Package 'lite'

October 13, 2022

Title Likelihood-Based Inference for Time Series Extremes

Version 1.0.0

Date 2022-04-06

Description Performs likelihood-based inference for stationary time series
extremes. The general approach follows Fawcett and Walshaw (2012)
<doi:10.1002/env.2133>. Marginal extreme value inferences are adjusted for
cluster dependence in the data using the methodology in Chandler and Bate
(2007) <doi:10.1093/biomet/asm015>, producing an adjusted log-likelihood
for the model parameters. A log-likelihood for the extremal index is
produced using the K-gaps model of Suveges and Davison (2010)
<doi:10.1214/09-AOAS292>. These log-likelihoods are combined to make
inferences about return levels.

Imports chandwich, exdex, graphics, revdbayes, sandwich, stats

License GPL (>= 2)

Encoding UTF-8

Depends R (>= 3.3.0)

RoxygenNote 7.1.2

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

URL <https://paulnorthrop.github.io/lite/>,
<https://github.com/paulnorthrop/lite>

BugReports <https://github.com/paulnorthrop/lite/issues>

Config/testthat/edition 3

NeedsCompilation no

Author Paul J. Northrop [aut, cre, cph]

Maintainer Paul J. Northrop <p.northrop@ucl.ac.uk>

Repository CRAN

Date/Publication 2022-04-08 08:32:29 UTC

R topics documented:

Bernoulli   *Frequentist inference for the Bernoulli distribution*

Description

Functions involved in making inferences about the probability of success in a Bernoulli distribution using maximum likelihood estimation.

Usage

```
fitBernoulli(data)

## S3 method for class 'Bernoulli'
coef(object, ...)

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

## S3 method for class 'Bernoulli'
nobs(object, ...)

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

Arguments

data   A numeric vector of outcomes from Bernoulli trials: 0 for a failure, 1 for a
            success. Alternatively, a logical vector with FALSE for a failure and TRUE for
            a success. Missing values are removed using `na.omit`.

object   A fitted model object returned from `fitBernoulli()`.

...   Further arguments. None are used currently.

Details

fitBernoulli: fit a Bernoulli distribution using maximum likelihood estimation, using an **independence** log-likelihood formed by summing contributions from individual observations. No adjustment for cluster dependence has been made in estimating the variance-covariance matrix stored as component in vcov in the returned object.

coef, vcov, nobs and logLik methods are provided.

Value

fitBernoulli returns an object of class "Bernoulli", a list with components: maxLogLik, mle, nobs, vcov, n0, n1, data, obs_data, where data are the input data and, obs_data are the input data after any missing values have been removed, using [na.omit](#) and n0 and n1 are, respectively, the number of failures and the number of successes.

coef.Bernoulli: a numeric vector of length 1 with name prob. The MLE of the probability of success.

vcov.Bernoulli: a 1×1 matrix with row and column name prob. The estimated variance of the estimator of the probability of success. No adjustment for cluster dependence has been made.

nobs.Bernoulli: a numeric vector of length 1 with name prob. The number of observations used to estimate the probability of success.

logLik.Bernoulli: an object of class "logLik": a numeric scalar with value equal to the maximised log-likelihood. The returned object also has attributes nobs, the numbers of observations used in this model fit, and "df" (degrees of freedom), which is equal to the number of total number of parameters estimated (1).

Examples

```
# Set up data
cdata <- c(exdex::cheeseboro)
u <- 45
exc <- cdata > u

# Fit a Bernoulli distribution
fit <- fitBernoulli(exc)

# Calculate the log-likelihood at the MLE
res <- logLikVector(fit)

# The logLik method sums the individual log-likelihood contributions.
logLik(res)

# nobs, coef, vcov, logLik methods for objects returned from fitBernoulli()
nobs(fit)
coef(fit)
vcov(fit)
logLik(fit)
```

---

estfun   *Functions for the* estfun *method*

Description

Functions to calculate contributions to the score vector from individual observations for a fitted model object.

Usage

```
## S3 method for class 'Bernoulli'
estfun(x, ...)

