Package: lite (via r-universe)

September 15, 2024

Title Likelihood-Based Inference for Time Series Extremes

Version 1.1.1

Date 2024-07-17

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 extreme values. Both maximum likelihood and Bayesian approaches are available.

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

License GPL (>= 2)

Encoding UTF-8

Depends R (>= 3.3.0)

RoxygenNote 7.2.3

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

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

RemoteUrl https://github.com/paulnorthrop/lite

RemoteRef HEAD

RemoteSha 1efc46871f00526bef4b37ee0c1773b276252f22

Contents

lite-package 2
Bernoulli 3
blite 4
bliteMethods 7
estfun 10
flite 11
fliteMethods 14
generalisedPareto 17
logLikVector 19
predict.blite 20
returnLevel 23
returnLevelMethods 25

Index **28**

`lite-package` *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` and `blite`, which perform frequentist and Bayesian inference for time series extremes, respectively.

See the vignettes `vignette("lite-1-frequentist", package = "lite")` and
`vignette("lite-2-bayesian", package = "lite")` for an overview of the package.

Author(s)

Maintainer: Paul J. Northrop <`p.northrop@ucl.ac.uk`> [copyright holder]

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.

`blite` for Bayesian threshold-based inference for time series extremes.

`predict.blite` for predictive inference for the largest value observed in N years.

`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)
```

blite *Bayesian threshold-based inference for time series extremes*

Description

Performs threshold-based Bayesian 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

```
blite(
  data,
  u,
  cluster,
  k = 1,
  inc_cens = TRUE,
  ny,
  gp_prior = revdbayes::set_prior(prior = "mdi", model = "gp"),
  b_prior = revdbayes::set_bin_prior(prior = "jeffreys"),
  theta_prior_pars = c(1, 1),
  n = 1000,
  type = c("vertical", "none"),
  ...
)
```

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` isvused 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 predictive inferences using `predict.blite`, but `ny` is not used by `blite` so it need not be supplied now. If `ny` is supplied to `blite` then it is stored for use by `predict.blite`. Alternatively, `ny` can be supplied in a later call to `predict.blite`. If `ny` is supplied to both `blite`

	and predict.blite then the value supplied to predict.blite will take precedence, with no warning given.
gp_prior	A list to specify a prior distribution for the GP parameters (σ_u, ξ), set using set_prior.
b_prior	A list to specify a prior distribution for the Bernoulli parameter σ_u, set using set_bin_prior.
theta_prior_pars	A numerical vector of length 2 containing the respective values of the parameters α and β of a Beta(α, β) prior for the extremal index θ.
n	An integer scalar. The size of posterior sample required.
type	A character scalar. Either `"vertical"` to adjust the independence log-likelihood vertically, or `"none"` for no adjustment. Horizontal adjustment is not offered because it does not preserve the correct support of the posterior distribution.
...	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

See flite for details of the (adjusted) likelihoods on which these Bayesian inferences are based.

The likelihood is based on a model for 3 independent aspects.

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).

The contributions to the likelihood for p_u and (σ_u, ξ) are based on the vertically-adjusted likelihoods described in flite. This is an example of Bayesian inference using a composite likelihood Ribatet et al (2012). Priors for p_u (σ_u, ξ) and θ are set using the arguments `gp_prior`, `b_prior` and `theta_prior_pars`. Currently, only priors where p_u (σ_u, ξ) and θ are independent a priori are allowed.

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.

Random samples are simulated from the posteriors for p_u and (σ_u, ξ) (using ru) and θ (using kgaps_post).

Value

An object of class `c("blite", "lite", "chandwich")`. This object is an `n` $\times 4$ matrix containing the posterior samples, with column names `c("p[u]", "sigma[u]", "xi", "theta")`.

The object also has the attributes `"Bernoulli"`, `"gp"`, `"theta"`, which provide the fitted model objects returned from adjust_loglik (for `"Bernoulli"` and `"gp"`) and kgaps (for `"theta"`). The named input arguments are returned in a list as the attribute `inputs`. If `ny` was not supplied then its value is `NA`. The call to `blite` is provided in the attribute `"call"`. A call to flite is used to create adjusted log-likelihoods for p_u and (σ_u, ξ). The object returned from the call is provided as the attribute "flite_object".

Objects inheriting from class "blite" have coef, nobs, plot, summary, vcov and confint methods. See bliteMethods.

predict.blite can be used to make predictive inferences about the largest value to be observed in N years.

References

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

Ribatet, M., Cooley, D., & Davison, A. C. (2012). Bayesian inference from composite likelihoods, with an application to spatial extremes. *Statistica Sinica*, **22**(2), 813-845.

See Also

bliteMethods, including plotting the posterior samples.

predict.blite to make predictive inferences about future extreme values.

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

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

Examples

