.

Details

The ML estimates are obtained thanks to a reparameterisation with $\eta = scale^{1/shape}$ in place of `shape`. This allows the maximisation of a one-dimensional likelihood L since the η parameter can be concentrated out of L. This also allows the determination of the *expected* information matrix for $[shape, \eta]$ rather than the usual *observed* information.

Value

A list

`estimate`	Parameter ML estimates.
`sd`	The (asymptotic) standard deviation for estimate.
`cov`	The (asymptotic) covariance matrix computed from theoretical or observed Information matrix.
`eta`	The estimated value for eta.

Note

The default value of `info.observed` was set to `TRUE` from version `3.0-1` because standard deviations obtained with this choice are usually better.

Author(s)

Yves Deville

See Also

`weibplot` for Weibull plots.

Examples

```
n <- 1000
set.seed(1234)
shape <- 2 * runif(1)
x <- 100 * rweibull(n, shape = 0.8, scale = 1)
res <- fweibull(x)
```

```
## compare with MASS
if (require(MASS)) {
   res2 <- fitdistr(x , "weibull")
   est <- cbind(res$estimate, res2$estimate)
   colnames(est) <- c("Renext", "MASS")
   loglik <- c(res$loglik, res2$loglik)
   est <- rbind(est, loglik)
   est
}

## Weibull plot
weibplot(x,
         shape = c(res$estimate["shape"], res2$estimate["shape"]),
         scale = c(res$estimate["scale"], res2$estimate["scale"]),
         labels = c("Renext 'fweibull'", "MASS 'fitdistr'"),
         mono = TRUE)
```

Garonne *Flow of the french river La Garonne*

Description

Flow of the french river La Garonne at le Mas d'Agenais

Usage

```
Garonne
```

Format

The format is: List of 7

- `$info` : List of 6
  - `$name` : chr "`Garonne`"
  - `$shortLab` : chr "`La Garonne at Le Mas d'Agenais`"
  - `$longLab` : chr "`River flow of La Garonne at Le Mas d'Agenais`"
  - `$varName` : chr "`Flow`"
  - `$varShortLab` : chr "`Flow`"
  - `$varUnit` : chr "`m3/s`"
- `$describe` : chr "`Flow of the french river La Garonne ...`"
- `$OTinfo` :List of 4
  - `$start` : POSIXct[1:1], format: "`1913-01-01`"
  - `$end` : POSIXct[1:1], format: "`1978-01-01`"
  - `$effduration` : num `65`
  - `$threshold` : num `2500`

- `$OTdata` : 'data.frame': 151 obs. of 3 variables:
  - `$date` : POSIXct[1:151], format: `"1913-04-08" "1913-04-25"` ...
  - `$Flow` : num [1:151] `2600 2800 2700 4579 3400` ...
  - `comment` : chr [1:151] "" "" "" "" ...
- `$OTmissing` : `NULL`
- `$MAXinfo` :'data.frame': 1 obs. of 3 variables:
  - `$start` : POSIXct[1:1], format: `"1770-01-01"`
  - `$end` : POSIXct[1:1], format: `"1913-01-01"`
  - `$duration` : num `143`
- `$MAXdata` :'data.frame': 12 obs. of 4 variables:
  - `$block` : num [1:12] `1 1 1 1 1 1 1 1 1 1` ...
  - `date` : POSIXct[1:12], format: `NA NA` ...
  - `$Flow` : num [1:12] `7500 7400 7000 7000 7000 6600 6500 6500 6400 6300` ...
  - `$comment` : chr [1:12] "1 (1875)" "2 (1770)" "3 (1783)" "4 (1855)" ...

- `attr(*, "class")= chr "Rendata"`

Details

The data concern the french river *La Garonne* at the gauging station named *Le Mas d'Agenais* where many floods occurred during the past centuries.

The data consist in OT data and historical data. The variable is the river flow in cube meter per second $(\mathrm{m}^3/\mathrm{s})$ as estimated from the river level using a rating curve. The precision is limited and many ties are present among the flow values.

The OT data or "OTdata" contain flows values over the threshold $u = 2500\,\mathrm{m}$ for the 65 years period 1913-1977. The historical data or "MAXdata" is simply the $r = 12$ largest flows for the period of 143 years 1770-1912. The exact dates of these events are not known with precision but the years are known and given as comments.

Source

The data were taken from the book by Miquel.

References

Miquel J. (1984) *Guide pratique d'estimation des probabilités de crues*, Eyrolles (coll. EDF DER).

Parent E. and Bernier J. (2003) Bayesian POT modeling for Historical data. *Journal of Hydrology* vol. 274, pp. 95-108.

Examples

```
plot(Garonne)
```

gev2Ren *Translate a vector of GEV parameters into renewal model*

Description

Translate a (named) vector of GEV parameters into a renewal model.

Usage

```
gev2Ren(parGev,
        threshold = NULL,
        lambda = NULL,
        w = 1,
        distname.y = c("gpd", "GPD", "lomax", "maxlo"),
        vcovGev = NULL,
        jacobian = TRUE,
        plot = FALSE)
```

Arguments

`parGev`	Named vector of GEV coefficients. This must be a vector of length 3, with names `"loc"`, `"scale"` and `"shape"`.
`threshold`	The threshold of the renewal model.
`lambda`	The rate of the renewal model.
`w`	The duration corresponding to the GEV parameters. Positive numeric scalar assumed to be in years.
`distname.y`	The name of the distributions for the excesses in the renewal model.
`vcovGev`	A (co)variance matrix for the parameter. This must be a symmetric positive matrix with 3 rows and 3 columns, with rownames in correspondence with the parameter names.
`jacobian`	Logical. If `TRUE` a Jacobian matrix will be returned as a `"jacobian"` attribute of the result.
`plot`	If `TRUE` a rough plot will be produced to check the accordance of the GEV and the renewal models. It is a return level plot with the two return level curves shown.

Value

A vector of parameters similar to the coefficient vector of an object with class `"Renouv"`. This vector has an element named `"lambda"` corresponding to the rate of the Homogeneous Poisson Process. The other elements are the parameters for the distribution of POT excesses.

The result has attributes named `"distname.y"` and `"threshold"` which can be used to build a `Renouv` object using the `RenouvNoEst` function. The result may as well have attributes `"jacobian"` and `"vcov"` according to the arguments values. These objects should be used with attention to their element names, since the parameter order may not be the one you expect.

Author(s)

Yves Deville

See Also

The Ren2gev function provide a reciprocal transformation.

Examples

```
## GEV parameters and block duration
set.seed(1234)
muGev <- rnorm(1); sigmaGev <- rgamma(1, shape = 5)
xiGev <- runif(1, min = -0.2, max = 0.3)
parGev <- c("loc" = muGev, "scale" = sigmaGev, "shape" = xiGev)
parRen <- gev2Ren(parGev, lambda = 1, jacobian = TRUE, plot = TRUE)
## check by reverse transform
parGevCheck <- Ren2gev(parRen, threshold = attr(parRen, "threshold"))
rbind(parGev, parGevCheck)

##=========================================================================
## slightly positive shape convert to "gpd" and "lomax" distributions
##=========================================================================
x <- rgev(n = 100, loc = 3500, scale = 1000, shape = 0.1)
fit.gev <- fgev(x, start = list("loc" = 3000, "scale" = 1000, "shape" = 0.1))
distNames <- c("gpd", "lomax")
namesRen <- c("lambda", "scale", "shape") # works for the 2 target dists
fitNew <- list()
opar <- par(mfrow = c(3, 1))
for (nm in distNames) {
  parRen <- gev2Ren(parGev = fit.gev$estimate, threshold = 2800,
                    vcov = fit.gev$var.cov, distname.y = nm)
  namesRen <- c("lambda", "scale", "shape")
  myVcov <- attr(parRen, "vcov")[namesRen, namesRen]
  fitNew[[nm]] <- RenouvNoEst(threshold = attr(parRen, "threshold"),
                              estimate = parRen,
                              distname.y = attr(parRen, "distname.y"),
                              cov = myVcov)
  plot(fitNew[[nm]], Tlim = c(1, 200))
}
plot(fit.gev, which = 4)
par(opar)
##=========================================================================
## slightly negative shape convert to "gpd" and "maxlo" distribution
##=========================================================================
x <- rgev(n = 100, loc = 3500, scale = 1000, shape = -0.2)
fit.gev <- fgev(x, start = list("loc" = 3000, "scale" = 1000, "shape" = 0.1))
distNames <- c("gpd", "maxlo")
namesRen <- c("lambda", "scale", "shape") # works for the 2 target dists
fitNew <- list()
opar <- par(mfrow = c(3, 1))
for (nm in distNames) {
  parRen <- gev2Ren(parGev = fit.gev$estimate, threshold = 2800,
                    vcov = fit.gev$var.cov, distname.y = nm)
  myVcov <- attr(parRen, "vcov")[namesRen, namesRen]
  fitNew[[nm]] <- RenouvNoEst(threshold = attr(parRen, "threshold"),
                              estimate = parRen,
                              distname.y = attr(parRen, "distname.y"),
                              cov = myVcov)
  plot(fitNew[[nm]], Tlim = c(1, 200))
}
plot(fit.gev, which = 4)
par(opar)
```

gof.date *Goodness-of-fit for the distribution of dates*

Description

Goodness-of-fit diagnostics for the distribution of event dates in a (assumed) Poisson process

Usage

```
gof.date(date,
         start = NULL,
         end = NULL,
         plot = TRUE,
         main = NULL,
         skip = NULL,
         plot.type = "skip")
```

Arguments

`date`	Object of class `POSIXct` (or that can be coerced to this class) giving the dates to be tested. Must be in strictly increasing order.
`start`	The beginning of the interval, a `POSIXct` object. If `NULL`, the first event in `date` is used.
`end`	Object of class `POSIXct` the end of the interval. If `NULL`, the last event in `date` is used.
`plot`	Should a plot be shown?
`main`	Character giving the main title of the plot. The default `NULL` stands for a default main describing the period.
`skip`	Optional data.frame with columns `start` and `end` indicating start and end of skipped periods. The two columns need to be coerced to POSIXct objects. They can be POSIXct or character with POSIX datetime format.

`plot.type`	Character indicating the type of plot to produce when a `skip` data.frame is given. With `plot.type = "skip"` the plot shows missing periods as greyed rectangles and the displays the results of a Kolmogorov-Smirnov (KS) test performed on the events. For the `"omit"` case the missing periods are collapsed into vertical lines on the plot and the displayed results are for an "effective" KS test of uniformity performed omitting the missing periods.

Details

In the homogeneous Poisson process, events occur on a time interval in a uniform fashion. More precisely, for a given time interval the distribution of the event dates conditional to their number n is the distribution of the order statistics of a sample of size n of the uniform distribution on this interval.

When the interval has limits taken at events the uniformity statement remains true, but for *inner* events. This behaviour is met when `start` and `end` are not given and taken as the first and last events in `date`.

Value

A list

`effKS.statistic`, `KS.statistic`	Kolmogorov-Smirnov global test statistic for uniformity (bilateral test) omitting slipped periods or not.
`effKS.pvalue`, `KS.pavalue`	Critical probability in the KS test omitting skipped periods or not.
`effnevt`, `nevt`	Number of events omitting skipped periods or not.
`effduration`, `duration`	Effective duration i.e. total duration of non-skipped periods. In years, omitting skipped periods or not.
`effrate`, `rate`	Occurrence rate in **number of events by year**, omitting skipped periods or not.
`effduration`, `duation`	Total duration in **years**, omitting missing periods or not.
`noskip`	Data.frame object giving indications on the periods that are NOT skipped over (hence usually non-missing periods). These are : `start`, `end` (POSIX), `duration` (in years) `rate` (in number of events by year) and Kolmogorov test statistic and p-value. This data.frame is only available when a suitable `skip` has been given.

When the number of events corresponding to the indications of args is `0`, the function returns `NULL` with a warning. When the number of events is less than `6` a warning is shown.

Warning

When skipped periods exist the number of events, duration, rate the global KS test must be computed by omitting the skipped periods in the duration and retaining only valid interevents. The indication given in `nevt` `rate` and `duration` should be used only when no skipped period exist (`skip = NULL` on input) and replaced by `effnevt`, `effrate` and `effduration` otherwise.

Note

In practical contexts missing periods are often met in the datasets. The diagnostic should therefore be applied on *every period with no missing data*. Even if the event dates seem reasonably uniform, it is a good idea to check that the rates do not differ significantly over intervals.

When some events are missing and no suitable information is given via the `skip` argument, the global `rate`, `KS.statistic` and `KS.pvalue` are of little interest. Yet the graph might be instructive.

Author(s)

Yves Deville

See Also

`interevt` function for the determination of interevents ans subsequent diagnostics.

Examples

```
## Use "Brest" dataset
## simple plot. Kolmogorov-Smirnov is not useful
gof1 <- gof.date(date = Brest$OTdata$date)

## consider missing periods. Much better!
gof2 <- gof.date(date = Brest$OTdata$date,
         skip = Brest$OTmissing,
         start = Brest$OTinfo$start,
         end = Brest$OTinfo$end)

print(gof2$noskip)

## Second type of graph
gof3 <- gof.date(date = Brest$OTdata$date,
         skip = Brest$OTmissing,
         start = Brest$OTinfo$start,
         end = Brest$OTinfo$end,
         plot.type = "omit")

## non-skipped periods at Brest
ns <- skip2noskip(skip = Brest$OTmissing,
                 start = Brest$OTinfo$start,
                 end = Brest$OTinfo$end)

## say 9 plots/diagnostics
oldpar <- par(mar = c(3, 4, 3, 2), mfcol = c(3, 3))

for (i in 1:9) {
  GOF <- gof.date(date = Brest$OTdata$date,
           start = ns$start[i],
           end = ns$end[i])
}

par(oldpar)
```

`gofExp.test` *Goodness-of-fit test for exponential distribution*

Description

Bartlett's goodness-of-fit test for exponential distribution

Usage

```
gofExp.test(x)
```

Arguments

`x`	Sample with positive values.

Value

A list with elements

`statistic`	Statistic.
`p.value`	Critical value.

Author(s)

Yves Deville

References

See

Yagouti A., Abi-Zeid I., Ouarda, T.B.M.J. and B. Bobée (2001), Revue de processus ponctuels et synthèse de tests statistiques pour le choix d'un type de processus *Revue des Sciences de l'Eau*, **1**, pp. 323-361.

See Also

Among other goodness-of-fit tests `ks.test` in the `stats` package. See `expplot` for a graphical diagnostic.

Examples

```
## a sample of size 30
x <- rexp(30)
res <- gofExp.test(x)

## ns samples: p.values should look as uniform on (0, 1)
ns <- 100
xmat <- matrix(rexp(30*ns), nrow = ns, ncol = 30)
p.values <- apply(xmat, 1, function(x) gofExp.test(x)$p.value)
plot(sort(p.values), type = "p", pch = 16)
```

GPD *Generalised Pareto Distribution*

Description

Density function, distribution function, quantile function, random generation, hazard and cumulative hazard functions for the Generalised Pareto Distribution.

Usage

```
dGPD(x, loc = 0.0, scale = 1.0, shape = 0.0, log = FALSE)
pGPD(q, loc = 0.0, scale = 1.0, shape = 0.0, lower.tail = TRUE)
qGPD(p, loc = 0.0, scale = 1.0, shape = 0.0, lower.tail = TRUE)
rGPD(n, loc = 0.0, scale = 1.0, shape = 0.0)
hGPD(x, loc = 0.0, scale = 1.0, shape = 0.0)
HGPD(x, loc = 0.0, scale = 1.0, shape = 0.0)
```

Arguments

`x, q`	Vector of quantiles.
`p`	Vector of probabilities.
`n`	Number of observations.
`loc`	Location parameter μ.
`scale`	Scale parameter σ.
`shape`	Shape parameter ξ.
`log`	Logical; if `TRUE`, the log density is returned.
`lower.tail`	Logical; if `TRUE` (default), probabilities are Pr$[X <= x]$, otherwise, Pr$[X > x]$.

Details

Let μ, σ and ξ denote `loc`, `scale` and `shape`. The distribution values y are $\mu \leq y < y_{\text{max}}$.

When $\xi \neq 0$, the survival function value for $y \geq \mu$ is given by

$$S(y) = \left[1 + \xi(y - \mu)/\sigma\right]^{-1/\xi} \qquad \mu < y < y_{\text{max}}$$

where the upper end-point is $y_{\text{max}} = \infty$ for $\xi > 0$ and $y_{\text{max}} = \mu - \sigma/\xi$ for $\xi < 0$.

When $\xi = 0$, the distribution is exponential with survival

$$S(y) = \exp\left[-(y - \mu)/\sigma\right] \qquad \mu \leq y.$$

Value

`dGPD` gives the density function, `pGPD` gives the distribution function, `qGPD` gives the quantile function, and `rGPD` generates random deviates. The functions `hGPD` and `HGPD` return the hazard rate and the cumulative hazard.

Note

The functions are slight adaptations of the `[r,d,p,q]gpd` functions in the **evd** package. The main difference is that these functions return `NaN` when `shape` is negative, as it might be needed in unconstrained optimisation. The quantile function can be used with `p=0` and `p=1`, then returning the lower and upper end-point.

See Also

`fGPD` to fit such a distribution by Maximum Likelihood.