## S3 method for class 'GP'
estfun(x, eps = 1e-05, m = 3, ...)
```

Arguments

x	A fitted model object.
...	Further arguments. None are used for estfun.Bernoulli or estfun.GP.
eps, m	These control the estimation of the observed information in gpObsInfo when the GP shape parameter ξ is very close to zero. In these cases, direct calculation is unreliable. eps is a (small, positive) numeric scalar. If the absolute value of the input value of ξ, that is, pars[2], is smaller than eps then we approximate the [2, 2] element using a Taylor series expansion in ξ, evaluated up to and including the mth term.

Details

An [estfun](#) method is used by [meatCL](#) to calculate the [meat](#) in the sandwich covariance estimator on which the log-likelihood adjustments in [flite](#) are based. Specifically, [meatCL](#) is used to calculate the argument V passed to [adjust_loglik](#).

Value

An $n \times k$ matrix containing contributions to the score function from n observations for each of the k parameters.

estfun.Bernoulli: an $n \times 2$ matrix, where n is the sample size, the length of the input data to [fitBernoulli](#). The column is named prob.

estfun.GP: an $n \times 2$ matrix, where n is the sample size, the length of the input data to [fitGP](#). The columns are named sigma[u] and xi

See Also

[Bernoulli](#) for maximum likelihood inference for the Bernoulli distribution.

[generalisedPareto](#) for maximum likelihood inference for the generalised Pareto distribution.

Examples

```
library(sandwich)

# estfun.Bernoulli
bfit <- fitBernoulli(c(exdex::cheeseboro) > 45)
head(estfun(bfit))

# estfun.generalisedPareto
gpfit <- fitGP(c(exdex::cheeseboro), u = 45)
head(estfun(gpfit))
```

---

flite   *Frequentist threshold-based inference for time series extremes*

Description

Performs threshold-based frequentist inference for 3 aspects of stationary time series extremes: the probability that the threshold is exceeded, the marginal distribution of threshold excesses and the extent of clustering of extremes, as summarised by the extremal index.

Usage

