Package 'lax'

October 21, 2025

Title Loglikelihood Adjustment for Extreme Value Models

Version 1.2.4

Date 2025-10-21

Description Performs adjusted inferences based on model objects fitted, using maximum likelihood estimation, by the extreme value analysis packages 'eva' <https://cran.r-project.org/package=eva>, 'evd' <https://cran.r-project.org/package=evd>, 'evir' <https://cran.r-project.org/package=evir>, 'extRemes' <https://cran.r-project.org/package=extRemes>, 'fExtremes' <https://cran.r-project.org/package=fExtremes>, 'ismev' <https://cran.r-project.org/package=ismev>, 'mev' <https://cran.r-project.org/package=mev>, 'POT' <https://cran.r-project.org/package=POT> and 'texmex' <https://cran.r-project.org/package=texmex>. Adjusted standard errors and an adjusted loglikelihood are provided, using the 'chandwich' package <https://cran.r-project.org/package=chandwich> and the object-oriented features of the 'sandwich' package <https://cran.r-project.org/package=sandwich>. The adjustment is based on a robust sandwich estimator of the parameter covariance matrix, based on the methodology in Chandler and Bate (2007) <doi:10.1093/biomet/asm015>. This can be used for cluster correlated data when interest lies in the parameters of the marginal distributions, or for performing inferences that are robust to certain types of model misspecification. Univariate extreme value models, including regression models, are supported.

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

License GPL (>= 2)

LazyData TRUE

Encoding UTF-8

Depends R (>= 3.3.0)

RoxygenNote 7.3.3

Suggests distillery, eva, evd, evir, extRemes, fExtremes, ismev, knitr, mev, POT, rmarkdown, testthat, texmex

VignetteBuilder knitr

URL https://paulnorthrop.github.io/lax/,
https://github.com/paulnorthrop/lax

BugReports https://github.com/paulnorthrop/lax/issues

Config/testthat/edition 3

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2025-10-21 20:20:02 UTC

Contents

lax-package *lax: Loglikelihood Adjustment for Extreme Value Models*

Description

Performs adjusted inferences based on model objects fitted, using maximum likelihood estimation, by the extreme value analysis packages eva, evd, evir, extRemes, fExtremes, ismev, mev, POT and texmex. Univariate extreme value models, including regression models, are supported. Adjusted standard errors and an adjusted loglikelihood are provided, using the chandwich package and the object-oriented features of the sandwich package.

Details

The adjustment is based on a robust sandwich estimator of the parameter covariance matrix, based on the methodology in Chandler and Bate (2007). This can be used for cluster correlated data when interest lies in the parameters of the marginal distributions, or for performing inferences that are robust to certain types of model misspecification.

The main function is `alogLik`, which works in an object-oriented way, operating on fitted model objects. This function performs the loglikelihood adjustments using `adjust_loglik`. See the following package-specific help pages for details and examples: `eva`, `evd`, `evir`, `extRemes`, `fExtremes`, `ismev`, `mev`, `POT`, `texmex`.

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

Author(s)

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

Authors:

- Camellia Yin [copyright holder]

References

Bader, B. and Yan, J. (2020). eva: Extreme Value Analysis with Goodness-of-Fit Testing. R package version 0.2.6. https://CRAN.R-project.org/package=eva

Belzile, L., Wadsworth, J. L., Northrop, P. J., Grimshaw, S. D. and Huser, R. (2019). mev: Multivariate Extreme Value Distributions. R package version 1.12.2. https://github.com/lbelzile/mev/

Berger S., Graham N., Zeileis A. (2017). Various Versatile Variances: An Object-Oriented Implementation of Clustered Covariances in R. Technical Report 2017-12, Working Papers in Economics and Statistics, Research Platform Empirical and Experimental Economics, Universitat Innsbruck. https://EconPapers.RePEc.org/RePEc:inn:wpaper:2017-12.

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

Gilleland, E. and Katz, R. W. (2016). extRemes 2.0: An Extreme Value Analysis Package in R. *Journal of Statistical Software*, **72**(8), 1-39. doi:10.18637/jss.v072.i08

Northrop, P. J. and Chandler, R. E. (2018). chandwich: Chandler-Bate Sandwich Loglikelihood Adjustment. R package version 1.1. https://CRAN.R-project.org/package=chandwich.