Examples

```
qGPD(p = c(0, 1), shape = -0.2)
shape <- -0.3
xlim <- qGPD(p = c(0, 1), shape = shape)
x <- seq(from = xlim[1], to = xlim[2], length.out = 100)
h <- hGPD(x, shape = shape)
plot(x, h, type = "o", main = "hazard rate for shape < 0")
shape <- 0.2
xlim <- qGPD(p = c(0, 1 - 1e-5), shape = shape)
x <- seq(from = xlim[1], to = xlim[2], length.out = 100)
h <- hGPD(x, shape = shape)
plot(x, h, type = "o", main = "hazard rate shape > 0 ")
```

`gumbel2Ren` *Translate a vector of Gumbel parameters into a vector of parameters for a renewal model*

Description

Translate a (named) vector of Gumbel parameters into a vector of parameters for a renewal model.

Usage

```
gumbel2Ren(parGumbel,
           threshold = NULL,
           lambda = NULL,
           w = 1,
           distname.y = c("exponential"),
           vcovGumbel = NULL,
           jacobian = TRUE,
           plot = FALSE)
```

Arguments

`parGumbel`	Named vector of Gumbel parameters, with name `"scale"` and `"shape"`.
`threshold`	The threshold for the target Renouv parameters.
`lambda`	The rate for the target Renouv parameters.
`w`	A block duration for the conversion.
`distname.y`	The distribution of the excesses.
`vcovGumbel`	A covariance matrix for the Gumbel parameters.
`jacobian`	Logical. If `TRUE`, the jacobian is used.
`plot`	Logical. If `TRUE`, a rough plot will be drawn.

Details

Given a vector of Gumbel parameters and a block duration, there exits an infinity of `Renouv` models with exponential excesses leading to the prescribed Gumbel distributions for the maximum of the marks on a block with duration `w`. One of these models may be chosen by specifying either a threshold or a rate `lambda`.

Value

A vector of Renouv parameters, which can be used with `RenouvNoEst`.

Caution

All Renouv models lead to the same return level curve whatever be the choice of `threshold` or `lambda`. However, when a covariance matrix is given, the covariance matrix for the Renouv parameters and consequently the confidence bounds **depend on the threshold or rate**. So the computed covariance matrix must in general be considered as putative.

Author(s)

Yves Deville

See Also

`gev2Ren` for a similar translation from the GEV distribution.

`Hpoints` *Plotting positions for exponential return levels*

Description

Plotting positions for exponential return level plots.

Usage

```
Hpoints(n)
```

Arguments

`n`	Sample size.

Details

The plotting positions are numeric values to use as the abscissae corresponding to the order statistics in an exponential return level plot. They range from 1 to about $\log n$. They can be related to the plotting positions given by `ppoints`.

The returned vector $\mathbf{H}$ has elements

$$H_i = \frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{n+1-i}$$

for $1 \leq i \leq n$. This is the expectation of the i-th order statistic for a sample of the standard exponential distribution, see e.g. chap. 4 of Embrechts et al.

Value

Numeric vector of plotting positions with length `n`.

Note

For n large enough, the largest value H_n is approximately $\gamma + \log n$ where γ is the Euler-Mascheroni constant, and $\exp H_n$ is about $1.78n$. Thus if the Hpoints are used as plotting positions on a return level plot, the largest observation has a return period of about $1.78n$ years.

Author(s)

Yves Deville

References

Embrechts P., Klüppelberg C. and Mikosch T. (1997) *Modelling Extremal Events for Insurance and Finance*. Springer.

See Also

`ppoints`.

Examples

```
n <- 30
set.seed(1234)
x <- rGPD(n, shape = 0.2)
plot(exp(Hpoints(n)), sort(x), log = "x",
     main = "Basic return level plot")
```

`ini.mixexp2` *Simple estimation for the mixture of two exponential distributions*

Description

Compute a simple (preliminary) estimation for the tree parameters of the mixture of two exponential distributions

Usage

```
ini.mixexp2(x, plot = FALSE)
```

Arguments

`x`	Sample: numerical vector with elements >`0`.
`plot`	Should a graphic be displayed?

Details

This function gives estimators using several methods if necessary. The goal is to find the rates `rate1`, `rate2` and the mixing probability `prob1` with the 'feasibility' constraints `0` < `rate1` < `rate2` and `0` < `prob1` < `1`.

First the method of moments is used. If the estimates are feasible they are returned with `method = "moments"`. If not, the estimates are derived using two linear regressions. A regression without constant using only the smallest values gives an estimator of the mean rate. A regression using only the largest values gives `rate1` and `prob1`. Yet the constraints must be fulfilled. If they are, the estimates are returned (together with `method = "Hreg"` suggesting a cumulative hazard regression). If not, a (poor) default estimate is returned with `method = "arbitrary"`.

Value

A list

`estimate`	A vector with named elements `"prob1"`, `"rate1"` and `"rate2"`.
`method`	The method that really produced the estimators.

Note

The method of moments is implemented in `mom.mixexp2`. Further investigations are needed to compare the estimators (moments or Hreg) and select the best strategy.

Note that this function returns the estimate within a list and no longer as a vector with named elements as was the case before.

Author(s)

Yves Deville

See Also

See `MixExp2`, `mom.mixexp2`.

Examples

```
set.seed(1234)
x <- rmixexp2(n = 100, prob1 = 0.5, rate2 = 4)
res <- ini.mixexp2(x, plot = TRUE)
```

`interevt` *Interevents (or interarrivals) from events dates*

Description

Compute intervent durations from events dates

Usage

```
interevt(date,
         skip = NULL, noskip = NULL)
```

Arguments

`date`	A `POSIXct` vector containing the date(time) of the events.
`skip`	A data.frame containing two `POSIXct` columns `start` and `end` describing the periods to skip over.
`noskip`	A data.frame like `skip` but where the periods define the NON skipped part of the events.

Details

Interevents are the time differences between successive dates. When the `date` argument contains occurrence times T_i for successive events of an homogeneous Poisson process, interevents $T_i - T_{i-1}$ are mutually independent with the same exponential distribution.

When some time intervals are skipped independently from the event point process, we may consider the interevents $T_i - T_{i-1}$ between two non-skipped events such that the time interval $(T_{i-1},\, T_i)$ does not contains any skipped interval. These interevents still are mutually independent with the same exponential distribution. When `skip` or `noskip` is not `NULL` the computation therefore only retains couples of two successive datetimes "falling" in the same non-skipped period, which number can therefore be associated with the interevent.

Value

A list mainly containing a `interevt` data.frame.

`interevt`	Data.frame. Each row describes a retained interevent through a `period` integer giving the "noskip" period, a `start` and `end` `POSIXct` and a `duration` in **days**.
`noskip`	Only when `skip` or `noskip` args have been given. A data.frame containing broadly the same information as the `noskip` arg is it was given or the information deduced from the `skip` arg if given.
`axis`	When needed, a list with some material to build an axis with uneven ticks as in the `gof.date` with `skip.action = "omit"`.

Note

Only one of the two arguments `skip` and `noskip` should be given in the call. In each case, the rows of the returned data.frame objects describe periods in chronological order. That is: `start` at row `2` must be after the `end` value of row `1` and so on.

Note that there are usually less interevents than dates since two successive dates will be retained for an interevent only when they are not separated by missing period. As a limit case, there can be no interevents if the `noskip` periods contain only one date from the `date` vector.

Author(s)

Yves Deville

See Also

`gof.date` for goodness-of-fit diagnostics for dates of events `expplot` for diagnostics concerning the exponential distribution.

Examples

```
## Use Brest data
ie <- interevt(date = Brest$OTdata$date, skip = Brest$OTmissing)

expplot(ie$interevt$duration, rate = 1 / mean(ie$interevt$duration),
  main = "No threshold")
```

```
## keep only data over a threshold
ind1 <- Brest$OTdata$Surge >= 35
ie1 <- interevt(Brest$OTdata$date[ind1], skip = Brest$OTmissing)
expplot(ie1$interevt$duration, main = "Threshold = 35")

## increase threshold
ind2 <- Brest$OTdata$Surge >= 55
ie2 <- interevt(date = Brest$OTdata$date[ind2], skip = Brest$OTmissing)
expplot(ie2$interevt$duration, main = "Threshold = 55 cm")
```

`Jackson` *Jackson's statistic*

Description

Jackson's statistic for the exponentiality test.

Usage

```
Jackson(x, norm = FALSE)
```

Arguments

`x`	Numeric vector or matrix. In the second case, rows are considered as samples.
`norm`	Logical: if `TRUE`, the statistic is normalized by using the *asymptotic* mean and standard deviation, respectively 2 and 1.

Details

The value(s) of the statistic are the ratio of two weighted means of the order statistics.

Value

A numeric vector of length `1` when `x` is a vector, or with length `nrow(x)` when `x` is a matrix.

References

J. Beirlant and T. de Weit and Y. Goegebeur(2006) A Goodness-of-fit Statistic for Pareto-Type Behaviour, *J. Comp. Appl. Math.*, 186(1), pp. 99-116

`Jackson.test` *Jackson's test of exponentiality*

Description

Jackson's test of exponentiality

Usage

```
Jackson.test(x, method = c("num", "sim", "asymp"), nSamp = 15000)
```

Arguments

`x`	numeric vector or matrix.
`method`	Character: choice of the method used to compute the p-value. See the **Details** section.
`nSamp`	Number of samples used to compute the p-value if `method` is `"sim"`.

Details

Compute the Jackson's test of exponentiality. The test statistic is the ratio of weighted sums of the order statistics. Both sums can also be written as weighted sums of the *scalings*.

The Jackson's statistic for a sample of size n of the exponential distribution can be shown to be approximately normal. More precisely $\sqrt{n}(J_n - 2)$ has approximately a standard normal distribution. This distribution is used to compute the p-value when `method` is `"asymp"`. When `method` is `"num"`, a numerical approximation of the distribution is used. Finally, when `method` is `"sim"` the p-value is computed by simulating `nSamp` samples of size `length(x)` and estimating the probability to have a Jackson's statistic larger than that of the 'observed' `x`.

Value

A list of results.

`statistic`, `p.value`	The statistic and p-value.
`df`	Number n of observations.
`method`	Description of the test implemented, regardless of how the p-value has been computed.

Note

Jackson's test of exponentiality works fine for a Lomax alternative (GPD with heavy tail). It then reaches nearly the same power as a Likelihood Ratio (LR) test, see Kozubowski et al. It can be implemented more easily than the LR test because simulated values of the test statistic can be obtained quickly enough to compute the p-value by simulation.

Author(s)

Yves Deville

References

J. Beirlant and T. de Weit and Y. Goegebeur(2006) "A Goodness-of-fit Statistic for Pareto-Type Behaviour", *J. Comp. Appl. Math.*, 186(1), pp. 99-116.

T.J. Kozubowski, A. K. Panorska, F. Qeadan, A. Gershunov and D. Rominger (2009) "Testing Exponentiality Versus Pareto Distribution via Likelihood Ratio" *Comm. Statist. Simulation Comput.* 38(1), pp. 118-139.

See Also

The `Jackson` function computing the statistic and the `LRExp.test` function.

Examples

```
set.seed(1234)
x <- rGPD(n = 50, loc = 0, scale = 1, shape = 0.1)
Jackson.test(x, method = "num")$p.value
Jackson.test(x, method = "asymp")$p.value
Jackson.test(x, method = "sim")$p.value
```

`logLik.Renouv` *Log-likelihood of a "Renouv" object*

Description

Log-likelihood, AIC, BIC and number of observations of an object of class "Renouv".

Usage

```
## S3 method for class 'Renouv'
AIC(object, ..., k = 2)
## S3 method for class 'Renouv'
BIC(object, ...)
## S3 method for class 'Renouv'
logLik(object, ...)
## S3 method for class 'Renouv'
nobs(object, ...)
```

Arguments

`object`	Object of class `"Renouv"`.
`k`	See `AIC`.
`...`	Not used yet.

Caution

Comparing log-likelihoods, AIC or BIC for different `Renouv` objects makes sense only when these share the same data and the same threshold.

Note

`logLik`, `AIC` and `BIC` can be used with an object of class `"Renouv"` which makes use of historical data. In this case, the number of observations may be misleading since a single historical observation may concern dozens of years and thus have a much greater impact on the estimation of the tail than an "ordinary" observation.

Author(s)

Yves Deville

See Also

The `AIC`, `nobs` generic functions.

`Lomax` *Lomax distribution*

Description

Density function, distribution function, quantile function and random generation for the Lomax distribution.

Usage

```
dlomax(x, scale = 1.0, shape = 4.0, log = FALSE)
plomax(q, scale = 1.0, shape = 4.0, lower.tail = TRUE)
qlomax(p, scale = 1.0, shape = 4.0)
rlomax(n, scale = 1.0, shape = 4.0)
```

Arguments

`x, q`	Vector of quantiles.
`p`	Vector of probabilities.
`n`	Number of observations.
`scale, shape`	Scale and shape parameters. Vectors of length > 1 are not accepted.
`log`	Logical; if `TRUE`, the log density is returned.
`lower.tail`	Logical; if `TRUE` (default), probabilities are Pr$[X <= x]$, otherwise, Pr$[X > x]$.

Details

The Lomax distribution function with shape $\alpha > 0$ and scale $\beta > 0$ has survival function

$$S(y) = [1 + y/\beta]^{-\alpha} \qquad (y > 0)$$

This distribution has increasing hazard and decreasing mean residual life (MRL). The coefficient of variation decreases with α, and tends to 1 for large α. The default value $\alpha = 4$ corresponds to CV $= \sqrt{2}$.

Value

`dlomax` gives the density function, `plomax` gives the distribution function, `qlomax` gives the quantile function, and `rlomax` generates random deviates.

Note

This distribution is sometimes called *log-exponential*. It is a special case of Generalised Pareto Distribution (GPD) with positive shape $\xi > 0$, scale σ and location $\mu = 0$. The Lomax and GPD parameters are related according to

$$\alpha = 1/\xi, \qquad \beta = \sigma/\xi.$$

The Lomax distribution can be used in POT to describe excesses following GPD with shape $\xi > 0$ thus with decreasing hazard and increasing Mean Residual Life.

Note that the exponential distribution with rate ν is the limit of a Lomax distribution having large scale β and large shape α, with the constraint on the shape/scale ratio $\alpha/\beta = \nu$.

References

Johnson N. Kotz S. and N. Balakrishnan *Continuous Univariate Distributions* vol. 1, Wiley 1994.

Lomax distribution in Wikipedia

See Also

`flomax` to fit the Lomax distribution by Maximum Likelihood.

Examples

```
shape <- 5; scale <- 10
xl <- qlomax(c(0.00, 0.99), scale = scale, shape = shape)
x <- seq(from = xl[1], to = xl[2], length.out = 200)
f <- dlomax(x, scale = scale, shape = shape)
plot(x, f, type = "l", main = "Lomax density")
F <- plomax(x, scale = scale, shape = shape)
plot(x, F, type ="l", main ="Lomax distribution function")
```

LREXP *Likelihood Ratio statistic for exponential vs. GPD*

Description

Likelihood Ratio statistic for the exponential distribution vs. GPD.

Usage

```
LRExp(x, alternative = c("lomax", "GPD", "gpd", "maxlo"))
```

Arguments

`x`	Numeric vector of positive sample values. For the POT context, this should be the vector of excesses over the threshold.
`alternative`	Character string describing the alternative hypothesis

Details

The Likelihood-Ratio statistic is actually $W := -2\log \mathrm{LR}$ where LR is the ratio of the likelihoods *exponential* to *alternative distribution*.

Value

The LR statistic value.

Note

When the alternative is `"lomax"` or `"maxlo"`, the statistic has a distribution of *mixed type* under the null hypothesis of exponentiality. This is a mixture of a distribution of continuous type (with positive values) and of a Dirac mass at LR = 0. The probability mass $\Pr\{\mathrm{LR} = 0\}$ can be computed using the `pGreenwood1` function. More precisely, the probability mass is `pGreenwood1(n)` for the Lomax alternative and `1.0 - pGreenwood1(n)` for the maxlo alternative, where `n` is the sample size `length(x)`.

See Also

`LRExp.test` for the related LR test of exponentiality.

LRExp.test — *Likelihood Ratio test of exponentiality vs. GPD*

Description

Likelihood Ratio test of exponentiality vs. GPD.

Usage

```
LRExp.test(x,
           alternative = c("lomax", "GPD", "gpd", "maxlo"),
           method = c("num", "sim", "asymp"),
           nSamp = 15000,
           simW = FALSE)
```

Arguments

`x`	Numeric vector of positive sample values. For the POT context this should be the vector of excesses over the threshold.
`alternative`	Character string describing the alternative distribution.
`method`	Method used to compute the p-value.
`nSamp`	Number of samples for a simulation, if `method` is `"sim"`.
`simW`	Logical. If this is set to `TRUE` and `method` is `"sim"`, the simulated values are returned as an element `W` in the list.

Details

The Lomax and maxlo alternatives correspond to a GPD alternative with positive shape parameter $\xi > 0$ (Lomax) and GPD with $\xi < 0$ (maxlo).

The *asymptotic* distribution of the Likelihood-ratio statistic is known. For the GPD alternative, this is a chi-square distribution with one df. For the Lomax alternative, this is the distribution of a product BC where B and C are two independent random variables following a Bernoulli distribution with probability parameter $p = 0.5$ and a chi-square distribution with one df.

- When `method` is `"num"`, a numerical approximation of the distribution is used. This method is not unlike that used by Kozubowski et al., but a different approximation is used. However, if `x` has a length $n > 500$, the method is turned to `"asymp"`.
- When `method` is `"sim"`, `nSamp` samples of the exponential distribution with the same size as `x` are drawn and the LR statistic is computed for each sample. The p-value is simply the estimated probability that a simulated LR is greater than the observed LR.
- Finally when `method` is `"asymp"`, the asymptotic distribution is used.