```
flite(data, u, cluster, k = 1, inc_cens = TRUE, ny, ...)
```

Arguments

data	A numeric vector or numeric matrix of raw data. If data is a matrix then the log-likelihood is constructed as the sum of (independent) contributions from different columns. A common situation is where each column relates to a different year.
	If data contains missing values then [split_by_NAs](#) is used to divide the data further into sequences of non-missing values, stored in different columns in a matrix. Again, the log-likelihood is constructed as a sum of contributions from different columns.
u	A numeric scalar. The extreme value threshold applied to the data. See **Details** for information about choosing u.
cluster	This argument is used to set the argument cluster to [meatCL](#), which calculates the matrix V passed as the argument V to [adjust_loglik](#). If data is a matrix and cluster is missing then cluster is set so that data in different columns are in different clusters. If data is a vector and cluster is missing then cluster is set so that each observation forms its own cluster.
	If cluster is supplied then it must have the same structure as data: if data is a matrix then cluster must be a matrix with the same dimensions as data and if data is a vector then cluster must be a vector of the same length as data. Each entry in cluster sets the cluster of the corresponding component of data.

k, inc_cens	Arguments passed to [kgaps](). k sets the value of the run parameter K in the K-gaps model for the extremal index. inc_cens determines whether contributions from right-censored inter-exceedance times are used. See **Details** for information about choosing k.
ny	A numeric scalar. The (mean) number of observations per year. Setting this appropriately is important when making inferences about return levels, using [returnLevel](), but ny is not used by flite so it need not be supplied now. If ny is supplied to flite then it is stored for use by [returnLevel](). Alternatively, ny can be supplied in a later call to [returnLevel](). If ny is supplied to both flite and [returnLevel]() then the value supplied to [returnLevel]() will take precedence.
...	Further arguments to be passed to the function [meatCL]() in the sandwich package. In particular, the clustering adjustment argument cadjust may make a difference if the number of clusters is not large.

Details

There are 3 independent parts to the inference, all performed using maximum likelihood estimation.

1. A Bernoulli(p_u) model for whether a given observation exceeds the threshold u.
2. A generalised Pareto, GP(σ_u, ξ), model for the marginal distribution of threshold excesses.
3. The K-gaps model for the extremal index θ.

The general approach follows Fawcett and Walshaw (2012).

For parts 1 and 2, inferences based on a mis-specified independence log-likelihood are adjusted to account for clustering in the data. Here, we follow Chandler and Bate (2007) to estimate adjusted log-likelihood functions for p_u and for (σ_u, ξ), with the argument cluster defining the clusters. This aspect of the calculations is performed using the [adjust_loglik]() in the [chandwich]() package (Northrop and Chandler, 2021). The GP distribution initial fit of the GP distribution to threshold excesses is performed using the [grimshaw_gp_mle]() function in the [revdbayes]() package (Northrop, 2020).

In part 3, the methodology described in Suveges and Davison (2010) is implemented using the [exdex]() package (Northrop and Christodoulides, 2022).

Two tuning parameters need to be chosen: a threshold u and the K-gaps run parameter K. The [exdex]() package has a function [choose_uk]() to inform this choice.

Each of part of the inference produces a log-likelihood function (adjusted for parts 1 and 2). These log-likelihoods are combined (summed) to form a log-likelihood function for the parameter vector $(p_u, \sigma_u, \xi, \theta)$. Return levels are a function of these parameters and therefore inferences for return levels can be based on this log-likelihood.

Value

An object of class c("flite", "lite", "chandwich"). This object is a function with 2 arguments:

- pars, a numeric vector of length 4 to supply the value of the parameter vector $(p_u, \sigma_u, \xi, \theta)$,
- type, a character scalar specifying the type of adjustment made to the independence log-likelihood in parts 1 and 2, one of "vertical", "none", "cholesky", or "spectral". For details see Chandler and Bate (2007). The default is "vertical" for the reason given in the description of the argument adj_type in [plot.flite]().

The object also has the attributes `"Bernoulli"`, `"gp"`, `"kgaps"`, which provide the fitted model objects returned from `adjust_loglik` (for `"Bernoulli"` and `"gp"`) and `kgaps` (for `"kgaps"`).

The named input arguments are returned in a list as the attribute `inputs`. If `ny` was not supplied then its value is `NA`.

Objects inheriting from class `"flite"` have `coef`, `logLik`, `nobs`, `plot`, `summary` and `vcov` methods. See `fliteMethods`.

References

Chandler, R. E. and Bate, S. (2007). Inference for clustered. data using the independence loglikelihood. *Biometrika*, **94**(1), 167-183. doi: 10.1093/biomet/asm015

Fawcett, L. and Walshaw, D. (2012), Estimating return levels from serially dependent extremes. *Environmetrics*, **23**, 272-283. doi: 10.1002/env.2133

Northrop, P. J. and Chandler, R. E. (2021). chandwich: Chandler-Bate Sandwich Loglikelihood Adjustment. R package version 1.1.5. https://CRAN.R-project.org/package=chandwich.

Northrop, P. J. and Christodoulides, C. (2022). exdex: Estimation of the Extremal Index. R package version 1.1.1. https://CRAN.R-project.org/package=exdex/.

Northrop, P. J. (2020). revdbayes: Ratio-of-Uniforms Sampling for Bayesian Extreme Value Analysis. R package version 1.3.9. https://paulnorthrop.github.io/revdbayes/

Suveges, M. and Davison, A. C. (2010) Model misspecification in peaks over threshold analysis, *Annals of Applied Statistics*, **4**(1), 203-221. doi: 10.1214/09AOAS292

See Also

`fliteMethods`, including plotting (adjusted) log-likelihoods for $(p_u, \sigma_u, \xi, \theta)$.

`Bernoulli` for maximum likelihood inference for the Bernoulli distribution.

`generalisedPareto` for maximum likelihood inference for the generalised Pareto distribution.

`kgaps` for maximum likelihood inference from the K-gaps model for the extremal index.

`choose_uk` to inform the choice of the threshold u and run parameter K.

Examples