```
### Cheeseboro wind gusts

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

## Plots of posterior samples
plot(cpost)

## Credible intervals
confint(cpost)
```

---

bliteMethods *Methods for objects of class* "blite"

Description

Methods for objects of class "blite" returned from blite. confint.blite is a misnomer: it returns (equi-tailed) Bayesian credible intervals.

Usage

```
## S3 method for class 'blite'
plot(x, which = c("all", "pu", "gp", "xi", "theta"), ...)

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

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

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

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

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

## S3 method for class 'blite'
confint(object, parm = "all", level = 0.95, ...)
```

Arguments

`x`	An object inheriting from class `"blite"`, a result of a call to `blite`.
`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**.
`...`	For `plot.blite`: arguments passed to `plot`, such as graphical parameters. For `coef.blite`: additional arguments passed to `fun`. For `print.summary.blite`: additional arguments passed to `print.default`. Otherwise `...` is unused.
`object`	An object of class `"blite"`, returned by `blite`.
`fun`	A summary function to be applied to each column of the simulated values in `object`. If `fun` is missing then `mean` is used.
`short`	A logical scalar that determines the form of the output. See **Details**.
`mean`	A logical scalar. Determines the form of the output if `short = TRUE`. See **Details**.
`digits`	An integer. Passed to `signif` to round the values in the summary.

`parm`	A character vector specifying the parameters for which confidence intervals are required. The default, `which = "all"`, produces confidence intervals for all the parameters, that is, p_u, σ_u, ξ and θ. If `which = "gp"` then intervals are produced only for σ_u and ξ. Otherwise, `parm` must be a subset of `c("pu", "sigmau", "xi", "theta")`.
`level`	The credible level required. A numeric scalar in (0, 1).

Details

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

- Top left: histogram of the posterior sample for the threshold exceedance probability p_u.
- Top right: scatter plot of posterior sample for the GP parameters (σ_u, ξ). The linear constraint $\xi > -\sigma_u/x_{(n)}$ is drawn on the plot.
- Bottom left: histogram of the posterior sample for the GP shape parameter ξ.
- Bottom right: histogram of the posterior sample for the extremal index θ.

Value

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

`coef.blite`: a numeric vector of length 4 with names `c("p[u]", "sigma[u]", "xi", "theta")`. The values of summary statistics calculated using the function `fun`.

`vcov.blite`: a 4×4 matrix with row and column names `c("p[u]", "sigma[u]", "xi", "theta")`. An estimate of the posterior covariance matrix, calculated using `cov`.

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

`summary.blite`: an object containing the original function call and a matrix of summaries of the posterior samples for each of the parameters. If `short = TRUE` then there are 2 columns, containing either the sample posterior means and standard deviations (`mean = TRUE`) or the sample posterior medians and inter-quartile ranges (`mean = FALSE`). If `short = FALSE` then there are 4 columns, with each column containing the usual 6-number summary produced by `summary`. The object is printed by `print.summary.blite`.

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

`confint.blite`: a numeric matrix with 2 columns giving the lower and upper credible limits for each parameter. These columns are labelled as `(1-level)/2` and `1-(1-level)/2`, expressed as a percentage, by default `2.5%` and `97.5%`. The row names are the names of the parameters supplied in `parm`.

See Also

`blite` to perform frequentist threshold-based inference for time series extremes.

`predict.blite`: for predictive inference for the largest value observed in N years.

`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 `m`th 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, with no warning given.
`...`	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 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, σ_u, ξ, θ). 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, σ_u, ξ, θ),

- 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", "theta", which provide the fitted model objects returned from adjust_loglik (for "Bernoulli" and "gp") and kgaps (for "theta"). The named input arguments are returned in a list as the attribute inputs. If ny was not supplied then its value is NA. The call to flite is provided in the attribute "call".

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

returnLevel can be used to make frequentist inferences about return levels.

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 $(p_u, \sigma_u, \xi, \theta)$.

returnLevel to make frequentist inferences about return levels.

blite for Bayesian threshold-based inference for time series extremes.

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")

## Confidence intervals
# Based on an adjusted profile log-likelihood
confint(cfit)
# Symmetric intervals based on large sample normality
confint(cfit, profile = FALSE)
```