Pfaff, B. and McNeil, A. (2018). evir: Extreme Values in R. R package version 1.7-4. https://CRAN.R-project.org/package=evir

Ribatet, M. and Dutang, C. (2019). POT: Generalized Pareto Distribution and Peaks Over Threshold. R package version 1.1-7. https://CRAN.R-project.org/package=POT

Southworth, H., Heffernan, J. E. and Metcalfe, P. D. (2017). texmex: Statistical modelling of extreme values. R package version 2.4. https://CRAN.R-project.org/package=texmex.

Stephenson, A. G. evd: Extreme Value Distributions. *R News*, **2**(2):31-32, June 2002. https://CRAN.R-project.org/doc/Rnews/

Stephenson, A. G., Heffernan, J. E. and Gilleland, E. (2018). ismev: An Introduction to Statistical Modeling of Extreme Values. R package version 1.42. https://CRAN.R-project.org/package=ismev.

Wuertz, D., Setz, T. and Chalabi, Y. (2017). fExtremes: Rmetrics - Modelling Extreme Events in Finance. R package version 3042.82. https://CRAN.R-project.org/package=fExtremes

Zeileis A. (2004). Econometric Computing with HC and HAC Covariance Matrix Estimators. *Journal of Statistical Software*, **11**(10), 1-17. doi:10.18637/jss.v011.i10.

Zeileis A. (2006). Object-Oriented Computation of Sandwich Estimators. *Journal of Statistical Software*, **16**(9), 1-16. doi:10.18637/jss.v016.i09.

See Also

Useful links:

- https://paulnorthrop.github.io/lax/
- https://github.com/paulnorthrop/lax
- Report bugs at https://github.com/paulnorthrop/lax/issues

`alogLik` *Loglikelihood adjustment for model fits.*

Description

This function is generic. It performs adjustment of the loglikelihood associated with fitted model objects, following Chandler and Bate (2007). Certain classes of extreme value model objects are supported automatically. For details see the `alogLik` help pages for the packages: `evd`, `evir`, `extRemes`, `fExtremes`, `ismev`, `mev`, `POT`, `texmex`. User-supplied objects can also be supported: the requirements for these objects are explained in **Details**.

Usage

```
alogLik(
  x,
  cluster = NULL,
  use_vcov = TRUE,
  binom = FALSE,
  k,
  inc_cens = TRUE,
  ...
)
```

Arguments

x	A fitted model object with certain associated S3 methods. See **Details**.
cluster	A vector or factor indicating from which cluster the respective log-likelihood contributions from loglik originate. The length of cluster must be consistent with the estfun method to be used in the estimation of the 'meat' V of the sandwich estimator of the covariance matrix of the parameters to be passed to adjust_loglik. In most cases, cluster must have length equal to the number of observations in data. The exception is the GP (only) model (binom = FALSE), where the cluster may either contain a value for each observation in the raw data, or for each threshold exceedance in the data. If cluster is not supplied (is NULL) then it is assumed that each observation forms its own cluster. See **Details** for further details.
use_vcov	A logical scalar. Should we use the vcov S3 method for x (if this exists) to estimate the Hessian of the independence loglikelihood to be passed as the argument H to adjust_loglik? Otherwise, H is estimated inside adjust_loglik using optimHess.
binom	A logical scalar. This option is only relevant to **GP models** and is only available in the **stationary** (no covariates) case. If binom = FALSE then loglikelihood adjustment is only performed using the GP model. If binom = TRUE then loglikelihood adjustment is also performed for inferences about the probability of threshold exceedance, using a Bernoulli model for the instances of threshold exceedance.
k	A non-negative integer scalar. This option is only relevant to **GP models** and is only available in the **stationary** (no covariates) case. If k is supplied then it is passed as the run parameter K to kgaps for making inferences about the extremal index θ using the K-gaps model of Suveges and Davison (2010).
inc_cens	A logical scalar. This argument is only relevant if k is supplied. Passed to kgaps to indicate whether or not to include censored inter-exceedance times, relating to the first and last observations.
...	Further arguments to be passed to the functions in the sandwich package meat (if cluster = NULL), or meatCL (if cluster is not NULL).