```
### Cheeseboro wind gusts

# Make inferences
cdata <- exdex::cheeseboro
# Each column of the matrix cdata corresponds to data from a different year
# flite() sets cluster automatically to correspond to column (year)
cfit <- flite(cdata, u = 45, k = 3)
summary(cfit)

# 2 ways to find the maximised log-likelihood value
cfit(coef(cfit))
logLik(cfit)

# Plots of (adjusted) log-likelihoods
plot(cfit)
plot(cfit, which = "gp")
```

---

fliteMethods   *Methods for objects of class* `"flite"`

Description

Methods for objects of class `"flite"` returned from [flite](#).

Usage

```
## S3 method for class 'flite'
plot(
  x,
  which = c("all", "pu", "gp", "xi", "theta"),
  adj_type = c("vertical", "none", "cholesky", "spectral"),
  ...
)

## S3 method for class 'flite'
coef(object, ...)

## S3 method for class 'flite'
vcov(object, adjust = TRUE, ...)

## S3 method for class 'flite'
nobs(object, ...)

## S3 method for class 'flite'
logLik(object, ...)

## S3 method for class 'flite'
summary(object, adjust = TRUE, digits = max(3, getOption("digits") - 3L), ...)

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

Arguments

x	An object inheriting from class `"flite"`, a result of a call to [flite](#).
which	A character scalar indicating which plot(s) to produce. If which = `"all"` then all 4 plots described in **Details** are produced. Otherwise, only one of these plots is produced, with the possible names of the arguments being in the order that the plots are described in **Details**.
adj_type	A character scalar passed to [conf_intervals](#) and [conf_region](#) as the argument type to select the type of adjustment applied to the independence log-likelihood. Of the 3 adjustments, `"vertical"` is preferred because it preserves

constraints on the parameters, whereas the `"cholesky"` and `"spectral"` adjustment do not. In the generalised Pareto case the constraint that $\xi > -\sigma_u/x_{(n)}$, where $x_{(n)}$ is the largest excesses of the threshold u, is preserved.

`...`	For `plot.flite`: arguments passed to `plot`, such as graphical parameters.
	For `print.summary.flite`: additional arguments passed to `print.default`.
`object`	An object inheriting from class `"flite"`, a result of a call to `flite`.
`adjust`	A logical scalar. If `adjust = TRUE` then the elements of the variance-covariance matrix corresponding to (p_u, σ_u, ξ) are estimated using a sandwich estimator. See `flite`. Otherwise, this matrix is the inverse of the observed information matrix.
`digits`	An integer. Passed to `signif` to round the values in the summary.

Details

For `plot.flite`, if `which = "all"` then 4 plots are produced.

- Top left: (adjusted) log-likelihood for the threshold exceedence probability p_u, with a horizontal line indicating a 95% confidence interval for p_u.
- Top right: contour plot of the (adjusted) log-likelihood for the GP parameters (σ_u, ξ), showing (25, 50, 75, 90, 95)% confidence regions. The linear constraint $\xi > -\sigma_u/x_{(n)}$ is drawn on the plot.
- Bottom left: (adjusted) log-likelihood for ξ, with a horizontal line indicating a 95% confidence interval for ξ.
- Bottom right: log-likelihood for the extremal index θ, with a horizontal line indicating a 95% confidence interval for θ.

Value

`plot.flite`: No return value, only the plot is produced.