---

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, ...)

## S3 method for class 'flite'
confint(
  object,
  parm = "all",
  level = 0.95,
  adj_type = c("vertical", "none", "cholesky", "spectral"),
  profile = TRUE,
  ...
)
```

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 loglikelihood. 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`. For `confint.flite`: additional arguments passed to `conf_intervals`. Otherwise `...` is unused.
`object`	An object of class `"flite"`, returned by `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.
`parm`	A character vector specifying the parameters for which confidence intervals are required. The default, `which = "all"`, produces confidence intervals for all the parameters, that is, p_u, σ_u, ξ and θ. If `which = "gp"` then intervals are produced only for σ_u and ξ. Otherwise, `parm` must be a subset of `c("pu", "sigmau", "xi", "theta")`.
`level`	The confidence level required. A numeric scalar in (0, 1).
`profile`	A logical scalar. If `TRUE` then confidence intervals based on an (adjusted) profile loglikelihood are returned. If `FALSE` then intervals based on approximate large sample normal theory, which are symmetric about the MLE, are returned.

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.

`confint.flite`: a numeric matrix with 2 columns giving the lower and upper confidence limits for each parameter. These columns are labelled as `(1-level)/2` and `1-(1-level)/2`, expressed as a percentage, by default `2.5%` and `97.5%`. The row names are the names of the parameters supplied in `parm`.

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 `m`th 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)
```

`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)
```

`predict.blite` *Predictive inference for the largest value observed in N years.*

Description

`predict` method for class "blite". Performs predictive inference about the largest value to be observed over a future time period of N years. Predictive inferences accounts for uncertainty in model parameters and for uncertainty owing to the variability of future observations.

Usage