Value

A list of results with elements `statistic`, `p.value` and `method`. Other elements are

`alternative`	Character describing the alternative hypothesis.
`W`	If `simW` is `TRUE` and `method` is `"sim"` only. A vector of `nSamp` simulated values of the statistic $W := -2\log \text{LR}$.

Note

For the Lomax alternative, the distribution of the test statistic has *mixed type*: it can take any positive value as well as the value 0 with a positive probability mass. The probability mass is the probability that the ML estimate of the GPD shape parameter is negative, and a good approximation of it is provided by the `pGreenwood1` function. Note that this probability converges to its limit 0.5 *very slowly*, which suggests that the asymptotic distribution provides poor results for medium sample sizes, say < 100.

Similarly for a maxlo alternative, the distribution of the test statistic has mixed type: it can take any positive value as well as the value 0 with a positive probability mass approximately given by `1 -pGreenwood1(n)` where n is the sample size.

Author(s)

Yves Deville

References

T.J. Kozubowski, A. K. Panorska, F. Qeadan, A. Gershunov and D. Rominger (2009) "Testing Exponentiality Versus Pareto Distribution via Likelihood Ratio" *Comm. Statist. Simulation Comput.* 38(1), pp. 118-139.

The approximation method used is described in the *Renext Computing Details* report.

See Also

`Lomax`, `Maxlo`, `GPD` for the alternatives used here.

Examples

```
set.seed(1234)
x <- rGPD(n = 50, loc = 0, scale = 1, shape = 0.1)
LRExp.test(x, method = "num")$p.value
LRExp.test(x, method = "asymp")$p.value
## Not run:
## requires much time
LRExp.test(x, method = "sim")$p.value

## End(Not run)
```

`LRGumbel` *Likelihood Ratio statistic for Gumbel vs. GEV*

Description

Likelihood Ratio statistic for the Gumbel distribution vs. GEV.

Usage

```
LRGumbel(x, alternative = c("frechet", "GEV"))
```

Arguments

`x`	Numeric vector of sample values.
`alternative`	Character string describing the alternative.

Details

The Likelihood-Ratio statistic is actually $W := -2\log \text{LR}$ where LR is the ratio of the likelihoods *Gumbel* to *alternative distribution*.

Value

The LR statistic value.

Note

When the alternative is `"frechet"`, the statistic has a distribution of mixed type under the null hypothesis of a Gumbel distribution.

Author(s)

Yves Deville

See Also

`LRGumbel.test` for the related LR test of Gumbelity.

`LRGumbel.test` *Likelihood Ratio test for the Gumbel distribution*

Description

Likelihood Ratio test of Gumbel vs. GEV

Usage

```
LRGumbel.test(x,
              alternative = c("frechet", "GEV"),
              method = c("num", "sim", "asymp"),
              nSamp = 1500,
              simW = FALSE)
```

Arguments

`x`	Numeric vector of sample values.
`alternative`	Character string describing the alternative distribution.
`method`	Method used to compute the p-value.
`nSamp`	Number of samples for a simulation, if `method` is `"sim"`.
`simW`	Logical. If this is set to `TRUE` and `method` is `"sim"`, the simulated values are returned as an element `W` in the list.

Details

The asymptotic distribution of the Likelihood-ratio statistic is known. For the GEV alternative, this is a chi-square distribution with one df. For the Fréchet alternative, this is the distribution of a product XY where X and Y are two independent random variables following a Bernoulli distribution with probability parameter $p = 0.5$ and a chi-square distribution with one df.

- When `method` is `"num"`, a numerical approximation of the distribution is used.
- When `method` is `"sim"`, `nSamp` samples of the Gumbel distribution with the same size as `x` are drawn and the LR statistic is computed for each sample. The p-value is simply the estimated probability that a simulated LR is greater than the observed LR. This method requires more computation time than the tow others.
- Finally when `method` is `"asymp"`, the asymptotic distribution is used.

Value

A list of results with elements `statistic`, `p.value` and `method`. Other elements are

`alternative`	Character describing the alternative hypothesis.
`W`	If `simW` is `TRUE` and `method` is `"sim"` only. A vector of `nSamp` simulated values of the statistic $W := -2\log \text{LR}$.

Note

For the Fréchet alternative, the distribution of the test statistic has *mixed type*: it can take any positive value as well as the value 0 with a positive probability mass. The probability mass is the probability that the ML estimate of the GEV shape parameter is negative.

When `method` is `"sim"`, the computation can be slow because each of the `nSamp` simulated values requires two optimisations. The `"asymp"` method provides an acceptable precision for $n \geq 50$, and may even be used for $n \geq 30$.

Author(s)

Yves Deville

Examples

```
set.seed(1234)
x <- rgumbel(60)
res <- LRGumbel.test(x)
```

`Maxlo` *'maxlo' distribution*

Description

Density function, distribution function, quantile function and random generation for the 'maxlo' distribution.

Usage

```
dmaxlo(x, scale = 1.0, shape = 4.0, log = FALSE)
pmaxlo(q, scale = 1.0, shape = 4.0, lower.tail = TRUE)
qmaxlo(p, scale = 1.0, shape = 4.0)
rmaxlo(n, scale = 1.0, shape = 4.0)
```

Arguments

`x, q`	Vector of quantiles.
`p`	Vector of probabilities.
`n`	Number of observations.
`scale, shape`	Shift and shape parameters. Vectors of length > 1 are not accepted.
`log`	Logical; if `TRUE`, the log density is returned.
`lower.tail`	Logical; if `TRUE` (default), probabilities are Pr$[X <= x]$, otherwise, Pr$[X > x]$.

Details

The 'maxlo' distribution function with shape $\alpha > 0$ and scale $\beta > 0$ is a special case of Generalised Pareto (GPD) with *negative shape* $\xi < 0$ and location at zero. This is the finite upper endpoint case of the GPD. Its name is nonstandard and was chosen to suggest some form of symmetry with respect to the Lomax distribution.

The survival function is

$$S(y) = [1 - y/\beta]^{\alpha} \qquad 0 < y < \beta$$

This distribution has a coefficient of variation smaller than 1.

Value

`dmaxlo` gives the density function, `pmaxlo` gives the distribution function, `qmaxlo` gives the quantile function, and `rmaxlo` generates random deviates.

Note

The 'maxlo' and GPD parameters are related according to

$$\alpha = -1/\xi, \qquad \beta = -\sigma/\xi.$$

where σ is the scale parameter of the GPD. Since only GPD with $\xi > -0.5$ seem to be used in practice, this distribution should be used with $\alpha > 2$.

This distribution can be used in POT to describe bounded excesses following GPD with shape $\xi < 0$. The scale parameter β then represents the upper end-point of the excesses, implying the finite upper end-point $u + \beta$ for the levels, where u is the threshold. It can be used in `Renouv` with a fixed scale parameter, thus allowing a control of the upper end-point.

This distribution is simply a rescaled version of a beta distribution and also a rescaled version of a Kumaraswamy distribution. The name "maxlo" is used here to suggest a form of symmetry to Lomax distribution.

See Also

`fmaxlo` to fit such a distribution by Maximum Likelihood.

Examples

```
xs <- rmaxlo(500, shape = 2.2, scale = 1000)
hist(xs, main = "'maxlo' distribution"); rug(xs)

xs <- rmaxlo(500, shape = 4, scale = 1000)
hist(xs, main = "'maxlo' distribution"); rug(xs)

x <- seq(from = -10, to = 1010, by = 2)
plot(x = x, y = dmaxlo(x, shape = 4, scale = 1000),
     type = "l", ylab = "dens",
     col = "orangered", main = "dmaxlo and dgpd")
abline(h = 0)
lines(x = x, y = dgpd(x, shape = -1/4, scale = 250),
     type = "l",
        col = "SpringGreen3", lty = "dashed")
```

MixExp2 *Mixture of two exponential distributions*

Description

Probability functions associated to the mixture of two exponential distributions.

Usage

```
dmixexp2(x, prob1,
         rate1 = 1.0, rate2 = rate1 + delta, delta,
         log = FALSE)
pmixexp2(q, prob1,
         rate1 = 1.0, rate2 = rate1 + delta, delta,
         log = FALSE)
qmixexp2(p, prob1,
         rate1 = 1.0, rate2 = rate1 + delta, delta)
rmixexp2(n, prob1,
         rate1 = 1.0, rate2 = rate1 + delta, delta)
hmixexp2(x, prob1,
         rate1 = 1.0, rate2 = rate1 + delta, delta)
Hmixexp2(x, prob1,
         rate1 = 1.0, rate2 = rate1 + delta, delta)
```

Arguments

`x, q`	Vector of quantiles.
`p`	Vector of probabilities.
`n`	Number of observations.
`log`	Logical; if `TRUE`, the log density is returned.
`prob1`	Probability weight for the "number 1" exponential density.
`rate1`	Rate (inverse expectation) for the "number 1" exponential density.
`rate2`	Rate (inverse expectation) for the "number 2" exponential density. Should in most cases be > `rate1`. See *Details*.
`delta`	Alternative parameterisation `delta` = `rate2` - `rate1`.

Details

The density function is the mixture of two exponential densities

$$f(x) = \alpha_1 \lambda_1 \, e^{-\lambda_1 x} + (1 - \alpha_1) \lambda_2 \, e^{-\lambda_2 x} \qquad x > 0$$

where α_1 is the probability given in `prob1` while λ_1 andλ_2 are the two rates given in `rate1` and `rate2`.

A 'naive' identifiability constraint is

$$\lambda_1 < \lambda_2$$

i.e. `rate1 < rate2`, corresponding to the simple constraint `delta > 0`. The parameter `delta` can be given instead of `rate2`.

The mixture distribution has a decreasing hazard, increasing Mean Residual Life (MRL) and has a thicker tail than the usual exponential. However the hazard, MRL have a finite non zero limit and the distribution behaves as an exponential for large return levels/periods.

The quantile function is not available in closed form and is computed using a dedicated numerical method.

Value

`dmiwexp2`, `pmiwexp2`, `qmiwexp2`, evaluates the density, the distribution and the quantile functions. `dmixexp2` generates a vector of `n` random draws from the distribution. `hmixep2` gives hazard rate and `Hmixexp2` gives cumulative hazard.

Examples

```
rate1 <- 1.0
rate2 <- 4.0
prob1 <- 0.8
qs <- qmixexp2(p = c(0.99, 0.999), prob1 = prob1,
               rate1 = rate1, rate2 = rate2)
x <- seq(from = 0, to = qs[2], length.out = 200)
F <- pmixexp2(x, prob1 = prob1, rate1 = rate1, rate2 = rate2)
plot(x, F, type = "l", col = "orangered", lwd = 2,
     main = "Mixexp2 distribution and quantile for p = 0.99")
abline(v = qs[1])
abline(h = 0.99)
```

`mom.mixexp2` *Moment estimation for the mixture of two exponential distributions*

Description

Compute the moment estimation for the tree parameters of the mixture of two exponential distributions

Usage

```
mom.mixexp2(x)
```

Arguments

`x`	Sample. Vector containing values `>0`.

Details

The three parameters (probability and the two rates) are computed from the first three moments (theoretical and sample). It can be shown that the inverse rates are obtained solving a quadratic equation. However the roots can be negative or complex and the estimates are not valid ones.

Value

A list with elements

`estimate`	A vector with named elements `"prob1"`, `"rate1"` and `"rate2"`. When the moment estimators are not valid (negative or complex rates), a vector of three `NA` is returned.
`method`	Character `"moments"`.

Note

The theoretical coefficient of variation (CV) of a mixture of two exponential distributions always exceeds 100%. When the sample CV is smallest than 100%, no valid estimates exist since the two first moments can not be matched.

Author(s)

Yves Deville

References

Paul R. Rider. The Method of Moments Applied to a Mixture of Two Exponential Distributions. *Ann. Math. Statist.* Vol. 32, Number 1 (1961), 143-147.

See Also

See `ini.mixexp2` for a more versatile initial estimation.

Examples

```
x <- rmixexp2(n = 100, prob1 = 0.5, rate1 = 1.0, rate2 = 3.0)
est <- mom.mixexp2(x)
```

mom2par *Parameters from moments*

Description

Compute parameters from (theoretical) moments

Usage

```
mom2par(densfun = "exponential",
        mean,
        sd = NA)
```

Arguments

`densfun`	Name of the distribution. This can be at present time: `"exponential"`, `"weibull"`, `"gpd"`, `"gamma"`, `"negative binomial"`.
`mean`	Theoretical mean (expectation) of the distribution. Can be a vector, in which case the parameters will be vectors.
`sd`	Standard deviation.

Details

For some distributions like Weibull, it is necessary to find a numerical solution since the parameters have no closed form expression involving the moments.

Value

A named list containing the parameters values e.g. with names `shape` and `scale`. When `mean` or `sd` is vector the list contains vectors.

Note

The name of the formal argument `densfun` is for compatibility with `fitdistr` from the MASS package. However, unlike in `fitdistr` this formal can not be given a density value, i.e. an object of the class `"function"` such as `dnorm`.

Author(s)

Yves Deville

Examples

```
## Weibull
mom2par(densfun = "weibull", mean = 1, sd = 2)
## Genrealised Pareto
mom2par(densfun = "gpd", mean = 1, sd = 2)
## Gamma
mom2par(densfun = "gamma", mean = 1:10, sd = 1)
```

NBlevy — *Negative Binomial Levy process*

Description

Negative Binomial Lévy process estimation from partial observations (counts)

Usage

```
NBlevy(N,
       gamma = NA,
       prob = NA,
       w = rep(1, length(N)),
       sum.w = sum(w),
       interval = c(0.01, 1000),
       optim = TRUE,
       plot = FALSE, ...)
```

Arguments

`N`	Vector of counts, one count by time period.
`gamma`	The `gamma` parameter if known (NOT IMPLEMENTED YET).
`prob`	The `prob` parameter if known (NOT IMPLEMENTED YET).
`w`	Vector of time length (durations).
`sum.w`	NOT IMPLEMENTED YET. The effective duration. If `sum.w` is strictly inferior to `sum(w)`, it is to be understood that missing periods occur within the counts period. This can be taken into account with a suitable algorithm (Expectation Maximisation, etc.)
`interval`	Interval giving min and max values for `gamma`.
`optim`	If `TRUE` a one-dimensional optimisation is used. Else the zero of the derivative of the (concentrated) log-likelihood is searched for.
`plot`	Should a plot be drawn? *May be removed in the future.*
`...`	Arguments passed to `plot`.

Details

The vector $\mathbf{N}$ contains counts for events occurring on non-overlapping time periods with lengths given in $\mathbf{w}$. Under the NB Lévy process assumptions, the observed counts (i.e. elements of $\mathbf{N}$) are independent random variables, each following a negative binomial distribution. The size parameter r_k for N_k is $r_k = \gamma w_k$ and the probability parameter p is `prob`. The vector $\boldsymbol{\mu}$ of the expected counts has elements

$$\mu_k = \mathrm{E}(N_k) = \frac{1-p}{p}\,\gamma\,w_k.$$

The parameters γ and p (`prob`) are estimated by Maximum Likelihood using the likelihood concentrated with respect to the `prob` parameter.

Value

A list with the results

`estimate`	Parameter estimates.
`sd`	Standard deviation for the estimate.
`score`	Score vector at the estimated parameter vector.
`info`	Observed information matrix.
`cov`	Covariance matrix (approx.).

Note

The Negative Binomial Lévy process is an alternative to the Homogeneous Poisson Process when counts are subject to overdispersion. In the NB process, all counts share the same index of dispersion (variance/expectation ratio), namely `1/prob`. When `prob` is close to 1, the counts are nearly Poisson-distributed.

Author(s)

Yves Deville

References

Kozubowski T.J. and Podgórsky K. (2009) "Distributional properties of the negative binomial Lévy process". *Probability and Mathematical Statistics* **29**, pp. 43-71. Lund University Publications.

See Also

`NegBinomial` for the negative binomial distribution, `glm.nb` from the MASS package for fitting Generalised Linear Model of the negative binomial family.

Examples

```
## known parameters
nint <- 100
gam <- 6; prob <- 0.20

## draw random w, then the counts N
w <- rgamma(nint, shape = 3, scale = 1/5)
N <- rnbinom(nint, size = w * gam, prob = prob)
mu <- w * gam * (1 - prob) / prob
Res <- NBlevy(N = N, w = w)

## Use example data 'Brest'
## compute the number of event and duration of the non-skipped periods
gof1 <- gof.date(date = Brest$OTdata$date,
                 skip = Brest$OTmissing,
                 start = Brest$OTinfo$start,
                 end = Brest$OTinfo$end,
                 plot.type = "omit")
ns1 <- gof1$noskip
```

```
## fit the NBlevy
fit1 <- NBlevy(N = ns1$nevt, w = ns1$duration)

## use a higher threshold
OT2 <- subset(Brest$OTdata, Surge > 50)
gof2 <- gof.date(date = OT2$date,
                 skip = Brest$OTmissing,
                 start = Brest$OTinfo$start,
                 end = Brest$OTinfo$end,
                 plot.type = "omit")
ns2 <- gof2$noskip
## the NBlevy prob is now closer to 1
fit2 <- NBlevy(N = ns2$nevt, w = ns2$duration)

c(fit1$prob, fit2$prob)
```

OT2MAX *Temporal aggregation of a Marked Process*

Description

Temporal aggregation of a Marked Process, leading to block maxima or r-largest observations.