Details

Object x *must* have the following S3 methods:

- logLikVec: returns a vector of the contributions to the independence loglikelihood from individual observations;
- coef: returns a vector of model coefficients, see coef;
- nobs: returns the number of (non-missing) observations used in a model fit, see nobs;

and *may* have the following S3 methods

- vcov: returns the estimated variance-covariance matrix of the (main) parameters of a fitted model, see vcov;
- estfun: returns an n by k matrix, in which each column gives the derivative of the loglikelihood at each of n observation with respect to the k parameters of the model, see estfun.

Loglikelihood adjustment is performed using the adjust_loglik function in the chandwich package. The relevant arguments to adjust_loglik, namely loglik, mle, H and V, are created based on the class of the object x.

If a vcov method is not available, or if use_vcov = FALSE, then the variance-covariance matrix of the MLE (from which H is calculated) is estimated inside adjust_loglik using optimHess.

The sandwich package is used to estimate the variance matrix V of the score vector: meat is used if cluster = NULL; meatCL is used if cluster is not NULL. If cluster is NULL then any arguments of meatCL present in ... will be ignored. Similarly, if cluster is not NULL then any arguments of meat present in ... will be ignored. meat and meatCL require an estfun method to be available, which, in the current context, provides matrix of score contributions. If a bespoke estfun method is not provided then this is constructed by estimating the score contributions using jacobian.

Value

An object inheriting from class "chandwich". See adjust_loglik.

The original fitted model object is available as an attribute named "original_fit", accessible using attr(name, "original_fit"), where name is the name of the object to which the object returned from alogLik is assigned.

If binom = TRUE then the returned object has an extra attribute named pu_aloglik that contains an object inheriting from class "chandwich" relating specifically to inferences about the probability of threshold exceedance. Also, the 4th component of the class of the returned object becomes "bin-gpd".

If k is supplied then the returned object has an extra attribute named theta that contains an object inheriting from class c("kgaps", "exdex") relating specifically to inferences about the extremal index θ. See the **Value** section in kgaps.

If x is one of the supported models then the class of the returned object is a vector of length 5. The first 3 components are c("lax", "chandwich", "name_of_package"), where "name_of_package" is the name of the package from which the input object x originated. The remaining 2 components depend on the model that was fitted. See the documentation of the relevant package for details: evd, evir, extRemes, fExtremes, ismev, mev, POT, texmex.

Otherwise, the class of the returned object is c("lax", "chandwich", class(x)).

Objects returned from 'aloglik' have 'anova', 'coef', 'confint', 'logLik', 'nobs', 'plot', 'print', 'summary' and 'vcov' methods.

Examples

See the (package-specific) examples in evd, evir, extRemes,fExtremes, ismev, mev, POT and texmex.

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

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

Zeileis (2006) Object-Oriented Computation and Sandwich Estimators. *Journal of Statistical Software*, **16**, 1-16. doi:10.18637/jss.v016.i09

See Also

summary.chandwich, plot.chandwich, confint.chandwich, anova.chandwich, coef.chandwich, vcov.chandwich and logLik.chandwich for S3 methods for objects of class "chandwich".

conf_region for confidence regions for pairs of parameters.

adjust_loglik in the chandwich package to adjust a user-supplied loglikelihood.

meat and meatCL in the sandwich package.

---

anova.lax *Comparison of nested models*

Description

anova method for objects of class "lax". Compares two or more nested models using the adjusted likelihood ratio test statistic (ALRTS) described in Section 3.5 of Chandler and Bate (2007). The nesting must result from the simple constraint that a subset of the parameters of the larger model is held fixed.

Usage

```
## S3 method for class 'lax'
anova(object, object2, ...)
```

Arguments

object	An object of class "lax", inheriting from class "chandwich", returned by alogLik.
object2	An object of class "lax", inheriting from class "chandwich", returned by alogLik.
...	Further objects of class "lax" and/or arguments to be passed to anova.chandwich, and then on to compare_models, in particular type, which chooses the type of adjustment.

Details

The objects of class "lax" need not be provided in nested order: they will be ordered inside anova.lax based on the values of attr(., "p_current").

Value

An object of class "anova" inheriting from class "data.frame", with four columns:

Model.Df	The number of parameters in the model
Df	The decrease in the number of parameter compared the model in the previous row
ALRTS	The adjusted likelihood ratio test statistic
Pr(>ALRTS)	The p-value associated with the test that the model is a valid simplification of the model in the previous row.

The row names are the names of the model objects.

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

See Also

anova.chandwich: the anova method on which anova.lax is based.

alogLik: loglikelihood adjustment for model fits.