```
## S3 method for class 'blite'
predict(
  object,
  type = c("i", "p", "d", "q", "r"),
  x = NULL,
  x_num = 100,
  n_years = 100,
  ny = NULL,
  level = 95,
  hpd = FALSE,
  lower_tail = TRUE,
  log = FALSE,
  big_q = 1000,
  ...
)
```

Arguments

`object`	An object of class `"blite"` returned from `blite`.
`type`	A character vector. Indicates which type of inference is required: • "i" for predictive intervals, • "p" for the predictive distribution function, • "d" for the predictive density function, • "q" for the predictive quantile function, • "r" for random generation from the predictive distribution.
`x`	A numeric vector or a matrix with `n_years` columns. The meaning of `x` depends on `type`. • `type = "p"` or `type = "d"`: `x` contains quantiles at which to evaluate the distribution or density function. No element of `x` can be less than the threshold `attr(object, "inputs")$u`. If `x` is not supplied then `n_year`-specific defaults are set: vectors of length `x_num` from the 0.1% quantile to the 99% quantile, subject all values being greater than the threshold. • `type = "q"`: `x` contains probabilities in (0,1) at which to evaluate the quantile function. Any values outside (0, 1) will be removed without warning. No element of `p` can correspond to a predictive quantile that is below the threshold, `attr(object, "inputs")$u`. That is, no element of `p` can be less than the value of `predict.evpost(object, type = "q", x = attr(object, "inputs")$u)`. If `x` is not supplied then a default value of `c(0.025, 0.25, 0.5, 0.75, 0.975)` is used. • `type = "i"` or `type = "r"`: `x` is not relevant.
`x_num`	A numeric scalar. If `type = "p"` or `type = "d"` and `x` is not supplied then `x_num` gives the number of values in `x` for each value in `n_years`.
`n_years`	A numeric vector. Values of N.
`ny`	A numeric scalar. The (mean) number of observations per year. **Setting this appropriately is important**. See **Details**.
`level`	A numeric vector of values in (0, 100). Only relevant when `type = "i"`. Levels of predictive intervals for the largest value observed in N years, i.e. level% predictive intervals are returned.
`hpd`	A logical scalar. Only relevant when `type = "i"`. If `hpd = FALSE` then the interval is equi-tailed, with its limits produced by `predict.evpost(object, type ="q", x = p)`, where `p = c((1-level/100)/2, (1+level/100)/2)`. If `hpd = TRUE` then, in addition to the equi-tailed interval, the shortest possible level% interval is calculated. If the predictive distribution is unimodal then this is a highest predictive density (HPD) interval.
`lower_tail`	A logical scalar. Only relevant when `type = "p"` or `type = "q"`. If TRUE (default), (output or input) probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.
`log`	A logical scalar. Only relevant when `type = "d"`. If TRUE the log-density is returned.

`big_q`	A numeric scalar. Only relevant when `type = "q"`. An initial upper bound for the desired quantiles to be passed to `uniroot` (its argument `upper`) in the search for the predictive quantiles. If this is not sufficiently large then it is increased until it does provide an upper bound.
`...`	Additional optional arguments. At present no optional arguments are used.

Details

The function `predict.evpost` in the `revdbayes` package is used to perform the predictive inferences. The effect of adjusting for the values of the extremal index θ in the posterior sample in `object$sim_vals[, "theta"]` is to change the effective time horizon from N to θN.

`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 `blite`. If `ny` is supplied by the user in the call to `predict.blite` 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.

Value

An object of class "evpred", a list containing a subset of the following components:

`type` The argument `type` supplied to `predict.blite`. Which of the following components are present depends `type`.

`x` A matrix containing the argument `x` supplied to `predict.blite`, or set within `predict.blite` if `x` was not supplied, replicated to have `n_years` columns if necessary. Only present if `type` is `"p"`, `"d"` or `"q"`.

`y` The content of `y` depends on `type`:

- `type = "p"`, `"d"`, `"q"`: A matrix with the same dimensions as `x`. Contains distribution function values (`type = "p"`), predictive density (`type = "d"`) or quantiles (`type = "q"`).
- `type = "r"`: A numeric matrix with `length(n_years)` columns and number of rows equal to the size of the posterior sample.
- `type = "i"`: `y` is not present.

`long` A `length(n_years)*length(level)` by 4 numeric matrix containing the equi-tailed limits with columns: lower limit, upper limit, n_years, level. Only present if `type = "i"`. If an interval extends below the threshold then `NA` is returned.

`short` A matrix with the same structure as `long` containing the HPD limits. Only present if `type = "i"`. Columns 1 and 2 contain `NA`s if `hpd = FALSE` or if the corresponding equi-tailed interval extends below the threshold.

The arguments `n_years`, `level`, `hpd`, `lower_tail`, `log` supplied to `predict.blite` are also included, as is the value of `ny` and `model = "bingp"`.

Examples

```
### Cheeseboro wind gusts

cdata <- exdex::cheeseboro
# Each column of the matrix cdata corresponds to data from a different year
# blite() sets cluster automatically to correspond to column (year)
cpost <- blite(cdata, u = 45, k = 3, ny = 31 * 24)