Usage

```
OT2MAX(OTdata,
       OTmissing = NULL,
       start = NULL,
       end = NULL,
       MAX.r = 1L,
       blockDuration = "year",
       monthGapStat = TRUE,
       maxMissingFrac = 0.05,
       dataFrames = FALSE,
       infMAX = FALSE,
       plot = TRUE,
       plotType = c("max", "gaps"),
       jitterSeed = 123,
       trace = 0L,
       ...)
```

Arguments

`OTdata`	Data frame containing a `POSIXct` column `date` and the marks variable.
`OTmissing`	Optional data frame with columns `start` and `end` (coerced to `POSIXct`) giving the beginning and the end of gaps.

`start`	An object coerced to `POSIXct` indicating the beginning of reliable/usable information. Unless this is a beginning of block (1-st of January for years), the 1-st block will begin *after* `start` in order to use only qualified information.
`end`	An object indicating the end of the reliable/usable information. Unless this is a end of block (1-st of January for years), the last block will end *before* `end` in order to use only qualified information.
`MAX.r`	*Target* number of observations in the blocks. Can be of length one (same number of observations for all blocks) or of length equal to the number of blocks, the values being then for the blocks in the same order. In both cases, the target number may be impossible to reach because of a smaller number of events in the block. If `infMAX` is `TRUE`, the target number of observations will be reached by filling if needed with `-Inf` values. The rationale for this is that a non-existing event is assumed to have an arbitrarily small mark.
`blockDuration`	Duration of the blocks. Can only be `"year"` for now.
`monthGapStat`	Logical. Setting it to `TRUE` will compute statistics concerning the gaps and return them or show them on a plot.
`maxMissingFrac`	Maximal fraction of a block duration (between 0 and 1) that can be missing without leading to a `NA` aggregated value.
`dataFrames`	If `TRUE`, the result will contain data frames similar to those found in an object with class `"Rendata"`. If `FALSE` the result will contain *list* and *vector* objects, similar to those used as inputs in the `Renouv` function under the names `MAX.data` and `MAX.effDuration`. Note however, that `-Inf` values can be found in these objects when `infMAX` is `TRUE`.
`infMAX`	If `FALSE`, the target number of values the blocks will generally not be reached, because the total number of events in a block can be lower than the target number. Then, the target number value is revised to the number of found values in each block. If `TRUE`, the target number of values is reached by filling the values with `-Inf` and the datetimes with (`POSIXct`) `NA`s.
`plot`	If `TRUE` a simple plot is shown.
`plotType`	Character controlling the plot. With `"max"`, the block maxima are shown. With `plotType = "gap"`, the daily and monthly gap rates are shown. This is possible when suitable information concerning gaps is provided in `OTmissing`. The plot then shows the probability that a given day of the year falls in a gap, as well as monthly gap rates. Most often one wants that the gap rate does not show a seasonal behaviour. Note that gap rates for month-year combinations are shown as grey segments after jitterizing them since the values `0` and `1` may be observed for several years. An alternative way to is using the `monthGapTS` multivariate time series returned by the function, see **Examples**.
`jitterSeed`	Random seed for jittering. Used only when `plot` is `TRUE`, `plotType` is `"gap"` and when suitable information is provided in `OTmissing`.
`trace`	Integer level of verbosity.
`...`	Other arguments to be passed to `plot`.

Details

The data frame given in `OTdata` contains the *events* (or *arrivals*) given by the `date` column, as well as one mark column. Depending on the argument `MAX.r`, the maxima or the r-largest observations of the marks is computed for each time block. When known gaps exist in the data and when they are given in `OTmissing`, a block for which the total duration of gaps is too large will be omitted.

Value

A list, the content of which depends on the value of `dataFrames`. If this value is `TRUE`, the following elements are returned.

`MAXdata`	A data frame of largest values by block with one row for each observation. The largest values are given as columns with names equal to those in the `OTdata` data frame.
`MAXinfo`	A data frame describing the blocks, with one row by block. The two (`POSIXct`) columns `"start"` and `"end"` provide the beginning and the end of the block. The numeric column `duration` gives the *effective duration* (in year) within block.
`probMissing`	A vector with values corresponding to the days in a block (year). Each value is a estimation of the probability that the day falls in a gap. If `dataFrames` is `FALSE`, the list still contains `probMissing` as before, as well as other lists as used in `Renouv`.
`effDuration`, `r`	Vectors containing the effective duration (*in years*) and number of value for the blocks.
`data`	List of maxima or r-largest values for the blocks.
`monthGapStat`, `monthGapTS`	Summary information concerning gaps, if `monthGapStat` is `TRUE` and if relevant information is provide via the the `OTmissing` formal. The element `monthGapTS` is a multivariate time series with yearly observations and one series (column) for each of the 12 months. Each series contains the missing fraction of the month for the considered year, ranging from `0.0` (no gap) to `1.0` (full gap). This object can be dealt with standard methods for time-series, but the `plot` method will require to select a reduced number of columns first, see **Examples**.

Note

Remind that even when `maxMissingFrac` is set to its maximum value 1.0, there can still be blocks with no data. When the result is intended to be used in the `Renouv` function, the formal `dataFrames` should be `FALSE`; the elements `data` and `effDuration` can then be passed as `MAX.data` and `MAX.effDuration`. At the time `infMAX` should also then be set to `FALSE` since `-Inf` values are not yet allowed in the r-largest values.

Examples

```
## use Dunkerque data
OTdata <- Dunkerque$OTdata; OTmissing <- Dunkerque$OTmissing
## allow up to 50\% gap in a block, or only 5\%
MAX1 <- OT2MAX(OTdata = OTdata, OTmissing = OTmissing,
               maxMissingFrac = 0.5,
               main = "impact of the 'maxMissingFrac' formal")
MAX2 <- OT2MAX(OTdata = OTdata, OTmissing = OTmissing, dataFrames = TRUE,
               prefix = "Max", maxMissingFrac = 0.05, plot = FALSE)
lines(MAX2$MAXdata$date, MAX2$MAXdata$Surge, type = "h", col = "red", lwd = 3)
legend("topleft", lw = c(1, 3), col = c("black", "orangered"),
       legend = c("50\% max", " 5\% max"))

## r-largest obs for r = 4
MAX3 <- OT2MAX(OTdata, OTmissing = OTmissing, MAX.r = 4,
               maxMissingFrac = 0.9,
               dataFrames = FALSE, trace = TRUE,
               main = "r-largest with r = 4")

## restrict the period
MAX4 <- OT2MAX(OTdata, OTmissing = OTmissing, MAX.r = 4,
               start = "1962-01-01",
               end = "1990-01-01",
               maxMissingFrac = 0.9,
               dataFrames = FALSE, trace = TRUE,
               main = "r-largest with r = 4 with given 'start' and 'end'")
## Not run:
  ## use in a block maxima analysis, as if there were no gaps.
  fit <- fGEV.MAX(MAX.data = MAX3$data,
                  MAX.effDuration = rep(1, length(MAX3$effDuration)))

## End(Not run)
## plot the gap rate
MAX5 <- OT2MAX(OTdata = OTdata, OTmissing = OTmissing,
               maxMissingFrac = 0.5,
               main = "probability of being in a  gap",
               plotType = "gap")

## time series plot (only <= 10 months)
plot(MAX5$monthGapTS[ , c(1:4)], main = "gap rate by month")

## much better with lattice.
## Not run:
    require(lattice)
    xyplot(MAX5$monthGapTS)

## End(Not run)
```

`OTjitter` *Add a small amount of noise to a numeric vector*

Description

Add a small amount of noise to a numeric vector keeping all the values above the given threshold.

Usage

```
OTjitter(x, threshold = NULL)
```

Arguments

`x`	The numeric vector to which *jitter* should be added.
`threshold`	A threshold above which all elements of the modified vector must stay.

Value

A vector with the same length and nearly the same values as `x`. As in `jitter`, a small amount of noise is added to each value of `x`. The noise level is adjusted so that every noisy value remains above the specified threshold. When the a value is very close to the threshold, only a very small amount of negative noise can be added.

Note

The aim of this function is to remove possible ties in experimental OT data. Ties cause problems or warnings in some goodness-of-fit tests such as Kolmogorov-Smirnov.

Author(s)

Yves Deville

See Also

`jitter`

Examples

```
## Garonne data (heavily rounded)
x <- Garonne$OTdata$Flow
min(x)
xmod <- OTjitter(x, threshold = 2500)
length(x)
nlevels(as.factor(x))
nlevels(as.factor(xmod))
max(abs(x-xmod))
```

`parDeriv` *Derivation of probability functions with respect to the parameters*

Description

Derivation of probability functions with respect to the parameters by using closed forms.

Usage

```
parDeriv(par, x, distname, sum = TRUE)
```

Arguments

`par`	Vector of parameter values.
`x`	Observations or data at which the derivatives are to be computed.
`distname`	Name of the distribution. See **Details**.
`sum`	Logical. If `TRUE`, a summation over the element of `x` is carried. Otherwise, the first dimension of the result corresponds to the elements of `x`.

Details

Only a few distributions are and will be available. For now, these are: the two-parameter Weibull `c("shape", "scale")`, the two-parameter Generalised Pareto, `c("scale", "shape")` and the two-parameter Lomax and maxlo distributions.

Value

A list of arrays containing the first and second order derivatives.

`derLogdens`, `der2Logdens`
Derivatives of the log-density $\log f(x)$.

`derSurv`, `der2Surv`
Derivatives of the survival function $S(x)$.

When `x` has length n and the distribution depends on p parameters, the arrays of first and second order derivatives have dimension $n \times p$ and $n \times p \times p$ when `sum` is `FALSE`. If `sum` is `TRUE` the summation drops the first dimension and the arrays are p and $p \times p$.

Author(s)

Yves Deville

References

See the *Renext Computing Details* document.

See Also

`Maxlo` and `Lomax`.

Examples

```
set.seed(1234)
distname <- "maxlo"
if (distname == "weibull") {
    logL <- function(par) {
        sum(dweibull(x, shape = par["shape"], scale = par["scale"], log = TRUE))
    }
    sumS <- function(par) {
        sum(pweibull(x, shape = par["shape"], scale = par["scale"],
                     lower.tail = FALSE))
    }
```

```
        pars <- c("shape" = rexp(1), "scale" = 1000 * rexp(1))
        x <- rweibull(n = 100, shape = pars["shape"], scale = pars["scale"])
        Der <- parDeriv(par = pars, x = x, distname = "weibull")
    } else if (distname == "gpd") {
        require(evd)
        logL <- function(par) {
            sum(dgpd(x, loc = 0, shape = par["shape"], scale = par["scale"],
                     log = TRUE))
        }
        sumS <- function(par) {
            sum(pgpd(x, loc = 0, shape = par["shape"], scale = par["scale"],
                     lower.tail = FALSE))
        }
        pars <- c("scale" = 1000 * rexp(1),
                  "shape" = runif(1, min = -0.4, max = 0.4))
        x <- rgpd(n = 100, loc = 0, shape = pars["shape"], scale = pars["scale"])
        Der <- parDeriv(par = pars, x = x, distname = "gpd")
    } else if (distname == "lomax") {
        logL <- function(par) {
            sum(dlomax(x, shape = par["shape"], scale = par["scale"], log = TRUE))
        }
        sumS <- function(par) {
            sum(plomax(x, shape = par["shape"], scale = par["scale"],
                       lower.tail = FALSE))
        }
        pars <- c( "shape" = 1 + rexp(1), "scale" = 1000 * rexp(1))
        x <- rlomax(n = 100, shape = pars["shape"], scale = pars["scale"])
        Der <- parDeriv(par = pars, x = x, distname = "lomax")
    } else if (distname == "maxlo") {
        logL <- function(par) {
            sum(dmaxlo(x, shape = par["shape"], scale = par["scale"], log = TRUE))
        }
        sumS <- function(par) {
            sum(pmaxlo(x, shape = par["shape"], scale = par["scale"],
                       lower.tail = FALSE))
        }
        pars <- c( "shape" = 2.5 + runif(1), "scale" = 100 * rexp(1))
        x <- rmaxlo(n = 100, shape = pars["shape"], scale = pars["scale"])
        Der <- parDeriv(par = pars, x = x, distname = "maxlo")
    }

    ## check logdens
    H <- numDeriv::hessian(func = logL, x = pars)
    colnames(H) <- names(pars)
    Grad <- numDeriv::grad(func = logL, x = pars)

    cat("gradient for log density\n")
    print(cbind(parDeriv = Der$derLogdens, num = Grad))

    cat("hessian for log density\n")
    print(cbind(exact = Der$der2Logdens, num = H))

    ## check survival
```

```
HS <- numDeriv::hessian(func = sumS, x = pars)
HS <- (HS + t(HS))/2
colnames(HS) <- names(pars)
GradS <- numDeriv::grad(func = sumS, x = pars)

cat("gradient for Survival\n")
print(cbind(parDeriv = Der$derSurv, num = GradS))

cat("hessian for Survival\n")
print(cbind(exact = Der$der2Surv, num = HS))
```

---

parIni.MAX *Initial estimation of GPD parameters for an aggregated renewal model*

Description

Initial estimation for an aggregated renewal model with GPD marks.

Usage

```
parIni.MAX(MAX, threshold, distname.y = "exp")
parIni.OTS(OTS, threshold, distname.y = "exp")
```

Arguments

`MAX`	A list describing partial observations of `MAX` type. These are block maxima or block r-largest statistics. The list must contain elements named `block`, `effDuration`, `r`, and `data` (a by-block list of r-largest statistics).
`OTS`	A list describing partial observations of `OTS` type. These are observations above the block thresholds, each being not smaller than `threshold`. This list contains `block`, `effDuration`, `threshold`, `r` and `data` (a by-block list of observations).
`threshold`	The threshold of the POT-renewal model. This is the location parameter of the marks from which Largest Order Statistics are observed.
`distname.y`	The name of the distribution. For now this can be `"exp"` or `"exponential"` for exponential excesses implying Gumbel block maxima, or `"gpd"` for GPD excesses implying GEV block maxima. The initialisation is the same in all cases, but the result is formatted according to the target distribution.

Details

The functions estimate the Poisson rate `lambda` along with the shape parameter say `sigma` for exponential excesses. If the target distribution is GPD, then the initial shape parameter is taken to be 0.

In the "MAX" case, the estimates are obtained by regressing the maxima or r-Largest Order Statistics within the blocks on the log-duration of the blocks. The response for a block is the minimum of the r available Largest Order Statistics as found within the block, and r will generally vary across block. When some blocks contain $r > 1$ largest order statistics, the corresponding spacings are used to compute a spacings-based estimation of σ. This estimator is independent of the regression estimator for σ and the two estimators can be combined in a weighted mean.

In the "OTS" case, the estimate of `lambda` is obtained by a Poisson regression using the log durations of the blocks as an offset. The estimate of `sigma` is simply the mean of all the available excesses, which by assumption share the same exponential distribution.

Value

A vector containing the estimate of the Poisson rate λ, and the estimates of the parameters for the target distribution of the excesses. For exponential excesses, these are simply a `rate` parameter. For `GPD` excesses, these are the `scale` and `shape` parameters, the second taken as zero.

Note

In the MAX case, the estimation is possible only when the number of blocks is greater than 1, since otherwise no information about λ can be gained from the data; recall that the time at which the events occurred within a block is not known or used.

Author(s)

Yves Deville

References

See the document *Renext Computing Details*.

See Also

The `spacings` methods for the spacings used in the estimation.

Examples

```
set.seed(1234)
## initialisation for 'MAX' data
u <- 10
nBlocks <- 30
nSim <- 100
ParMAX <- matrix(NA, nrow = nSim, ncol = 2)
colnames(ParMAX) <- c("lambda", "sigma")

for (i in 1:nSim) {
  rd <- rRendata(threshold = u,
                 effDuration = 1,
                 lambda = 12,
                 MAX.effDuration = c(60, rexp(nBlocks)),
                 MAX.r = c(5, 2 + rpois(nBlocks, lambda = 1)),
                 distname.y = "exp", par.y = c(rate = 1 / 100))

  MAX <- Renext:::makeMAXdata(rd)
  pari <- parIni.MAX(MAX = MAX, threshold = u)
  ParMAX[i, ] <- pari
}
## the same for OTS data
u <- 10
nBlocks <- 10
nSim <- 100
ParOTS <- matrix(NA, nrow = nSim, ncol = 2)
colnames(ParOTS) <- c("lambda", "sigma")
rds <- list()

for (i in 1:nSim) {
  rd <- rRendata(threshold = u,
                 effDuration = 1,
                 lambda = 12,
                 OTS.effDuration = rexp(nBlocks, rate = 1 / 10),
                 OTS.threshold = u + rexp(nBlocks, rate = 1 / 10),
                 distname.y = "exp", par.y = c(rate = 1 / 100))
  rds[[i]] <- rd
  OTS <-  Renext:::makeOTSdata(rd)
  pari <- parIni.OTS(OTS = OTS, threshold = u)
  ParOTS[i, ] <- pari
}
```

pGreenwood1 *Probability that the Greenwood's statistic is smaller than one*

Description

Probability that the Greenwood's statistic is smaller than one.

Usage

```
pGreenwood1(n)
```

Arguments

n	Sample size.

Details

The probability was computed by using the approximation of the quantile function of the Greenwood's statistic returned by `qStat`. The result is found by interpolating the distribution function for $x = 1$.

Value

Probability that the Greenwood's statistic is smaller than one. For a random sample of an exponential distribution with size n, this is the probability that the coefficient of variation is less than one, or the probability that the ML estimate of the GPD shape parameter ξ is negative.