Examples

```
got_evd <- requireNamespace("evd", quietly = TRUE)
if (got_evd) {
  library(evd)
  small <- fgev(ow$temp, nsloc = ow[, "loc"])
  adj_small <- alogLik(small, cluster = ow$year)
  tiny <- fgev(ow$temp)
  adj_tiny <- alogLik(tiny, cluster = ow$year)
  anova(adj_small, adj_tiny)

  set.seed(4082019)
  uvdata <- evd::rgev(100, loc = 0.13, scale = 1.1, shape = 0.2)
  M0 <- fgev(uvdata)
  M1 <- fgev(uvdata, nsloc = (-49:50)/100)
  adj0 <- alogLik(M0)
  adj1 <- alogLik(M1)
  anova(adj1, adj0)
}

got_extRemes <- requireNamespace("extRemes", quietly = TRUE)
if (got_extRemes) {
  library(extRemes)
```

```
  large <- fevd(temp, ow, location.fun = ~ loc, scale.fun = ~ loc,
                shape.fun = ~ loc)
  medium <- fevd(temp, ow, location.fun = ~ loc, scale.fun = ~ loc)
  small <- fevd(temp, ow, location.fun = ~ loc)
  tiny <- fevd(temp, ow)
  adj_large <- alogLik(large, cluster = ow$year)
  adj_medium <- alogLik(medium, cluster = ow$year)
  adj_small <- alogLik(small, cluster = ow$year)
  adj_tiny <- alogLik(tiny, cluster = ow$year)
  anova(adj_large, adj_medium, adj_small, adj_tiny)
}
```

---

bernoulli *Inference for the Bernoulli distribution*

Description

Functions involved in making inferences about the probability of success in a Bernoulli distribution.

Usage

```
fit_bernoulli(data)

## S3 method for class 'bernoulli'
logLikVec(object, pars = NULL, ...)

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

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

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

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

## S3 method for class 'bernoulli'
alogLik(x, cluster = NULL, use_vcov = TRUE, ...)
```

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.
pars	A numeric parameter vector of length 1 containing the value of the Bernoulli success probability.

...	Further arguments to be passed to the functions in the sandwich package meat (if cluster = NULL), or meatCL (if cluster is not NULL).
x, object	A fitted model object returned from fit_bernoulli().
cluster	A vector or factor indicating from which cluster each observation in data originates.
use_vcov	A logical scalar. Should we use the vcov S3 method for x (if this exists) to estimate the Hessian of the independence loglikelihood to be passed as the argument H to adjust_loglik? Otherwise, H is estimated inside adjust_loglik using optimHess.

Details

fit_bernoulli: fit a Bernoulli distribution

logLikVec.bernoulli: calculates contributions to a loglikelihood based on the Bernoulli distribution. The loglikelihood is calculated up to an additive constant.

nobs, coef, vcov and logLik methods are provided.

Value

fit_bernoulli returns an object of class "bernoulli", a list with components: logLik, mle, nobs, vcov, 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.

logLikVec.bernoulli returns an object of class "logLikVec", a vector of length length(data) containing the loglikelihood contributions from the individual observations in data.

See Also

Binomial. The Bernoulli distribution is the special case where size = 1.

Examples

```
# Set up data
x <- exdex::newlyn
u <- quantile(x, probs = 0.9)
exc <- x > u

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

# Calculate the loglikelihood at the MLE
res <- logLikVec(fit)

# The logLik method sums the individual loglikelihood contributions.
logLik(res)

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

```
logLik(fit)

# Adjusted loglikelihood
# Create 5 clusters each corresponding approximately to 1 year of data
cluster <- rep(1:5, each = 579)[-1]
afit <- alogLik(fit, cluster = cluster, cadjust = FALSE)
summary(afit)
```

---

eva *Loglikelihood adjustment for eva fits*

Description

S3 `alogLik` method to perform loglikelihood adjustment for fitted extreme value model objects returned from the functions `gevrFit` and `gpdFit` in the eva package.

Usage

```
## S3 method for class 'gevrFit'
alogLik(x, cluster = NULL, use_vcov = TRUE, ...)