# Interval estimation
predict(cpost)$long
predict(cpost, hpd = TRUE)$short

# Density function
plot(predict(cpost, type = "d", n_years = c(100, 1000)))

# Distribution function
plot(predict(cpost, type = "p", n_years = c(100, 1000)))

# Quantiles
predict(cpost, type = "q", n_years = c(100, 1000))$y

# Random generation
plot(predict(cpost, type = "r"))
```

---

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 = 1/100,
  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 in (0, 1/2]. Only relevant if `prof = TRUE`. The increment, a fraction of the length of the symmetric confidence interval for the `m`-year 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.
`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 100`level`% limit, the MLE and the upper 100`level`% 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 = 1 / 10, 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 100`level`% 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$level` then approximate 100`level`% confidence limits are recalculated based on the information contained in `x$for_plot`.

`print.returnLevel` prints the call to `returnLevel` and the estimates and 100`x$level`% confidence limits for the `x$m`-year return level.

Value

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

_PACKAGE (lite-package), 2

adjust_loglik, *5*, *6*, *10–13*, *19*, *24*

Bernoulli, 3, *10*, *13*, *20*
blite, *2*, *3*, 4, *7–9*, *13*, *21*, *22*
bliteMethods, *7*, 7

chandwich, *12*
choose_uk, *6*, *7*, *12*, *13*
coef.Bernoulli (Bernoulli), 3
coef.blite (bliteMethods), 7
coef.flite (fliteMethods), 14
coef.GP (generalisedPareto), 17
conf_intervals, *15*
conf_region, *15*
confint.blite (bliteMethods), 7
confint.flite (fliteMethods), 14
cov, *9*

estfun, *10*, 10
exdex, *6*, *12*

fitBernoulli, *10*
fitBernoulli (Bernoulli), 3
fitGP, *10*
fitGP (generalisedPareto), 17
flite, *2*, *3*, *6*, *7*, *10*, 11, *14–16*, *19*, *23*, *24*
fliteMethods, *13*, 14

generalisedPareto, *10*, *13*, 17, *20*
gpObsInfo (generalisedPareto), 17
grimshaw_gp_mle, *12*, *18*

kgaps, *5*, *6*, *12*, *13*
kgaps_post, *6*

lite-package, 2
logLik.Bernoulli (Bernoulli), 3
logLik.flite (fliteMethods), 14
logLik.GP (generalisedPareto), 17
logLik.logLikVector (logLikVector), 19
logLikVector, 19

mean, *8*
meat, *10*, *17*
meatCL, *5*, *6*, *10–12*, *17*
missing, *26*

na.omit, *3*, *4*, *17*
nobs.Bernoulli (Bernoulli), 3
nobs.blite (bliteMethods), 7
nobs.flite (fliteMethods), 14
nobs.GP (generalisedPareto), 17

plot, *8*, *15*, *26*
plot.blite (bliteMethods), 7
plot.flite, *13*
plot.flite (fliteMethods), 14
plot.returnLevel (returnLevelMethods), 25
predict.blite, *3*, *5–7*, *9*, 20
predict.evpost, *22*
print.default, *8*, *15*, *26*
print.returnLevel (returnLevelMethods), 25
print.summary.blite, *9*
print.summary.blite (bliteMethods), 7
print.summary.flite, *16*
print.summary.flite (fliteMethods), 14
print.summary.returnLevel (returnLevelMethods), 25

returnLevel, *3*, *12*, *13*, 23, *25–27*
returnLevelMethods, *25*, 25
revdbayes, *12*, *22*
ru, *6*

set_bin_prior, *6*
set_prior, *6*
signif, *8*, *15*, *26*
split_by_NAs, *5*, *11*

summary, *9*
summary.blite *(*bliteMethods*)*, 7
summary.flite *(*fliteMethods*)*, 14
summary.returnLevel
 *(*returnLevelMethods*)*, 25

uniroot, *22*

vcov.Bernoulli *(*Bernoulli*)*, 3
vcov.blite *(*bliteMethods*)*, 7
vcov.flite *(*fliteMethods*)*, 14
vcov.GP *(*generalisedPareto*)*, 17