Author(s)

Yves Deville

Examples

```
n <- 8:500
plot(n, pGreenwood1(n), type = "l", col = "orangered", lwd = 2,
     log ="x", ylim =c(0.5, 0.7), main = "slow convergence to 0.5")
grid() ; abline(h = 0.5, col = "SpringGreen")
```

---

plot.Rendata *Plot a Rendata object*

Description

Plot 'Rendata' datasets with OT and historical data

Usage

```
    ## S3 method for class 'Rendata'
plot(x,
     textOver = quantile(x$OTdata[, x$info$varName], probs = 0.99),
     showHist = TRUE,
                 ...)
```

Arguments

`x`	Rendata object i.e. a list object as read with the `readXML` function.
`textOver`	Mark values of the variable in the `OTdata` part of `x`. Values above the `textOver` value (if any) will be marked with the character version of the block, typically a year
`showHist`	If `TRUE`, the historical periods (is any) are shown on the plot.
`...`	further args to be passed to `plot` function.

Details

The plot shows the main data of the object `x` (the `OTdata` part) as well as historical data `MAXdata` or `OTSdata` if any. Different colours are used on the background. This function is not intended to produce nice plots to be printed.

Note

This function is mainly a companion function of `readXML`. Its goal is to check the content of the data read.

Author(s)

Yves Deville

See Also

readXML

Examples

```
if (require(XML)) {
   ## use 'index.xml' file shipped with Renext
   dir1 <- system.file("Rendata", package = "Renext")
   BrestNew <- readXML(name = "Brest", dir = dir1)
   plot(BrestNew)
   GaronneNew <- readXML(name = "Garonne", dir = dir1)
   plot(GaronneNew)
   test1 <- readXML(name = "test1", dir = dir1)
   plot(test1)
}
```

---

plot.Renouv — *Plot an object of class "Renouv"*

Description

Plot an object of class "Renouv". The plot is a return level plot with some supplementary elements to display historical data.

Usage

```
## S3 method for class 'Renouv'
plot(x,
     pct.conf = x$pct.conf,
     show = list(OT = TRUE, quant = TRUE, conf = TRUE,
                 MAX = TRUE, OTS = TRUE),
     mono = TRUE,
     predict = FALSE,
     par = NULL,
     legend = TRUE,
     label = NULL,
     problim = NULL,
     Tlim = NULL,
     main = NULL, xlab = "periods", ylab = "level",
     posOptions = NULL,
     byBlockStyle = NULL,
     ...)
## S3 method for class 'Renouv'
lines(x,
      pct.conf = x$pct.conf,
      show = NULL,
      mono = TRUE,
      predict = FALSE,
      par = NULL,
      legend = FALSE,
      label = NULL,
      posOptions = NULL,
      byBlockStyle = NULL,
      ...)
```

Arguments

`x`	Object of class `"Renouv"`.
`pct.conf`	Percents for confidence limits (lower and upper). These levels should be found within those computed in the object `x`. By default, all computed levels will be used.
`show`	A *list with named elements* specifying which parts of the return level plot must be drawn. Element `OT` is for the the sample points (Over the Threshold data), `quant` is for the quantile curve (or Return Level curve), `conf` is for the confidence limits. These three elements can be set to `TRUE` or `FALSE`. When the element `conf` is `TRUE`, only the percent levels given in `pct.conf` are drawn. Moreover, the levels not already computed in the object given in `x` will be drawn only if the predictions are recomputed with `predict` set to `TRUE`. Finally, the `MAX` and `OTS` elements are for the two possible types of historical data. They can be logical vectors with length one or with length equal to the corresponding of blocks if only some blocks are to be shown. These two elements can also be character vectors indicating the names of the blocks which are to be shown (by partial matching). These names should match one or several elements of the character vector named `blockNames` within the lists `x$history.MAX` or `x$history.OTS` respectively.
`mono`	Logical, `TRUE` for a monochrome plot.
`predict`	Logical. When `TRUE`, predictions are re-computed from the model before plotting. One effect is that the points used to draw the curves are designed to cover the whole range (if specified by the user). One other effect is that the confidence limits are recomputed in order to include the percent levels given on entry.
`par`	A list such as returned by the `RLpar` function.
`legend`	Logical. If `TRUE`, a legend is built and drawn on the graph.
`label`	A character label used to build the labels used in the legend. The default is to use the name of the `x` object. Using an empty string `""` can be better in some cases.
`problim`	Limits for the x-axis in probability scale. Can be used as an alternative to `Tlim`.
`Tlim`	Limits for the x-axis in return period scale. The values are given as a numeric vector of length 2, containing values ≥ 1. The first element (minimal return period can be 0 in which case it will be replaced by a very small positive value.
`xlab`	Label of the x-axis (time periods, with log scale).

`main`	Main title (character).
`ylab`	Label of the y-axis (labels).
`posOptions`	A pair list to be passed as list of formals to the `SandT` function computing the plotting positions.
`byBlockStyle`	Logical list (or named logical vector) with elements `MAX` and `OTS`. The value indicates if each (MAX or OTS) block must be plotted with a specific style (plotting character and color), or if instead a common style is used for all blocks of the same type (MAX or OTS). In the first case, each block will create a line in the legend with a label taken from the `history.MAX` element of the object given in `x`. These legend lines will not appear if `legend` is `FALSE` but can be shown later using `RLlegend.show`. In the second case, only one legend line will be generated. When the number of blocks is large for one type and the corresponding value of `byBlockStyle` is `TRUE`, the styles will be recycled and the plot/legend might not be clear. When `byBlockStyle` is `NULL` or does not contain all needed information, default choices are made.
`...`	Other arguments passed to the default `plot` function e.g., `ylim` to adjust the y-axis.

Details

Historical data blocks (MAX or OTS) embedded in the `x` object (if any) can be plotted or not depending on the value of the corresponding element in `show`.

- If the `MAX` element is `TRUE` and if `x` embeds historical data of type `MAX`, then these will be shown with a symbol differing from the one for ordinary points.
- If `OTS` element is `TRUE` and is `x` embeds historical data of type `OTS`, then these will be shown with a symbol differing from the one for ordinary points. An exception is when one or several OTS block have no data. Then each such block is shown as an horizontal segment; its right end-point shows the effective duration of the block and the ordinate shows the OTS threshold for this block. No data exceeded the threshold within the block.

This function acts on a list variable named `.RLlegend` and stored in a special environment bound to the package. This variable is used to build legends for plots produced with multiple commands. See the `RLlegend` help page. Examples of possible combined uses of the argument of the `plot` and `lines` together with the `RLlegend*` functions are given in the "Renext Graphics" chapter of the *Renext Guide* document shipped with this package.

Value

No value returned.

Caution

Remind that the methods `plot` and `lines` may change the value of the variable `.RLlegend` in the environment `legendEnvir`. This variable describes the material to be used in the legend at the next call of `RLlegend.show`.

Note

The return level plot is of exponential type i.e. uses a log-scale for return periods. This contrasts with the Gumbel plot which is also used in similar contexts.

Author(s)

Yves Deville

See Also

The `RLlegend` page for the legend construction and `RLpar` to specify the graphical parameters (colors, line types, ...) of the elements.

Examples

```
## two fits for the Garonne data
fit.exp <- Renouv(x = Garonne, plot = FALSE)
fit.gpd <- Renouv(x = Garonne, distname.y = "gpd", plot = FALSE)

## simple plot (legend is TRUE here)
plot(fit.exp,
     main = "Two fits overlapped",
     label = "",
     ## Tlim = c(1, 5000),
     predict = TRUE)

## Now try 'lines' and RLlegend.xxx functions
plot(fit.exp,
     main = "Fancy legend",
     show = list(OT = FALSE, quant = FALSE, conf = FALSE,
                 OTS = FALSE, MAX = FALSE),
     legend = FALSE,
     Tlim = c(1, 5000))
RLlegend.ini(x = "bottomright", bg = "lightyellow") ## initialise legend
lines(fit.exp,
      show = list(quant = FALSE, conf = FALSE, OT = TRUE, MAX = TRUE),
      label = "expon",
      par = RLpar(quant.col = "orange",
        OT.pch = 21, OT.cex = 1.2, OT.col = "SeaGreen", OT.bg = "yellow",
        MAX.block1.col = "purple", MAX.block1.bg = "mistyrose",
        MAX.block1.lwd = 1.4))
lines(fit.gpd,
      pct.conf = c(95, 70),
      show = list(quant = TRUE, conf = TRUE),
      label = "GPD",
      par = RLpar(quant.col = "darkcyan", conf.conf1.col = "red"))
RLlegend.show() ## now draw legend
```

PPplot *Diagnostic plots for Renouv objects*

Description

Diagnostic plots for Renouv objects.

Usage

```
PPplot(x, ...)

QQplot(x, ...)

## S3 method for class 'Renouv'
PPplot(x,
       showHist = FALSE,
       legend = FALSE,
       par = NULL,
       posOptions = NULL,
       ...)

## S3 method for class 'Renouv'
QQplot(x,
       showHist = FALSE,
       legend = FALSE,
       par = NULL,
       posOptions = NULL,
       ...)
```

Arguments

x	Object containing a fitted model.
legend	Should a legend be shown to identify historical blocks? NOT IMPLEMENTED YET.
par	A list of graphical parameters as returned by RLpar and used to control the appearance of points. NOT IMPLEMENTED YET.
posOptions	A pair list to be passed as list of formals to the SandT function computing the plotting positions.
showHist	If TRUE, historical information contained in the object x (if any) will be shown using special plotting positions as computed by SandT.
...	Other arguments to be passed to plot.

Value

No value returned.

Author(s)

Yves Deville

See Also

SandT for the computation of the plotting positions used with historical data.

---

predict.Renouv *Compute return levels and confidence limits for a "Renouv" object*

Description

Compute return levels and confidence limits for an object of class "Renouv".

Usage

```
   ## S3 method for class 'Renouv'
predict(object,
        newdata = c(10, 20, 50, 100, 200, 500, 1000),
        cov.rate = TRUE,
        level = c(0.95, 0.7),
        prob = FALSE,
        trace = 1, eps = 1e-06,
        ...)
```

Arguments

`object`	An object of class `"Renouv"` typically created by using the `Renouv` function.
`newdata`	The return period at which return levels and confidence bounds are wanted.
`cov.rate`	If `FALSE`, the delta method will not take into account the uncertainty on the event rate `lambda` of the Poisson process. Note however that when `distname.y` is `"exponential"` and when no `MAX` or `OTS` data is used, the value of `cov.rate` has no impact for now, because the delta method is not used then.
`level`	Confidence levels as in other 'predict' methods (not percentages).
`prob`	If `TRUE` a `prob` column is found in the returned data frame. This column can be used to find which quantile was used to compute the return level.
`trace`	Some details are printed when `trace` is not zero.
`eps`	Level of perturbation used to compute the numerical derivatives in the delta method.
`...`	Further arguments passed to or from other methods.

Details

Unless in some very special cases, the confidence limits are approximated ones computed by using the delta method with numerical derivatives.

Value

A data frame with the expected return levels (col. named "quant") at the given return periods, and confidence limits. The returned object has an infer.method attribute describing the method used to compute the confidence limits.

Note

Despite of its name, this method does not compute true predictions. A return period is to be interpreted as an average interevent time rather than the duration of a specific period of time. For instance, the expected return level for a given return period with length 100 years is the level that would be on average exceeded once every 100 years (assuming that the model description in object is correct).

Author(s)

Yves Deville

References

Coles S. (2001) *Introduction to Statistical Modelling of Extremes Values*, Springer.

See Also

Renouv to fit Renouv model.

Examples

```
## Use Brest data
fit <- Renouv(Brest)
pred <- predict(fit, newdata = c(100, 125, 150, 175, 200),
                level = c(0.99, 0.95))
```

---

qStat *Quantiles of a test statistic*

Description

Quantile of a test statistic.

Usage

```
qStat(p, n,
      type = c("Greenwood", "Jackson", "logLRGPD", "logLRLomax",
               "logLRGEV", "logLRFrechet"),
       outNorm = FALSE)
```

Arguments

`p`	Numeric vector of probabilities. Very small values (`p < 0.01`) or very large ones (`p > 0.99`) will be truncated as `0.00` or `1.00` to maintain a realistic level of precision.
`n`	Sample size.
`type`	The type of statistic, see **Details**.
`outNorm`	Logical. If `TRUE` the output is normalized in a such fashion that its distribution is the asymptotic one (i.e. standard normal in practice). When `FALSE`, the quantiles are given in the true scale of the statistic: CV^2, Jackson. For LR statistics this argument has no impact.

Details

The function provides an approximation of the distribution for several statistics.

- For `"Greenwood"`, the statistic is *Greenwood's statistic*. The distribution is that of the squared coefficient of variation CV^2 of a sample of size `n` from the exponential distribution as computed by `CV2`.
- For `"Jackson"`, the statistic is Jackson's statistic, see `Jackson`.
- For `"logLRGPD"` and `"logLRLomax"`, the statistic is the log of the likelihood ratio of a sample from the exponential distribution. The log-likelihoods are for an exponential distribution compared to a GPD with non-zero shape, or to a GPD with *positive shape* (equivalently, a Lomax distribution).
- For `"logLRGEV"` and `"logLRFrechet"`, the statistic is the log of the likelihood ratio of a sample from the Gumbel distribution. The log-likelihoods are for a Gumbel distribution compared to a GEV with non-zero shape, or to a GEV with *positive shape* (equivalently, a Fréchet distribution).

The log of Likelihood Ratios are multiplied by `2`, so that they compare to a chi-square statistic with one degree of freedom.

Value

A vector of quantiles.

Note

The precision of the result given is limited, and is about two-digits. This function is not intended to be used as such and is only provided for information.

Author(s)

Yves Deville

Examples

```
res <- qStat(n = 40, type = "Greenwood")
plot(res$q, res$p, type = "o")
```

`readXML` *Read data using an XML index file*

Description

Read one or several dataset(s) using an XML index file specifying the data sets to read and the structure of each

Usage

```
readXML(name,
        dir,
        index = "index.xml",
        TZ = "GMT",
        trace = 0)
```

Arguments

name	Name for the dataset that will be matched against the name attribute of datasets as they are given in the index file.
dir	Path to the directory where the index file and all data files should be found.
index	Name (short) of the index file. This file must be in the directory given by dir.
TZ	A time zone as in strptime. The time zone "GMT" should be preferred, since it should work on all platforms and can cope with dates in the remote past.
trace	Level of verbosity (integer). Can be used to trace the successive steps by short indications.

Details

The XML index file is parsed within R. Then according to the indications within the index file, other files are read (e.g. csv files). In this way, data returned as a list can contain heterogeneous data: Over Threshold (OT) data, missing periods, MAX data, etc. Various pieces of information are also stored in list elements with name containing the "info" string.

This function requires the CRAN package XML.

Value

A list with the data read.

info	General information about the data: varName, varShorlLab and varUnit give the variable name unit and short label.
OTinfo	Information for the Over the Threshold (OT).
OTdata	Over the Threshold (OT) data.
OTmissing	Missing periods within the OTdata period.
MAXinfo	Information for the MAX (r-largest) supplement data.

`MAXdata`	MAX supplement data.
`OTSinfo`	Information for the Over the Threshold Supplement (OTS) data.
`OTSdata`	Over the Threshold (OT) supplement data.

Note

The flat files (usually `.csv` files) can also be read in a more conventional way e.g. through `read.table`. However, conform to the `index.xml` examples or to the `index.xsd` schema to see how to adjust the reading of parameters such as `sep`, etc.

Author(s)

Yves Deville

See Also

See `Brest` for an example of such a list.

Examples

```
## Not run:
## Examples of datasets that can be read
## Browse should work for browsers with xslt support
browseURL(file.path(system.file("Rendata", package = "Renext"), "index.xml"))
if (require(XML)) {
   ## use 'index.xml' file shiped with Renext
   dir1 <- system.file("Rendata", package = "Renext")
   BrestNew1 <- readXML(name = "Brest", dir = dir1)
   test1 <- readXML(name = "test1", dir = dir1)
}

## End(Not run)
```

`Ren2gev` *Translate a vector of coefficients from a Renewal-POT model with Pareto excesses into a vector of GEV parameters*

Description

Translate a vector of coefficients from a Renewal-POT model with Pareto excesses into a vector of GEV parameters.

Usage

```
Ren2gev(object,
        threshold = NULL,
        w = 1,
        distname.y = c("gpd", "GPD", "lomax", "maxlo"),
        jacobian = (length(w) == 1L),
        vcovRen = NULL)
```

Arguments

`object`	A named vector of parameters or an object of class `"Renouv"`. In the first case, the names of the vector element must conform to the distribution given in `distname.y`.
`threshold`	The threshold associated with the renewal-POT model. This must be provided and be a non NA finite numeric value. It is the location parameter of the GPD.
`w`	The duration of the blocks.
`distname.y`	The distribution of the excesses in the renewal-POT model. This is normally a `"gpd"` but can be a `"lomax"` or a `"maxlo"` distribution provided that the GEV parameters given in `object` specify a positive or a negative shape respectively.
`jacobian`	Logical. If `TRUE`, the jacobian matrix of the transformation is computed. This is only possible at the time when `w` has length 1.
`vcovRen`	A covariance matrix for the "Ren" vector of parameters. If `object` has class `"Renouv"`, then the covariance matrix embedded in the object is used.

Details

Given Renewal-POT parameters, it is possible to compute the distribution of block maxima. When the distribution is in the Pareto family, the marginal distribution of maxima is GEV. The location and the scale GEV parameters depend on the block duration w, while the GEV shape parameter is identical to that of the GPD input distribution.

Value

When `w` has length `1`, a named vector of GEV parameters as the one estimated by `fgev`. This vector has an elements named `"loc"`, `"scale"` and `"shape"`.