## S3 method for class 'gpdFit'
alogLik(x, cluster = NULL, use_vcov = TRUE, ...)
```

Arguments

`x`	A fitted model object with certain associated S3 methods. See **Details**.
`cluster`	A vector or factor indicating from which cluster the respective log-likelihood contributions from `loglik` originate. The length of `cluster` must be consistent with the `estfun` method to be used in the estimation of the 'meat' `V` of the sandwich estimator of the covariance matrix of the parameters to be passed to `adjust_loglik`. In most cases, `cluster` must have length equal to the number of observations in data. The exception is the GP (only) model (`binom = FALSE`), where the `cluster` may either contain a value for each observation in the raw data, or for each threshold exceedance in the data. If `cluster` is not supplied (is `NULL`) then it is assumed that each observation forms its own cluster. See **Details** for further details.
`use_vcov`	A logical scalar. Should we use the `vcov` S3 method for `x` (if this exists) to estimate the Hessian of the independence loglikelihood to be passed as the argument `H` to `adjust_loglik`? Otherwise, `H` is estimated inside `adjust_loglik` using `optimHess`.
`...`	Further arguments to be passed to the functions in the sandwich package `meat` (if `cluster = NULL`), or `meatCL` (if `cluster` is not `NULL`).

Details

See alogLik for details.

In the stationary case (no covariates) the function gevrFit and gpdFit in the eva package offer standard errors based on the expected information or on the observed information, via the argument information. In contrast, alogLik() always bases calculations on the observed information matrix. Therefore, unadjusted standard errors resulting from alogLik() may be different the corresponding standard errors from gevrFit or gpdFit.

For gevrFit only GEV fits (gumbel = FALSE) are supported.

Value

An object inheriting from class "chandwich". See adjust_loglik. class(x) is a vector of length 5. The first 3 components are c("lax", "chandwich", "eva"). The 4th component depends on which model was fitted. "rlarg" if gevrFit was used; "gpd" if gpdFit was used. The 5th component is "stat" if there are no covariates in the mode and "nonstat" otherwise.

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

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

Zeileis (2006) Object-Oriented Computation and Sandwich Estimators. *Journal of Statistical Software*, **16**, 1-16. doi:10.18637/jss.v016.i09

See Also

alogLik: loglikelihood adjustment for model fits.

Examples

```
# We need the eva package
got_eva <- requireNamespace("eva", quietly = TRUE)

if (got_eva) {
  library(eva)
  # An example from the eva::gpdFit documentation
  set.seed(7)
  x <- eva::rgpd(2000, loc = 0, scale = 2, shape = 0.2)
  mle_fit <- eva::gpdFit(x, threshold = 4, method = "mle")
  adj_mle_fit <- alogLik(mle_fit)
  summary(adj_mle_fit)

  # Another example from the eva::gpdFit documentation
  # A linear trend in the scale parameter
  set.seed(7)
  n <- 300
  x2 <- eva::rgpd(n, loc = 0, scale = 1 + 1:n / 200, shape = 0)
  covs <- as.data.frame(seq(1, n, 1))
```

```
  names(covs) <- c("Trend1")
  result1 <- eva::gpdFit(x2, threshold = 0, scalevars = covs,
                         scaleform = ~ Trend1)
  adj_result1 <- alogLik(result1)
  summary(adj_result1)

  # An example from the eva::gevrFit documentation
  set.seed(7)
  x1 <- eva::rgevr(500, 1, loc = 0.5, scale = 1, shape = 0.3)
  result1 <- eva::gevrFit(x1, method = "mle")
  adj_result1 <- alogLik(result1)
  summary(adj_result1)

  # Another example from the eva::gevrFit documentation
  # A linear trend in the location and scale parameter
  n <- 100
  r <- 10
  x2 <- eva::rgevr(n, r, loc = 100 + 1:n / 50,  scale = 1 + 1:n / 300,
                   shape = 0)
  covs <- as.data.frame(seq(1, n, 1))
  names(covs) <- c("Trend1")
  # Create some unrelated covariates
  covs$Trend2 <- rnorm(n)
  covs$Trend3 <- 30 * runif(n)
  result2 <- eva::gevrFit(data = x2, method = "mle", locvars = covs,
                          locform = ~ Trend1 + Trend2*Trend3,
                          scalevars = covs, scaleform = ~ Trend1)
  adj_result2 <- alogLik(result2)
  summary(adj_result2)
}
```

---

evd *Loglikelihood adjustment for evd fits*

Description

S3 `alogLik` method to perform loglikelihood adjustment for fitted extreme value model objects returned from the functions `fgev` and `fpot` in the evd package. If `x` is returned from `fgev` then the call must have used `prob = NULL`.

Usage