`coef.flite`: a numeric vector of length 4 with names `c("p[u]", "sigma[u]", "xi", "theta")`. The MLEs of the parameters p_u, σ_u, ξ and θ.

`vcov.flite`: a 4×4 matrix with row and column names `c("p[u]", "sigma[u]", "xi", "theta")`. The estimated variance-covariance matrix for the model parameters. If `adjust = TRUE` then the elements corresponding to p_u, σ_u and ξ are adjusted for cluster dependence using a sandwich estimator; otherwise they are not adjusted.

`nobs.flite`: a numeric vector of length 3 with names `c("p[u]", "gp", "theta")`. The respective number of observations used to estimate p_u, (σ_u, ξ) and θ.

`logLik.flite`: an object of class `"logLik"`: a numeric scalar with value equal to the maximised log-likelihood. This is the sum of contributions from three fitted models, from a Bernoulli model for occurrences of threshold exceedances, a generalised Pareto model for threshold excesses and a K-gaps model for the extremal index. The returned object also has attributes `nobs`, the numbers of observations used in each of these model fits, and `"df"` (degrees of freedom), which is equal to the number of total number of parameters estimated (4).

`summary.flite`: an object containing the original function call and a matrix of estimates and estimated standard errors with row names `c("p[u]", "sigma[u]", "xi", "theta")`. The object is printed by `print.summary.flite`.

`print.summary.flite`: the argument x is returned, invisibly.

See Also

[flite](#) to perform frequentist threshold-based inference for time series extremes.

---

generalisedPareto   *Frequentist inference for the generalised Pareto distribution*

Description

Functions involved in making inferences about the scale and shape parameters of a generalised Pareto distribution using maximum likelihood estimation.

Usage

```
fitGP(data, u)

## S3 method for class 'GP'
coef(object, ...)

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

## S3 method for class 'GP'
nobs(object, ...)

## S3 method for class 'GP'
logLik(object, ...)

gpObsInfo(pars, excesses, eps = 1e-05, m = 3)
```

Arguments

data	A numeric vector of raw data. Missing values are removed using [na.omit](#).
u	A numeric scalar. The extremal value threshold.
object	A fitted model object returned from fitGP().
...	Further arguments to be passed to the functions in the sandwich package [meat](#) (if cluster = NULL), or [meatCL](#) (if cluster is not NULL).
pars	A numeric parameter vector of length 2 containing the values of the generalised Pareto scale and shape parameters.
excesses	A numeric vector of threshold excesses, that is, amounts by which exceedances of u exceed u.
eps, m	These control the estimation of the observed information in gpObsInfo when the GP shape parameter ξ is very close to zero. In these cases, direct calculation is unreliable. eps is a (small, positive) numeric scalar. If the absolute value of the input value of ξ, that is, pars[2], is smaller than eps then we approximate the [2, 2] element using a Taylor series expansion in ξ, evaluated up to and including the mth term.

Details

`fitGP`: fit a generalised Pareto distribution using maximum likelihood estimation, using an **independence** log-likelihood formed by summing contributions from individual observations. No adjustment for cluster dependence has been made in estimating the variance-covariance matrix stored as component in vcov in the returned object. This function calls [grimshaw_gp_mle](#).

`coef`, `vcov`, `nobs` and `logLik` methods are provided for objects of class "GP" returned from `fitGP`.

`gpObsInfo`: calculates the observed information matrix for a random sample `excesses` from the generalized Pareto distribution, that is, the negated Hessian matrix of the generalized Pareto independence log-likelihood, evaluated at `pars`.

Value

`fitGP` returns an object of class "GP", a list with components: `maxLogLik`, `threshold`, `mle`, `vcov`, `exceedances`, `nexc`, where `exceedances` is a vector containing the values that exceed the threshold `threshold` and `nexc` is the length of this vector.

`coef.GP`: a numeric vector of length 2 with names `c("sigma[u]", "xi")`. The MLEs of the GP parameters σ_u and ξ.