When `w` has length > `1`, a matrix with `length(w)` rows, each representing a vector of GEV parameters as before.

The returned object has attributes named `"threshold"`. and `"distname.y"` to recall how it was built.

Author(s)

Yves Deville

See Also

The `gev2Ren` function provides a reciprocal transformation.

Examples

```
fit1 <- Renouv(Garonne, distname.y = "maxlo")
Ren2gev(fit1)
fit2 <- Renouv(Garonne, distname.y = "gpd")
Ren2gev(fit2)
```

`Ren2gumbel` *Translate a vector of coefficients from a Renewal-POT model with exponential excesses to a vector of Gumbel parameters*

Description

Translate a vector of coefficients from a Renewal-POT model with exponential excesses to a vector of Gumbel parameters.

Usage

```
Ren2gumbel(object,
           threshold = NULL,
           w = 1,
           distname.y = c("exponential", "exp"),
           jacobian = (length(w) == 1L),
           vcovRen = NULL)
```

Arguments

`object`	A named vector of parameters or an object of class `"Renouv"`. In the first case, the names of the vector element must conform to the exponential distribution so the vector must be of length 2 with names `"lambda"` and `"rate"`.
`threshold`	A threshold associated with the parameters. If `object` is an object with class `"Renouv"`, its threshold slot will be used.
`w`	A block duration or a vector of block durations.
`distname.y`	The name of the distribution for the excesses. Can be either `"exponential"` or `"exp"`. The choice has no impact on the computations, but this name will be attached to the result as an attribute and may affect later use.
`jacobian`	Logical. If `TRUE` the jacobian matrix of the transformation will be computed and attached to the result as an attribute.
`vcovRen`	A covariance matrix for the Renouv parameters.

Value

A vector of GEV parameters if `w` has length 1, and a matrix if `w` has length `> 1`. The returned objects has attributes.

Author(s)

Yves Deville

See Also

`Ren2gev` for the translation of Renouv parameters corresponding to GPD excesses.

Examples

```
## Fit a Renouv model with exponential excesses (default)
fit <- Renouv(Garonne)
## Convert to gumbel (usable for one-year block maxima)
parGumbel <- Ren2gumbel(fit)
## Retrieve the 'Renouv' model by giving the right threshold
parRen <- gumbel2Ren(parGumbel,
                     threshold = 2500,
                     vcovGumbel = attr(parGumbel, "vcov"),
                     plot = TRUE)
## Build a compatible model under the assumption of one event by
## year
parRen2 <- gumbel2Ren(parGumbel,
                      lambda = 1.00,
                      vcovGumbel = attr(parGumbel, "vcov"),
                      plot = TRUE)
parRenNames <- c("lambda", "rate")
## Build a 'Renouv' object without estimation
myVcov <- attr(parRen, "vcov")[parRenNames, parRenNames]
fitNew <- RenouvNoEst(threshold = attr(parRen, "threshold"),
                      estimate = parRen,
                      distname.y = "exp",
                      cov = myVcov)
## Compare return levels
cbind(roundPred(fit$pred)[ , -2], roundPred(fitNew$pred)[ , -2])
## idem for the putative 'Renouv' with rate 1
myVcov2 <- attr(parRen2, "vcov")[parRenNames, parRenNames]
fitNew2 <- RenouvNoEst(threshold = attr(parRen2, "threshold"),
                       estimate = parRen2,
                       distname.y = "exp",
                       cov = myVcov2)
cbind(roundPred(fit$pred)[ , -2], roundPred(fitNew2$pred)[ , -2])
```

Renouv *Fit a 'Renouvellement' model*

Description

Fit a 'renouvellement' POT model using Over the Threshold data and possibly historical data of two kinds.

Usage

```
Renouv(x,
       threshold = NULL,
       effDuration = NULL,
       distname.y = "exponential",
       MAX.data = NULL,
       MAX.effDuration = NULL,
       OTS.data = NULL,
       OTS.effDuration = NULL,
       OTS.threshold = NULL,
       fixed.par.y = NULL,
       start.par.y = NULL,
       force.start.H = FALSE,
       numDeriv = TRUE,
       trans.y = NULL,
       jitter.KS = TRUE,
       pct.conf = c(95, 70),
       rl.prob = NULL,
       prob.max = 1.0-1e-04 ,
       pred.period = NULL,
       suspend.warnings = TRUE,
       control = list(maxit = 300, fnscale = -1),
       control.H = list(maxit = 300, fnscale = -1),
       trace = 0,
       plot = TRUE,
       label = "",
       ...)
```

Arguments

`x`	Can be a numeric vector, an object of the class `"Rendata"` or `NULL`. In the first case, `x` contains all the levels above the threshold for a variable of interest. In the second case, most formal arguments take values in accordance with the object content, and can be by-passed by giving the formal explicitly. When `x` is `NULL`, the model is fitted using the data provided using the `OTS` and `MAX` formals.
`threshold`	Value of the threshold for the OT data.
`effDuration`	Effective duration, i.e. duration of the OT period.
`distname.y`	Name of the distribution for the excesses over the threshold. See **Details** below.
`MAX.data`	Either a numeric vector or a list of numeric vectors representing historical data r-max by blocks. When *a vector* is given, there is only one block, and the data are the corresponding r-max observed levels where r is the vector length; the block duration is given in `MAX.effDuration`. When *a list* is given, each list element contains the data for one block, and the effective duration are in `MAX.effDuration`
`MAX.effDuration`	Vector of (effective) durations, one by block MAX data.
`OTS.data`	A numeric vector or a list of numeric vectors representing supplementary Over Threshold data in blocks. When a *vector* is given, there is only one block, and the data contain all the 'historical' levels over the corresponding threshold given in `OTS.threshold`. The block duration is given in `OTS.effDuration`. When a *list* is given, each list element contains the data for one block, and the threshold and effective duration are in `OTS.threshold` and `OTS.effDuration`.
`OTS.effDuration`	A numeric vector giving the (effective) durations for the OTS blocks.

`OTS.threshold`	A vector giving the thresholds for the different OTS blocks. The given values must be greater than or equal to the value of `threshold`.
`fixed.par.y`	Named list of known (or fixed) parameter values for the `y`-distribution.
`start.par.y`	Named list of parameter initial values for the `y`-distribution. Only used when the distribution does not belong to the list of special distributions.
`force.start.H`	Logical. When `TRUE`, the values in `start.par.y` (which must then be correct) will be used also as starting values in the maximisation of the global likelihood : OT data and historical data. This is useful e.g. when the historical data fall outside of the support for the distribution fitted without historical data. See below the **Details** section.
`numDeriv`	Logical: should the hessian be computed using the `numDeriv` package (value `TRUE`) or should it be taken from the results of `optim`?
`trans.y`	Transformation of the levels *before thresholding* (if not `NULL`). This is only possible with the `"exponential"` value `distname.y`. The two allowed choices are `"square"` and `"log"` meaning that the fitted (exponentially distributed) values are `x.OT^2 -threshold^2` and `log(x.OT) -log(threshold)` respectively. The corresponding distributions for `x.OT` may be called "square-exponential" and "log-exponential".
`jitter.KS`	Logical. When set to `TRUE`, a small amount of noise is added to the "OT" data used in the Kolmogorov-Smirnov test in order to remove ties. This is done using the `OTjitter` function.
`pct.conf`	Character or numeric vector specifying the percentages for the confidence (bilateral) limits on quantiles.
`rl.prob`	Vector of probabilities for the computation of return levels. These are used in plots (hence must be dense enough) and appear on output in the data.frame `ret.lev`.
`prob.max`	Max value of probability for return level and confidence limits evaluations. This argument 'shortens' the default `prob` vector: values > `prob.max` in the default `prob` vector are omitted. Ignored when a `prob` argument is given.
`pred.period`	A vector of "pretty" periods at which return level and probability will be evaluated and returned in the `pred` data.frame.
`suspend.warnings`	Logical. If `TRUE`, the warnings will be suspended during optimisation steps. This is useful when the parameters are subject to constraints as is usually the case.
`control`	A named list used in `optim` for the no-history stage (if any). Note that `fnscale = -1` says that maximisation is required (not minimisation) and must not be changed!
`control.H`	A named list used in `optim` for the historical stage (if any).
`trace`	Level of verbosity. Value `0` prints nothing.
`plot`	Draw a return level plot?
`label`	Label to be used in the legend when `plot` is `TRUE`.
`...`	Arguments passed to `plot.Renouv`, e.g. `main`, `ylim`.

Details

The model is fitted using Maximum Likelihood (ML).

Some distributions listed below and here called "special" are considered in a special manner. For these distributions, it is not necessary to give starting values nor parameter names which are unambiguous.

distribution	parameters
`exponential`	`rate`
`weibull`	`shape`, `scale`
`GPD`	`scale`, `shape`
`gpd`	`scale`, `shape`
`lomax`	`scale`, `shape`
`maxlo`	`scale`, `shape`
`log-normal`	`meanlog`, `sdlog`
`gamma`	`shape`, `scale`
`mixexp2`	`prob1`, `rate1`, `delta`

Other distributions can be used. Because the probability functions are then used in a "black-box" fashion, these distributions should respect the following *formal requirements*:

1. The name for the *density*, *distribution* and *quantile* functions must obey to the *classical "prefixing convention"*. Prefixes must be respectively: `"d"`, `"p"`, `"q"`. This rules applies for distribution of the `stats` package and those of many other packages such `evd`.
2. *The first (main) argument must be vectorisable* in all three functions, i.e. a vector of `x`, `q` or `p` must be accepted by the density, the distribution and the quantile functions.
3. *The density must have a* `log` *formal* argument. When `log` is `TRUE`, the log-density is returned instead of the density.

For such a distribution, it is necessary to give arguments names in `start.par.y`. The arguments list must have exactly the required number of parameters for the family (e.g. `2` for `gamma`). Some parameters can be fixed (known); then the parameter set will be the reunion of those appearing in `start.par.y` and those in `fixed.par.y`. Anyway, in the present version, *at least one parameter must be unknown* for the `y` part of the model.

Mathematical requirements exist for a correct use of ML. They are referred to as "regularity conditions" in ML theory. Note that the support of the distribution must be the set of non-negative real numbers.

The estimation procedure differs according to the existence of the different types of data: main sample, MAX and OTS.

1. When no historical data is given, the whole set of parameters contains orthogonal subsets: a "point process" part concerning the process of events, and an "observation" part concerning the excesses over the threshold. The parameters can in this case be estimated separately. The rate of the Poisson process is estimated by the empirical rate, i.e. the number of events divided by the total duration given in `effDuration`. The "Over the Threshold" parameters are estimated from the excesses computed as `x.OT` minus the threshold.

2. When historical data is given, the two parameter vectors must be coped with together in maximising the global likelihood. In this case, we begin the estimation ignoring the historical data and then use the estimates as starting values for the maximisation of the global likelihood. In some circumstances, the estimates obtained in the first stage can not be used with historical data because some of these fall outside the support of the distribution fitted. This can happen e.g. with a `distname.y = "gpd"` when historical data exceed `threshold - scale/shape` for the values of `shape` and `scale` computed in the first stage.
3. From version 2.1-1 on, it is possible to use `OTS` and/or `MAX` data with no `OT` data by specifying `x = NULL`. Yet at the time this is only possible `distname.y` takes one of the two values: `"exp"`, or `"gpd"`. The initial values for the parameter are then obtained by using the `parIni.OTS`, `parIni.MAX` functions and possibly by combining the two resulting initial parameter vectors. This possibility can be used to fit a model from *block maxima* or r*-largest* classical data but with more flexibility since the duration of the blocks may here not be constant.

The returned `Renouv` object contains a `MAX` element concerning the distribution of block maxima in the two following cases.

1. When `distname.y` is `"exponential"` or `"exp"`, the distribution of the maximum is Gumbel. The estimated parameters can be used with the `gumbel` function of the **evd** package.
2. When `distname.y` is `"gpd"`, `"lomax"`, `"maxlo"` or `"GPD"` the distribution of the maximum is a Generalised Extreme Values distribution. The estimated parameters can be used with the `gev` function of the **evd** package.

Value

An object with class `"Renouv"`. This is mainly a list with the various results.

`est.N`	Estimate(s) for the count `"N"` part. This estimate does not use the historical data, even if is available.
`est.y`	Estimate(s) for the excess `"y"` part. This estimate does not use the historical data, even if available.
`cov.N`, `cov.y`	The (co-)variances for the estimates above.
`estimate`	Estimate(s) for the whole set of parameters based on OT data **and on historical data** if available.
`ks.test`	Kolmogorov-Smirnov goodness-of-fit test.
`ret.lev`	A data frame containing return levels and confidence limits. The corresponding probabilities are either provided by user or taken as default values.
`pred`	A data frame similar to `ret.lev`, but with "pretty" return periods. These are taken as the provided values `pred.period` if any or are chosen as "round" multiples of the time unit (taken from `effDuration`). The periods are chosen in order to cover periods ranging from 1/10 to 10 time units.
`MAX`	A list providing the estimated distribution of the maximum of the marks over a block of unit duration. This list element only exists when this distribution can be deduced from the fit, which is the case when `distname.y` is a GPD in a broad sense, see **Details**.

Other results are available. Use `names(result)` to see their list.

Except in the the special case where `distname.y` is `"exponential"` and where no historical data are used, the inference on quantiles is obtained with the *delta method* and using numerical derivatives. Confidence limits are unreliable for return levels much greater than the observation-historical period.

Due to the presence of estimated parameters, the Kolmogorov-Smirnov test is unreliable when less than 30 observations are available.

Warning

With some distributions or in presence of historical data, the estimation can fail due to some problem during the optimisation. Even when the optimisation converges, the determination of the (numerical) hessian can be impossible: This can happen if *one or more parameter is too small* to compute a finite difference approximation of gradient. For instance the 'rate' parameter of the exponential distribution (= inverse mean) will be small when the mean of the excesses is large.

A possible solution is then to **rescale the data** e.g. dividing them by 10 or 100. As a rule of thumb, an acceptable scaling leads to data (excesses) of a few units to a few hundreds, but **an order of magnitude of thousands or more should be avoided and reduced by scaling**. The rescaling is recommended for the square exponential distribution (obtained with `trans = "square"`) since the observations are squared.

Another possible way to solve the problem is to change the `numDeriv` value.

Note

The model only concerns the "Over the Threshold" part of the distribution of the observations. When historical data is used, observations should all be larger than the threshold.

The name of the elements in the returned list is indicative, and is likely to be changed in future versions. At the time, the effect of historical data on estimation (when such data exist) can be evaluated by comparing `c(res$est.N, res$est.y)` and `res$estimate` where `res` is the results list.

Some warnings may indicate that missing values are met during the optimisation process. This is due to the evaluation of the density at tail values. At the time the ML estimates are computed using an unconstrained optimisation, so invalid parameter values can be met during the maximisation or even be returned as (invalid) estimates.

Author(s)

Yves Deville

References

- Miquel J. (1984) *Guide pratique d'estimation des probabilités de crues*, Eyrolles (coll. EDF DER).
- Coles S. (2001) *Introduction to Statistical Modelling of Extremes Values*, Springer.
- Embrechts P., Klüppelberg C. and Mikosch T. (1997) *Modelling Extremal Events for Insurance and Finance*. Springer.

See Also

RLplot for the *return level plot*. See optim for the tuning of the optimisation. The RenouvNoEst can be used to create an object with S3 class "Renouv" from known parameters.

Examples

```
## Garonne data. Use a "Rendata" object as 'x'. Historical data are used!
fit <- Renouv(x = Garonne, distname = "weibull", trace = 1,
              main = "'Garonne' data")
summary(fit)

## generates a warning because of the ties
fit2 <- Renouv(x = Garonne, distname = "GPD",
               jitter.KS = FALSE,
               threshold = 2800, trace = 1,
               main = "'Garonne' data with threshold = 2800 and GPD fit")

## use a numeric vector as 'x'
fit3 <-
    Renouv(x = Garonne$OTdata$Flow,
           threshold = 2500,
           effDuration = 100,
           distname = "GPD",
           OTS.data = list(numeric(0), c(6800, 7200)),
           OTS.effDuration = c(100, 150),
           OTS.threshold = c(7000, 6000),
           trace = 1,
           main = "'Garonne' data with artificial \"OTS\" data")
## Add historical (fictive) data
fit4 <- Renouv(x = Garonne$OTdata$Flow,
               threshold = 2500,
               effDuration = 100,
               distname = "weibull",
               fixed.par.y = list(shape = 1.1),
               OTS.data = list(numeric(0), c(6800, 7200)),
               OTS.effDuration = c(100, 150),
               OTS.threshold = c(7000, 6000),
               trace = 0,
               main = "'Garonne' data with artificial \"OTS\" data")

##============================================================================
## use the 'venice' dataset in a r-largest fit from the 'evd' package
##============================================================================
## transform data: each row is a block
MAX.data <- as.list(as.data.frame(t(venice)))
## remove the NA imposed by the rectangular matrix format
MAX.data <- lapply(MAX.data, function(x) x[!is.na(x)])
MAX.effDuration <- rep(1, length(MAX.data))

## fit a Renouv model with no OT data. The threshold
## must be in the support of the gev distribution
u <- 66
```

```
fit.gpd <- Renouv(x = NULL,
                  MAX.data = MAX.data,
                  MAX.effDuration = MAX.effDuration,
                  distname.y = "GPD",
                  threshold = u,
                  numDeriv = FALSE,
                  trace = 0,
                  plot = FALSE)
## Not run:
  require(ismev)
  ## compare with results from the ismev package
  fit.gev <- rlarg.fit(venice)
  est.gev <- fit.gev$mle
  names(est.gev) <- c("loc", "scale", "shape")

  ## transform the 'gev' fit into a Ren parameter set.
  cov.gev <- fit.gev$cov
  rownames(cov.gev) <- colnames(cov.gev) <-  c("loc", "scale", "shape")
  trans <- gev2Ren(est.gev,
                   threshold = u,
                   vcovGev = cov.gev)
  est <- cbind(ismev = trans, RenextLab = coef(fit.gpd))
  colnames(est) <- c("ismev", "RenextLab")
  print(est)

  ## fill a 3d array with the two gpd covariance matrices
  cov2 <- attr(trans, "vcov")[c(1, 3, 4), c(1, 3, 4)]

  ## covariance
  covs <-
    array(dim = c(2, 3, 3),
          dimnames = list(c("ismev", "RenextLab"),
            colnames(fit.gpd$cov), colnames(fit.gpd$cov)))

  covs["ismev", , ] <- cov2
  covs["RenextLab", , ] <- fit.gpd$cov
  print(covs)

## End(Not run)
```

RenouvNoEst *Define a 'renouvellement' model without estimation*

Description

Build a 'renouvellement' model using parameters given by the user.

Usage

```
RenouvNoEst(threshold,
            estimate = NULL,
            distname.y = "exponential",
            fixed.par.y = NULL,
            trans.y = NULL,
            pct.conf = c(95, 70),
            rl.prob = NULL,
            prob.max = 1 - 1e-04,
            pred.period = NULL,
            cov = NULL,
            nb.OT = NULL,
            infer.method = NULL)
```

Arguments

`threshold`	The threshold.
`estimate`	Numeric named vector containing the estimates for the parameters. It must be compatible with the distribution chosen, and must contain in first position an element named `"lambda"` representing an estimated event rate in *events by year*.
`distname.y`	Character giving the name of the distribution.
`fixed.par.y`	Numeric named vector containing values for vectors which are considered as fixed (and not estimated).
`trans.y`	Transformation as in `Renouv`. Used only when `distname.y` is equal to `"exponential"`.
`pct.conf`	Vector of percents for confidence limits.
`rl.prob`	Probability used in the return level computations. These values are used for instance in return level plots produced with the `plot.Renouv` method. When `NULL` a default vector is used.
`prob.max`	Maximal probability for which computations are done.
`pred.period`	Vector of periods for which predicted return levels will be computed.
`cov`	Covariance matrix for the provided estimated parameters. Must have rownames and colnames in accordance with those of `estimate`. This covariance matrix is used to build confidence limits on parameters and on return levels using the *delta method*.
`nb.OT`	Number of data over the threshold used for estimation. This will be used only when `distname.y` is equal to `"exponential"`.
`infer.method`	Inference method. Will normally be the *delta method*.

Details

This function is used for plotting or comparing models with known parameter estimates but with no data available.

The parameters estimates should be accompanied with a covariance matrix assuming an approximately normal joint distribution of these. This matrix is usually obtained by computing the numerical derivatives of the log-likelihood at the second order at the estimates. This covariance is used to compute approximate confidence limits for the return levels of the unknown true distribution that was estimated.

Value

An object of class "Renouv" representing a 'renouvellement' model similar to those built with Renouv. This is mainly a list. Note however that some list elements found in Renouv objects built by Renouv can not be found here. For instance, the returned objects embeds no goodness-of-fit results since the object is created without making use of any data.

Author(s)

Yves Deville

See Also

Renouv to estimate such models.

Examples

```
##=====================================================================
## Example from S. Coles' book, page 86 'rainfall data'.
## Note that the first parameter is here the rate 'lambda', and no the
## probability of exceedance as in Coles' book.
##=====================================================================
estimate <- c(lambda = 152 / 48, scale = 7.44, shape = 0.184)
cov <- matrix(c(4.9e-7 * (17531 / 48)^2,  0.0000,  0.0000,
                0.0000,  0.9180, -0.0655,
                0.0000, -0.0655,  0.0102),
              nrow = 3)
colnames(cov) <- rownames(cov) <- names(estimate)
renNE <- RenouvNoEst(threshold = 30, distname.y = "gpd",
                     pct.conf = c(95, 70),
                     estimate = estimate,
                     nb.OT = 152, cov = cov)
summary(renNE)
plot(renNE, main = "Daily rainfall data SW England", ylim = c(0, 400))
```

RLlegend *Legend management for return level plots*

Description

Legend management for return level plots produced with the plot and lines method of the "Renouv" class.

Usage

```
RLlegend.ini(x = "topleft", bty = "n", ...)
RLlegend.show()
```

Arguments

`x`	A possible value for the `x` argument of `legend`. This will usually be a character giving the position e.g, `"topleft"` or `"bottomleft"`. See the `legend` function help.
`bty`	As in `legend`. The default value `"n"` differs from the default value of `legend`.
`...`	Other arguments to be kept in the list and passed later to `legend`. These arguments should be chosen among those of `legend` modifying the global legend appearance (e.g., bg) but not among those modifying the legend content (e.g. `col` `pt.bg`, `legend`, ...) since the content is here built semi-automatically.

Details

This function is to be used in conjunction with `plot.Renouv` and `lines.Renouv` methods. It allows the construction of a legend in a semi-automatic fashion, using the value of the `par` argument of the `plot` and `lines` methods to specify the legend construction.

Each call to the `plot.Renouv` or `lines.Renouv` changes the content of a list variable named `.RLlegend` in a special environment bound to the package. This list is re-created when `RLlegend.ini` is called, and is used later to draw a legend on the active device when `RLlegend.show` is called. Between these two calls, the `plot` and `lines` methods should be used with their arg `legend` set to `FALSE`.

Value

`RLlegend.ini` returns a copy of the variable which is set.

`RLlegend.show` returns nothing.

Note

The size of symbols (i.e, *plotting characters*) can be set by using the `RLpar` function and the `par` argument of the methods `plot.Renouv` and `lines.Renouv`. However it can not be changed in the legend.

Author(s)

Yves Deville

See Also

`plot.Renouv` and `lines.Renouv` for and the `RLpar` function to change the graphical parameters of the plot and the legend by using the `par` argument.

Examples

```
## use Garonne data
xG <- Garonne$OTdata$Flow
## use special "exponential" distribution
fit1 <- Renouv(x = xG, threshold = 2500, distname.y = "exponential",
               effDuration = 65, plot = FALSE)

## use 'exp' in black box fashion, hence with delta method
fit2 <- Renouv(x = xG, , threshold = 2500, distname.y = "exp",
               effDuration = 65, start.par.y = c(rate = 1), plot = FALSE)
RLlegend.ini() ## initialise legend
## sample points only
plot(fit1, main = "Two types of confidence lims",
     show = list(OT = TRUE, quant = FALSE, conf = FALSE),
     label = "",
     legend = FALSE)
## quant and confidence lims
lines(fit1,
     show = list(OT = FALSE, quant = TRUE, conf = TRUE),
     label = "exact",
     legend = FALSE)
## quant (overplot) and confidence lims
lines(fit2,
      show = list(OT = FALSE, quant = TRUE, conf = TRUE),
      par = RLpar(quant.lty = 2, quant.col = "SpringGreen2",
        conf.conf1.col = "orangered", conf.conf1.lwd = 3,
        conf.conf2.col = "orangered", conf.conf2.lwd = 3),
      label = "delta",
      legend = FALSE)
RLlegend.show() ## now draw legend
```

`RLpar` *Graphical parameters for Return Level plots*

Description

Build a hierarchical list of graphical parameters that can be used in the methods plot or lines for the class `"Renouv"`.

Usage

```
RLpar(mono = TRUE,
      trace = 0L,
      ...)
```

Arguments

`mono`	Logical. The default `TRUE` is for plots possibly using colors but that can be printed in grayscale. With the value `FALSE`, curves or symbols will appear distinctly on a color device but not necessarily when printed in grayscale.
`trace`	Integer level of verbosity. The default value `0` prints nothing.
`...`	Arguments with names corresponding to the hierarchical structure and the graphical parameter to be changed.

Details

The formals are in correspondence with the list hierarchy using a column `"."` as separator to define the tree. Thus a `quant.col` formal argument can be used to specify the color of the quantile (or return level) curve, while `conf.conf1.col` will be used for the first confidence limits (lower and upper).

Value

A list containing lists in a hierarchical fashion. At the root level, an element concerns a single curve (e.g. the return level curve), a single scatterplot (e.g. sample used in POT), a group of curves (e.g. the confidence limits) or a group of scatterplots (e.g. the collection of `MAX` historical blocks). For single elements (curve or scatterplot) the list contains graphical elements with values as they would be given in `plot` or `lines` calls. For group elements, each element is a list of such lists.

Note

A list of default parameter values is built first using the model suitable for the `mono` value. Then the values provided by the user overwrite the existing. Thus a curve can be coloured even if `mono = TRUE`, if a colour specification is given for the corresponding element.

When the same parameter name is used several times in `RLpar`, a warning is thrown.

Author(s)

Yves Deville

See Also

`plot.Renouv` and `lines.Renouv` with which `RLpar` is to be used.

Examples

```
## change color for quantile curve and type for confidence
## limits #1 (with largest confidence level).
newRLpar <- RLpar(quant.col = "red", conf.conf1.lty = "dashed")
newRLpar$quant

## show the names of all possible editable parameters
names(unlist(RLpar()))
```

`RLplot` *Return level plot*

Description

Return level plot for "Renouvellement" data.

Usage

```
RLplot(data,
       x = NULL,
       duration = 1,
       lambda,
       conf.pct = 95,
       mono = TRUE,
       mark.rl = 100,
       mark.labels = mark.rl,
       mark.col = NULL,
       main = NULL,
       ylim = NULL,
          ...)
```

Arguments

`data`	A data.frame object with a column named `quant`.
`x`	Optional vector of observed levels.
`duration`	The (effective) duration corresponding to `x` if this argument is used.
`lambda`	Rate, with unit inverse of that used for `duration`, e.g. in inverse years when `duration` is in years.
`conf.pct`	Vector (character or integer) giving confidence levels. See **Details** below.
`mono`	If `TRUE` colours are replaced by black.
`mark.rl`	Return levels to be marked on the plot.
`mark.labels`	Labels shown at positions in `mark.rl`.
`mark.col`	Colours for marked levels.
`main`	Main title for the return level plot (defaults to empty title).
`ylim`	Limits for the y axis (defaults to values computed from the data).
`...`	Further args to be passed to `plot`. Should be removed in future versions.

Details

Percents should match column names in the data.frame as follows. The upper and lower limits are expected to be `U.95` and `L.95` respectively. For a `70%` confidence percentage, columns should have names `"U.70"` and `"L.70"`.

The plot is comparable to the return level described in Coles'book and related packages, but the return level is here in log-scale while Coles uses a loglog-scale. A line corresponds here to a one parameter exponential distribution, while Coles'plot corresponds to Gumbel, however the two plots differ only for small return periods. This plot is identical to an `expplot` but with x and y scales changed: only axis tick-marks differ. The convexity of the scatter plot is therefore opposed in the two plots.

Note

Confidence limits correspond to *two-sided symmetrical intervals*. This means that the (random) confidence interval may be under or above the true unknown value with the same probabilities. E.g. the probability that the unknown quantile falls above `U.95` is `2.5%`. The two bounds are yet generally not symmetrical with respect to `quant`; such a behaviour follows from the use of "delta" method for approximate intervals.

It is possible to add graphical material (points, lines) to this plot using `log(returnlev)` and `quantile` coordinates. See **Examples** section.

Author(s)

Yves Deville

References

Coles S. (2001) *Introduction to Statistical Modelling of Extremes Values*, Springer.

See Also

See `expplot` for a classical exponential plot. See Also as `Renouv` to fit "Renouvellement" models. The `return.level` function in the `extRemes` package.

Examples

```
## Typical probability vector
prob <- c(0.0001,
  seq(from = 0.01, to = 0.09, by = 0.01),
  seq(from = 0.10, to = 0.80, by = 0.10),
  seq(from = 0.85, to = 0.99, by = 0.01),
  0.995, 0.996, 0.997, 0.998, 0.999, 0.9995)

## Model parameters rate = #evts by year, over nyear
lambda <- 4
nyear <- 30
theta.x <- 4

## draw points
n.x <- rpois(1, lambda = lambda*nyear)
x <- rexp(n.x, rate = 1/theta.x)

## ML estimation (exponential)
lambda.hat <- n.x / nyear
theta.x.hat <- mean(x)
```

```
## Compute bounds (here exact)
alpha <- 0.05

quant <- qexp(p = prob, rate = 1/theta.x.hat)

theta.L <- 2*n.x*theta.x.hat / qchisq(1 - alpha/2, df = 2*n.x)
theta.U <- 2*n.x*theta.x.hat / qchisq(alpha/2, df = 2*n.x)

L.95 <- qexp(p = prob, rate = 1/theta.L)
U.95 <- qexp(p = prob, rate = 1/theta.U)

## store in data.frame object
data <- data.frame(prob = prob, quant = quant, L.95 = L.95, U.95 = U.95)

RLplot(data = data, x = x, lambda = lambda.hat,
       duration = nyear,
       main = "Poisson-exponential return levels")

RLplot(data = data, x = x, lambda = lambda.hat, duration = nyear,
       mark.rl = 10, mark.labels = "10 ans", mono = FALSE, mark.col = "SeaGreen",
       main = "Poisson-exponential return levels")

points(x = log(50), y = 25, pch = 18, cex = 1.4, col = "purple")
text(x = log(50), y = 25, col ="purple", pos = 4, labels = "special event")
```

roundPred — *Round quantiles in a pseudo-prediction table*

Description

Round the quantiles of a pseudo prediction table such that computed by `predict.Renouv`.

Usage

```
roundPred(pred, dig.quant = NA)
```

Arguments

`pred`	The data.frame containing the predicted quantiles and return levels.
`dig.quant`	Number of digits. Guessed if not provided.

Details

Only the columns that can be considered as quantiles are rounded. These are assumed to have names `"quant"` for the expected return level and `"L."` or `"U."` followed by a percentage for lower and upper confidence limits (e.g. `"L.95"` and `"U.95"` for 95% percent confidence limits. The number of digits guessed is experimental.

Value

A data.frame with the same structure as that given, but with some columns rounded.

`rRendata` *Simulate a random RenData object*

Description

Simulate a random `RenData` object that can be used within the `Renouv` function for tests.

Usage

```
rRendata(lambda = 1,
         threshold = 0,
         effDuration = 100,
         distname.y = "exp",
         par.y = c(rate = 1),
         start = "1913-01-01",
         name = NULL,
         varName = "X", varUnit = "?",
         simDate = TRUE, roundDate = FALSE,
         MAX.effDuration = NULL,
         MAX.r = rep(1L, length(MAX.effDuration)),
         OTS.effDuration = NULL,
         OTS.threshold = NULL)
```

Arguments

`lambda`	The rate of the Homogeneous Poisson Process.
`threshold`	The threshold for the exceedances.
`effDuration`	The effective duration of the main Over Threshold (OT) period. This must be a positive value.
`distname.y`	Name of the distribution for the excesses to be simulated. See **Details**.
`par.y`	A named vector or list giving the parameters values for the distribution. The name must conform to the chosen distribution.
`start`	A `POSIXct` object, or character that can be coerced to `POSIXct` (e.g. a date given as a character in the `"YYYY-MM-DD"` format) giving the start of the main OT sample.
`name`	A name for the dataset which will be attached to it and be used by some methods for `"Rendata"`.
`varName`	Name of the simulated variable.
`varUnit`	Unit for the simulated variable (is used by plot).
`simDate`	Logical. If `TRUE` the dates will be reported for the historical data (MAX and OTS).

`roundDate`	Logical. If `TRUE` the time part ot the `date` column will be rounded. Not implemented yet.
`MAX.effDuration`	Vector of the durations for the `MAX` historical blocks.
`MAX.r`	Vector of the (positive) numbers of observations for `MAX` historical blocks. Must have the same length as `MAX.effDuration`. See **Caution** below for the effect of selection large values.
`OTS.effDuration`	Vector of durations for the `OTS` historical blocks.
`OTS.threshold`	Vector of numerical thresholds for the observations in `OTS` historical blocks. Must have the same length as `OTS.effDuration`. All values must be `>= threshold`.

Details

The distribution of the excesses named in `distname.y` can be any known distribution, provided that when prefixed with the usual `"r"` letter, the name gives the wanted simulation function. For example, with `distname.y = "exp"`, the `rexp` function is used and `par.y` must thus contain an element with name `"rate"`.

When a suitable numeric threshold is given, the simulated marks of the marked process are the sum of the threshold and of a random excess drawn from `distname.y`. When the threshold is not a finite numeric value, the observed marks are the simulated values themselves.

The main OT sample is assumed to begin at `start`. Historical MAX blocks (if any) are assumed to be just before `start`, and OTS are just before `start` or just before the beginning of the MAX blocks when there are some. The dates are computed without taking into consideration the problems of leap years or leap seconds.

Value

An object with S3 class `"Rendata"`. This class currently has `plot` and `summary` methods.

Caution

By construction, each `MAX` block contains at least one observation, while a random period of the same duration might have none. The simulated number of events on a MAX block is generated using a censored Poisson distribution. Care must be taken when estimations are made on such data, since creating `MAX` blocks obviously create a positive bias on `lambda`. Such bias then also affects the other parameters concerning the excesses, because these parameters are no longer orthogonal to the rate parameter `lambda` when historical data are used. The bias can be severe if `MAX` blocks with small durations are used, or if large number of events are chosen in `MAX.r`.