`vcov.GP`: a 2×2 matrix with row and column names `c("sigma[u]", "xi")`. The estimated variance-covariance matrix for the model parameters. No adjustment for cluster dependence has been made.

`nobs.GP`: a numeric vector of length 1. The number of observations used to estimate (σ_u, ξ).

`logLik.GP`: an object of class "logLik": a numeric scalar with value equal to the maximised log-likelihood. The returned object also has attributes `nobs`, the numbers of observations used in each of these model fits, and "df" (degrees of freedom), which is equal to the number of total number of parameters estimated (2).

`gpObsInfo` returns a 2 by 2 matrix with row and columns names `c("sigma[u]", "xi")`.

Examples

```
# Set up data and set a threshold
cdata <- c(exdex::cheeseboro)

# Fit a generalised Pareto distribution
fit <- fitGP(cdata, 45)

# Calculate the log-likelihood at the MLE
res <- logLikVector(fit)

# The logLik method sums the individual log-likelihood contributions.
logLik(res)

# nobs, coef, vcov, logLik methods for objects returned from fitGP()
nobs(fit)
coef(fit)
vcov(fit)
logLik(fit)
```

| lite | *lite: Likelihood-Based Inference for Time Series Extremes* |

Description

Performs likelihood-Based inference for stationary time series extremes. The general approach follows Fawcett and Walshaw (2012). Marginal extreme value inferences are adjusted for cluster dependence in the data using the methodology in Chandler and Bate (2007), producing an adjusted log-likelihood for the model parameters. A log-likelihood for the extremal index is produced using the K-gaps model of Suveges and Davison (2010). These log-likelihoods are combined to make inferences about return levels.

Details

The main functions are

- `flite`: makes frequentist threshold-based inference for time series extremes to produce an adjusted log-likelihood for the model parameters.

- `returnLevel`: performs inference for return levels using the adjusted log-likelihood.

The main function is `flite`, which performs frequentist inference for time series extremes.

See `vignette("introduction-to-lite", package = "lite")` for an overview of the package.

References

Chandler, R. E. and Bate, S. (2007). Inference for clustered. data using the independence loglikelihood. *Biometrika*, **94**(1), 167-183. doi: 10.1093/biomet/asm015

Fawcett, L. and Walshaw, D. (2012), Estimating return levels from serially dependent extremes. *Environmetrics*, **23**, 272-283. doi: 10.1002/env.2133

Suveges, M. and Davison, A. C. (2010) Model misspecification in peaks over threshold analysis, *Annals of Applied Statistics*, **4**(1), 203-221. doi: 10.1214/09AOAS292

See Also

`flite` for frequentist threshold-based inference for time series extremes.

`returnLevel` for frequentist threshold-based inference for return levels.

---

logLikVector   *Functions for log-likelihood contributions*

Description

Generic function for calculating log-likelihood contributions from individual observations for a fitted model object.

Usage

```
logLikVector(object, ...)

## S3 method for class 'Bernoulli'
logLikVector(object, pars = NULL, ...)

## S3 method for class 'GP'
logLikVector(object, pars = NULL, ...)

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

Arguments

object	A fitted model object.
...	Further arguments. None are used for either `logLikVector.Bernoulli` or `logLikVector.GP`.
pars	A numeric parameter vector.
	For `logLikVector.Bernoulli` this is a vector of length 1 containing a value of the Bernoulli success probability.
	For `logLikVector.GP` this is a numeric vector of length 2 containing the values of the generalised Pareto scale (σ_u) and shape (ξ) parameters.

Details

A `logLikVector` method is used to construct a log-likelihood function to supply as the argument `loglik` to the function [adjust_loglik](#), which performs log-likelihood adjustment for parts 1 and 2 of the inferences performed by [flite](#).

The `logLik` method `logLik.LogLikVector` sums the log-likelihood contributions from individual observations.

Value

For `logLikVector`: an object of class `logLikVec`. This is a numeric vector of length n containing contributions to the the independence log-likelihood from n observations, with attributes "df" (degrees of freedom), giving the number of estimated parameters in the model, and "nobs", giving the number observations used to perform the estimation.

For `logLik.logLikVector`: an object of class `logLik`. This is a number with the attributes "df" and "nobs" as described above.

See Also

[Bernoulli](#) for maximum likelihood inference for the Bernoulli distribution.

[generalisedPareto](#) for maximum likelihood inference for the generalised Pareto distribution.

Examples