Note

When `effDuration` is small relative to the inverse of `lambda` the number of simulated marks in the OT sample may be 0 which can cause problems for some uses of the created data.

Author(s)

Yves Deville

See Also

`plot.Rendata`, `summary.Rendata`.

Examples

```
set.seed(1234)
rd <- rRendata(effDuration = 60,
               MAX.effDuration = rep(3, 6),
               MAX.r = rep(4, 6),
               distname.y = "exp", par.y = c(rate = 1/100))
plot(rd)
summary(rd)
rd2 <- rRendata(effDuration = 10,
                MAX.effDuration = rep(60, 2),
                MAX.r = rep(3, 2),
                simDate = FALSE,
                distname.y = "gpd", par.y = c(scale = 20, shape = 0.16))
plot(rd2)
rd3 <- rRendata(effDuration = 10,
                OTS.effDuration = rep(60, 2),
                OTS.threshold = rep(80, 2),
                simDate = FALSE,
                distname.y = "gpd", par.y = c(scale = 20, shape = 0.16))
plot(rd3)
## Renouv fit with historical data
fit <- Renouv(rd)
summary(fit)
```

`SandT` *Compute empirical survivals (S) and return periods (T)*

Description

Compute the empirical survival values and the empirical return periods at the observations of an object. These are used as plotting positions in several plots.

Usage

```
SandT(object, points = c("p", "H"), a = 0, naive = FALSE)
```

Arguments

`object`	The object containing the data, with class `"Renouv"` or `"Rendata"`.
`points`	Option for the computation of the plotting positions. When `points` is set to `"p"`, the p-points formula is used with the selected value of `a`. This formula is used to compute the survival from which the return period is computed. When instead `points` is set to `"H"`, *Nelson's formula* is used to compute the return periods, the survival value still being given by the p-points formula. When the

data is heterogeneous, i.e. when `object` contains `MAX` and/or `OTS` data, Nelson's formula is used only to compute the return periods of the upper slice of levels.

`a`	Parameter used in the interpolation formula for the inverse return periods as in Hirsch and Stedinger (1987).
`naive`	Logical. When `TRUE`, naive plotting positions are used to display `MAX` or `OTS` data. These can be defined only when a main sample exists in `object` as a `x.OT` element. For each `MAX` or `OTS` block, the positions use the number of events predicted using the rate of events as estimated from the main sample. When the main sample has a small durations, such predictions are likely to be misleading.

Details

When the object contains historical information (`MAX` or `OTS`), the computation is an adaptation Hirsch and Stedinger (1987) for the Marked Process (MP) context. The original method is devoted to block maxima and interpolates the survival values at the thresholds which are computed first. For MP, the interpolation is done for the inverse return periods, and the survival values are deduced from those of the inverse return periods.

Nelson's formula provides unbiased estimates for the values of the cumulative hazard $H(x)$ at the order statistics, and thus can be used to estimate the log-return periods as required on the return level plot.

Value

A list with the following elements

`x`	Numeric vector containing the ordered values from all the available sources in the object: main sample, historical periods either 'MAX' or 'OTS'.
`group`, `groupNames`	Integer and character vectors giving the source of the values in `x`, in the same order of the values. For instance, `group[10]` gives the group form which `x[10]` was extracted, and the name of this group is `groupNames[group[10]]`.
`S`, `T`	Numeric vectors of the same length as `x` and containing the corresponding estimation of the survival value and of the return period.
`thresh`, `lambda.thresh`, `S.thresh`, `T.thresh`	Vector of thresholds and the corresponding estimation for the event rate, survival and return period. All the estimations are *for the threshold values*. The value of `T.thresh[i]` for a threshold `thresh[i]` results from a simple computation: divide the sum of the durations for blocks with thresholds >= `thresh[i]` by the number of events for these blocks.

seealso

The `ppoints` and `Hpoints` functions.

Warning

When using `points = "H"` the estimated values of the survival returned in `S` and those for the return period `T` no longer verify `T=1/S/lambda`, where `lambda` is the estimated rate. In this case, the values in `T` should be used in the return level plot, while the values in `S` should be used in the PP-plot.

Author(s)

Yves Deville

References

The original method for block maxima is described in

- Hirsch R.M. and Stedinger J.R.(1887) Plotting Positions for Historical Floods and their precision. *Water Ressources Research*, vol. 23, N. 4 pp. 715-727.
- Millard S. and Neerchal N. (2001). *Environmental Statistics with S-Plus*. CRC Press

The adaptation for the Marked Process context is described in the *Renext Computing Details* document.

Examples

```
## use an object with class "Rendata"
ST1 <- SandT(object = Garonne)
## basic return level plot
plot(ST1$T, ST1$x, col = ST1$group, log = "x")
## use an object with class "Renouv"
fit <- Renouv(x = Garonne, plot = FALSE)
ST2 <- SandT(object = fit)
plot(ST2$T, ST2$x, col = ST2$group, log = "x")
```

---

skip2noskip *Fix non-skipped periods from skipped ones*

Description

Compute non-skipped periods form start and end of skipped periods.

Usage

```
   skip2noskip(skip = NULL,
               start = NULL,
               end = NULL)
```

Arguments

skip	A data.frame object with `start` and end columns that can be coerced to `POSIXct`. Other columns can be present (and will be ignored). Each row describes a missing period. Rows must be sorted in chronological order and periods should not overlap. Validity checks are at the time very limited.
start	Beginning of the whole period, to be used in `as.POSIXct`.
end	End of the whole period to be used in `as.POSIXct`.

Details

In a 'normal' use of this function `start` and `end` are given, and are respectively *before the beginning* of the first `skip` period and *after the end* of the last `skip` period. Thus the returned dataframe will have `nrow(skip)+1` rows. However, `start` and `end` can be `NULL` in which case only the `nrows(skip)-1` "inner" non-skipped periods will be returned. If `start` and `end` are `NULL` and `skip` has only one row, the returned result is `NULL`.

Value

A data.frame object with two `POSIXct` columns named `start` and `end`. Each row corresponds to a non-skipped period

Author(s)

Yves Deville

See Also

`readXML` for reading data from XML and csv files.

Examples

```
## Brest data embeds a description of the gaps

ns <- skip2noskip(skip = Brest$OTmissing)

ns2 <- skip2noskip(skip = Brest$OTmissing,
                   start = Brest$OTinfo$start,
                   end = Brest$OTinfo$end)

## check durations. dur2 should be equal to the effective
## duration (with an error of a fraction of day)
dur <- as.numeric(sum(ns$end-ns$start))/365.25
dur2 <- as.numeric(sum(ns2$end-ns2$start))/365.25
```

`SLTW` *Shifted Left Truncated Weibull (SLTW) distribution*

Description

Density function, distribution function, quantile function and random generation for the Shifted Left Truncated Weibull distribution.

Usage

```
dSLTW(x, delta = 1.0, shape = 1.0, scale = 1.0, log = FALSE)
pSLTW(q, delta = 1.0, shape = 1.0, scale = 1.0, lower.tail = FALSE)
qSLTW(p, delta = 1.0, shape = 1.0, scale = 1.0)
rSLTW(n, delta = 1.0, shape = 1.0, scale = 1.0)
```

Arguments

`x, q`	Vector of quantiles.
`p`	Vector of probabilities.
`n`	Number of observations.
`delta, shape, scale`	Shift, shape and scale parameters. Vectors of length > 1 are not accepted.
`log`	Logical; if `TRUE`, the log density is returned.
`lower.tail`	Logical; if `TRUE` (default), probabilities are $\Pr[X \leq x]$, otherwise, $\Pr[X > x]$.

Details

The SLTW distribution function with shape $\alpha > 0$, scale $\beta > 0$ and shift $\delta > 0$ has survival function

$$S(y) = \exp\left\{-\left[\left(\frac{y+\delta}{\beta}\right)^{\alpha} - \left(\frac{\delta}{\beta}\right)^{\alpha}\right]\right\} \qquad (y > 0)$$

This distribution is that of $Y := X - \delta$ conditional to $X > \delta$ where X follows a Weibull distribution with shape α and scale β.

The hazard and mean residual life (MRL) are monotonous functions with the same monotonicity as their Weibull equivalent (with the same shape and scale). The moments or even expectation do not have simple expression.

This distribution is sometimes called *power exponential*. It is occasionally used in POT with the shift `delta` taken as the threshold as it should be when the distribution for the level X (and not for the exceedance Y) is known to be the standard Weibull distribution.

Value

`dSLTW` gives the density function, `pSLTW` gives the distribution function, `qSLTW` gives the quantile function, and `rSLTW` generates random deviates.

See Also

`Lomax` for the Lomax distribution which is a limit case of `SLTW`.

Examples

```
shape <- rexp(1)+1
delta = 10

xl <- qSLTW(c(0.001, 0.99), delta = delta, shape = shape)
x <- seq(from = xl[1], to = xl[2], length.out = 200)
```

```
f <- dSLTW(x, delta = delta, shape = shape)
plot(x, f, type = "l", main = "SLTW density")

F <- pSLTW(x, delta = delta, shape = shape)
plot(x, F, type = "l", main = "SLTW distribution")

X <- rSLTW(5000, delta = delta, shape = shape)
## Should be close to the uniform repartition
plot(ecdf(pSLTW(X, delta = delta, shape = shape)))
```

---

spacings *Methods computing spacings between Largest Order Statistics*

Description

Methods computing the random *spacings* for the Largest Order Statistics of the marks in a POT renewal.

Usage

```
spacings(object, ...)

## S3 method for class 'numeric'
spacings(object, wExp = TRUE, ...)

## S3 method for class 'data.frame'
spacings(object, varName, wExp = TRUE, ...)

## S3 method for class 'Rendata'
spacings(object, type = c("MAX", "OTS", "OT"), wExp = TRUE, ...)
```

Arguments

`object`	A object containing the marks X_i. This is can be a vector or a data frame for data aggregated by blocks. For an object of class `data.frame` one of the three data.frame slots: `OTdata`, `MAXdata` or `OTSdata` can be used using the suitable value of `type`.
`varName`	Character vector of length 1 giving the name of the variable when `object` is a data.frame.
`wExp`	Logical. If `TRUE`, the spacings are weighted as explained in **Details**.
`type`	Character specifying the data.frame to be used when object has class `"Rendata"`.
`...`	Not used yet.

Details

The spacings are the differences between the Largest Order Statistics. They are useful for some estimation tasks or diagnostics. Given random variables X_i, the i-th spacing Y_i is the difference $X_{(i)} - X_{(i+1)}$ between the i-th largest order statistic $X_{(i)}$ and the next in the decreasing order i.e. $X_{(i+1)}$.

When the r.vs X_i form a sample of an exponential or Gumbel distribution, the spacings associated with the largest order statistics are or tend to be independent and exponentially distributed. More precisely, the weighted spacings $i \times Y_i$ have or tend to have the same exponential distribution. This can be used to estimate the shape parameter of the underlying distribution using only the largest order statistics. Moreover the $r-1$ spacings Y_i built from the r largest order statistics $i \leq r$ are or tend to be independent from the r-th largest order statistic $X_{(r)}$.

When `wExp` is `TRUE`, the returned values are the weighted spacings $i \times Y_i$.

Value

A list or vector containing the spacings. When the data is structured in blocks as is the `MAXdata` slot of an object of class `"Rendata"`, the spacings are computed form the order statistics *within each block*, to maintain independence with the next order statistic in decreasing order.

Caution

By default, the spacings are scaled as explained above, thus assuming that the marks are exponentially distributed.

Author(s)

Yves Deville

References

Embrechts P., Klüppelberg C. and Mikosch T. (1997) *Modelling Extremal Events for Insurance and Finance*. Springer.

Examples

```
sp <- spacings(rgumbel(200, loc = 0, scale = 1))
expplot(sp)
sp1 <- spacings(rgev(200, loc = 0, scale = 1, shape = 0.3))
expplot(sp1)
## spacings are computed by block
sp2 <- spacings(object = Garonne$MAXdata,
                varName = Garonne$info$varName)
expplot(unlist(sp2))
sp3 <- spacings(object = Garonne, type = "OT")
expplot(sp3)
```

`summary.Rendata` *Summary and print methods for "Rendata" objects*

Description

Summary method for "Rendata" objects representing data to be used in renouvellement models.

Usage

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

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

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

Arguments

`object`	An object with class `"Rendata"`.
`x`	An object of class `"summary.Rendata"`, i.e. a result of a call to `summary.Rendata`.
`...`	Further arguments passed to or from other methods.

Examples

```
## Brest example: no historical data
summary(Brest)

## Garonne example:  with historical data
summary(Garonne)
```

`summary.Renouv` *Summary and print methods for "Renouv" objects*

Description

Summary method for "Renouv" objects representing 'Renouvellement' (POT) fitted models.

Usage

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

   ## S3 method for class 'Renouv'
summary(object,
        correlation = FALSE,
        symbolic.cor = FALSE,
        ...)

   ## S3 method for class 'summary.Renouv'
print(x,
      coef = TRUE,
      pred = TRUE,
      probT = FALSE,
      digits = max(3, getOption("digits") - 3),
      symbolic.cor = x$symbolic.cor,
      signif.stars = getOption("show.signif.stars"),
      ...)

   ## S3 method for class 'summary.Renouv'
format(x,
      ...)
```

Arguments

`object`	An object with class `"Renouv"`.
`x`	An object of class `"summary.Renouv"`, i.e. a result of a call to `summary.Renouv`.
`correlation`	Logical; if `TRUE`, the correlation matrix of the estimated parameters is returned and printed.
`coef`	Logical. If `FALSE`, the table of coefficients and t-ratios' will not be printed.
`pred`	Logical. If `FALSE`, the table of return periods/levels will not be printed.
`probT`	If `FALSE`, the p-values for the t-tests will not be printed nor displayed.
`digits`	the number of significant digits to use when printing.
`symbolic.cor`	logical. If `TRUE`, print the correlations in a symbolic form (see `symnum`) rather than as numbers.
`signif.stars`	logical. If `TRUE`, 'significance stars' are printed for each coefficient.
`...`	Further arguments passed to or from other methods.

Details

`print.summary.Renouv` tries to be smart about formatting the coefficients, standard errors, return levels, etc. `format.summary.Renouv` returns as a limited content as a character string. It does not embed coefficients values nor predictions.

Value

The function `summary.RenOUV` computes and returns a list of summary statistics concerning the object of class `"Rendata"` given in `object`. The returned list is an object with class `"summary.Renouv"`.

The function `print.summary.Rendata` does not returns anything.

See Also

The model fitting function `Renouv` (to build `"Renouv"` model objects), `summary`.

Examples

```
## use Brest data
fit <- Renouv(Brest)
summary(fit)
```

`translude` *Make translucient colors*

Description

Make translucient colors.

Usage

```
translude(colors, alpha = 0.6)
```

Arguments

`colors`	A vector of colors in a format that can be understood by `col2rgb`.
`alpha`	Vector of level(s) of opacity between `0` and `1` (`0` means fully transparent and `1` means opaque). After recycling to reach the required length, this value or vector is used as `alpha` in `rgb`.

Value

A vector of translucient (or semi-transparent) colors.

Note

Using the **RColorBrewer** package might be a better option that transluding chosen colors!

vcov.Renouv *Variance-covariance matrix of the estimates of a "Renouv" object*

Description

Variance-covariance matrix of the estimates of a "Renouv" object.

Usage

```
## S3 method for class 'Renouv'
vcov(object, ...)
```

Arguments

`object`	Object of class `"Renouv"`.
`...`	Not used at the time.

Value

A variance-covariance matrix. The rows an columns correspond to the parameters of the Renouv object. The are the rate `"lambda"` for the Poisson process, and the parameters of the distribution for the excesses over the threshold, with names depending on the chosen distribution.

Author(s)

Yves Deville

See Also

The `vcov` generic.

weibplot *Classical Weibull distribution plot*

Description

Plots a vector using Weibull distribution scales

Usage

```
   weibplot(x,
            plot.pos = "exp",
            shape = NULL, scale = NULL,
            labels = NULL,
            mono = TRUE,
            ...)
```

Arguments

`x`	The vector to be plotted.
`plot.pos`	plotting position for points: either "exp" for *expected* ranks or "med" for a *median* rank approximation (see **Details** below).
`shape`	Shape parameter for one or several Weibull lines to be plotted.
`scale`	Scale parameter for one or several Weibull lines to be plotted.
`labels`	Text to display in legend when Weibull lines are specified.
`mono`	Monochrome graph.
`...`	Arguments to be passed to `plot`.

Details

This plot shows $\log\{-\log[1-F(x)]\}$ against $\log(x)$ where $F(x)$ at point i is taken as $i/(n+1)$ if `plot.pos` is `"exp"`, or as the "median rank" approximation $(i-0.3)/(n+0.4)$ if `plot.pos` is `"med"`.

Note

The graph displayed uses a log scale for x. The log-log scale for y is emulated via the construction of suitable graduations. So be careful when adding graphical material (points, etc) to this graph with functions of the "add to plot" family (`points`, `lines`, ...).

Author(s)

Yves Deville

See Also

The `expplot` function for an "exponential distribution" plot (dedicated to the `shape = 1` case), and the `fweibull` function for ML estimation of the parameters.

Examples

```
x <- rweibull(200, shape = 1.2, scale = 1)
weibplot(x, main = "Classical Weibull plot")
## Weibull lines
weibplot(x, shape = c(0.9, 1.3), scale = 1)
weibplot(x, shape = c(0.9, 1.3), scale = 1,
         labels = c("before", "after"))
weibplot(x, shape = c(0.9, 1.3), scale = 1,
         labels = c("before", "after"),
         mono = TRUE)
```

Index