```
# logLikVector.Bernoulli
bfit <- fitBernoulli(c(exdex::cheeseboro) > 45)
bvec <- logLikVector(bfit)
head(bvec)
logLik(bvec)
logLik(bfit)

# estfun.generalisedPareto
gpfit <- fitGP(c(exdex::cheeseboro), u = 45)
gpvec <- logLikVector(gpfit)
head(gpvec)
logLik(gpvec)
logLik(gpfit)
```

---

returnLevel   *Frequentist threshold-based inference for return levels*

Description

Calculates point estimates and confidence intervals for m-year return levels for stationary time series fitted extreme value model objects returned from [flite](#). Two types of interval may be returned: (a) intervals based on approximate large-sample normality of the maximum likelihood estimator for return level, which are symmetric about the point estimate, and (b) profile likelihood-based intervals based on an (adjusted) log-likelihood.

Usage

```
returnLevel(
  x,
  m = 100,
  level = 0.95,
  ny,
  prof = TRUE,
  inc = NULL,
  type = c("vertical", "cholesky", "spectral", "none")
)
```

Arguments

x	An object inheriting from class "flite" returned from [flite](#).
m	A numeric scalar. The return period, in years.

level	A numeric scalar in (0, 1). The confidence level required for confidence interval for the m-year return level.
ny	A numeric scalar. The (mean) number of observations per year. **Setting this appropriately is important**. See **Details**.
prof	A logical scalar. Should we calculate intervals based on profile log-likelihood?
inc	A numeric scalar. Only relevant if prof = TRUE. The increment in return level by which we move upwards and downwards from the MLE for the return level in the search for the lower and upper confidence limits. If this is not supplied then inc is set to one hundredth of the length of the symmetric confidence interval for return level.
type	A character scalar. The argument type to the function returned by the function [adjust_loglik](), that is, the type of adjustment made to the independence log-likelihood function in creating an adjusted log-likelihood function. See **Details** and **Value** in [adjust_loglik]().

Details

For information about return levels see the "Introducing lite" vignette.

ny provides information about the (intended) frequency of sampling in time, that is, the number of observations that would be observed in a year if there are no missing values. If the number of observations may vary between years then ny should be set equal to the mean number of observations per year.

Supplying ny**.** The value of ny may have been set in the call to [flite](). If ny is supplied by the user in the call to returnLevel then this will be used in preference to the value stored in the fitted model object. If these two values differ then no warning will be given.

For details of the definition and estimation of return levels see the Inference for return levels vignette.

The profile likelihood-based intervals are calculated by reparameterising in terms of the m-year return level and estimating the values at which the (adjusted) profile log-likelihood reaches the critical value logLik(x) - 0.5 * stats::qchisq(level, 1). This is achieved by calculating the profile log-likelihood for a sequence of values of this return level as governed by inc. Once the profile log-likelihood drops below the critical value the lower and upper limits are estimated by interpolating linearly between the cases lying either side of the critical value. The smaller inc the more accurate (but slower) the calculation will be.

Value

A object (a list) of class "returnLevel", "lite" with the components

rl_sym,rl_prof	
	Named numeric vectors containing the respective lower 100level% limit, the MLE and the upper 100level% limit for the return level. If prof = FALSE then rl_prof will be missing.
rl_se	Estimated standard error of the return level.
max_loglik,crit,for_plot	
	If prof = TRUE then these components will be present, containing respectively: the maximised log-likelihood; the critical value and a matrix with return levels

in the first column (`ret_levs`) and the corresponding values of the (adjusted)
profile log-likelihood (`prof_loglik`).

`m,level`	The input values of `m` and `level`.
`ny`	The value of `ny` used to infer the return level.
`call`	The call to `returnLevel`.

References

Coles, S. G. (2001) *An Introduction to Statistical Modeling of Extreme Values*, Springer-Verlag,
London. doi: 10.1007/9781447136750_3

See Also

`returnLevelMethods`, including plotting the (adjusted) profile log-likelihood for a return level.

Examples

```
### Cheeseboro wind gusts

# Make inferences
cdata <- exdex::cheeseboro
# Each column of the matrix cdata corresponds to data from a different year
# flite() sets cluster automatically to correspond to column (year)
cfit <- flite(cdata, u = 45, k = 3)

# These data are hourly for one month (January) year so ny = 31 * 24
# Large inc set here for speed, sacrificing accuracy
# Default 95% confidence intervals
rl <- returnLevel(cfit, inc = 2.5, ny = 31 * 24)
summary(rl)
rl
oldrl <- plot(rl)
oldrl

# Quickly recalculate/replot the intervals based on profile log-likelihood
# provided that level is smaller than that used to produce rl
newrl <- plot(rl, level = 0.9)
newrl
```

---

returnLevelMethods   *Methods for objects of class* `"returnLevel"`

Description

Methods for objects of class `"returnLevel"` returned from `returnLevel`.

Usage

```
## S3 method for class 'returnLevel'
plot(x, level = NULL, legend = TRUE, digits = 3, plot = TRUE, ...)

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

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

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

Arguments

x	an object of class c("returnLevel", "lite"), a result of a call to [returnLevel](#), using prof = TRUE.
level	A numeric scalar in (0, 1). The confidence level required for the confidence interval for the m-year return level. If level is not supplied then x$level is used. level must be no larger than x$level.
legend	A logical scalar. Should we add a legend (in the top right of the plot) that gives the approximate values of the MLE and 100level% confidence limits?
digits	For plot.returnLevel: an integer. Passed to [signif](#) to round the values in the legend. For print.returnLevel: the argument digits to [print.default](#). For summary.returnLevel: an integer. For number formatting with [signif](#). If digits is not specified (i.e. [missing](#)) then signif() will not be called (i.e. no rounding will be performed).
plot	A logical scalar. If TRUE then the plot is produced. Otherwise, it is not, but the MLE and confidence limits are returned.
...	For plot.returnLevel: arguments passed to [plot](#), such as graphical parameters. For print.summary.returnLevel: arguments passed to [print.default](#).
object	an object of class c("returnLevel", "lite"), a result of a call to [returnLevel](#), using prof = TRUE.

Details

plot.returnLevel plots the profile log-likelihood for a return level, provided that x returned by a call to [returnLevel](#) using prof = TRUE. Horizontal lines indicate the values of the maximised log-likelihood and the critical level used to calculate the confidence limits. If level is smaller than x$level then approximate 100level% confidence limits are recalculated based on the information contained in x$for_plot.

print.returnLevel prints the call to [returnLevel](#) and the estimates and 100x$level% confidence limits for the x$m-year return level.

Value

`plot.returnLevel`: a numeric vector of length 3 containing the lower 100level% confidence limit, the MLE and the upper 100level% confidence limit is returned invisibly.

`print.returnLevel`: the argument x is returned, invisibly.

`summary.returnLevel`: a list containing the list element `object$call` and a matrix `matrix` containing the MLE and estimated SE of the return level.

`print.summary.returnLevel`: the argument x is returned, invisibly.

Examples

See [returnLevel](#).

See Also

[returnLevel](#) to perform frequentist threshold-based inference for return levels.

Index