```
## S3 method for class 'evd'
alogLik(x, cluster = NULL, use_vcov = TRUE, ...)
```

Arguments

`x`	A fitted model object with certain associated S3 methods. See **Details**.

cluster	A vector or factor indicating from which cluster the respective log-likelihood contributions from loglik originate. The length of cluster must be consistent with the estfun method to be used in the estimation of the 'meat' V of the sandwich estimator of the covariance matrix of the parameters to be passed to adjust_loglik. In most cases, cluster must have length equal to the number of observations in data. The exception is the GP (only) model (binom = FALSE), where the cluster may either contain a value for each observation in the raw data, or for each threshold exceedance in the data. If cluster is not supplied (is NULL) then it is assumed that each observation forms its own cluster. See **Details** for further details.
use_vcov	A logical scalar. Should we use the vcov S3 method for x (if this exists) to estimate the Hessian of the independence loglikelihood to be passed as the argument H to adjust_loglik? Otherwise, H is estimated inside adjust_loglik using optimHess.
...	Further arguments to be passed to the functions in the sandwich package meat (if cluster = NULL), or meatCL (if cluster is not NULL).

Details

See alogLik for details.

Value

An object inheriting from class "chandwich". See adjust_loglik. class(x) is a vector of length 5. The first 3 components are c("lax", "chandwich", "evd"). The remaining 2 components depend on the model that was fitted. If fgev was used then these components are c("gev", "stat") if nsloc was NULL and c("gev", "nonstat") if nsloc was not NULL. If fpot was used then these components are c("pot", "gpd") if model was "gpd" and c("pot", "pp") if model was "pp".

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

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

Zeileis (2006) Object-Oriented Computation and Sandwich Estimators. *Journal of Statistical Software*, **16**, 1-16. doi:10.18637/jss.v016.i09

See Also

alogLik: loglikelihood adjustment for model fits.

Examples

```
# We need the evd package
got_evd <- requireNamespace("evd", quietly = TRUE)

if (got_evd) {
  library(evd)
```

```
  # An example from the evd::fgev documentation
  set.seed(3082019)
  uvdata <- evd::rgev(100, loc = 0.13, scale = 1.1, shape = 0.2)
  M1 <- evd::fgev(uvdata, nsloc = (-49:50)/100)
  adj_fgev <- alogLik(M1)
  summary(adj_fgev)

  # An example from Chandler and Bate (2007)
  owfit <- fgev(ow$temp, nsloc = ow$loc)
  adj_owfit <- alogLik(owfit, cluster = ow$year)
  summary(adj_owfit)

  # An example from the evd::fpot documentation
  set.seed(3082019)
  uvdata <- evd::rgpd(100, loc = 0, scale = 1.1, shape = 0.2)
  M1 <- fpot(uvdata, 1)
  adj_fpot <- alogLik(M1)
  summary(adj_fpot)
  # Fit using the pp model, rather than the gpd
  M1 <- fpot(uvdata, 1, model = "pp", npp = 365)
  adj_fpot <- alogLik(M1)
  summary(adj_fpot)
}
```

`evir` *Loglikelihood adjustment for evir fits*

Description

S3 `alogLik` method to perform loglikelihood adjustment for fitted extreme value model objects returned from the functions `gev`, `gpd` and `pot` in the evir package. If `x` was returned from `pot` then the model will need to be re-fitted using `pot_refit`.

Usage

```
## S3 method for class 'gev'
alogLik(x, cluster = NULL, use_vcov = TRUE, ...)

## S3 method for class 'gpd'
alogLik(x, cluster = NULL, use_vcov = TRUE, ...)

## S3 method for class 'potd'
alogLik(x, cluster = NULL, use_vcov = TRUE, ...)
```

Arguments

`x`	A fitted model object with certain associated S3 methods. See **Details**.

`cluster` A vector or factor indicating from which cluster the respective log-likelihood contributions from `loglik` originate. The length of `cluster` must be consistent with the `estfun` method to be used in the estimation of the 'meat' `V` of the sandwich estimator of the covariance matrix of the parameters to be passed to `adjust_loglik`. In most cases, `cluster` must have length equal to the number of observations in data. The exception is the GP (only) model (`binom = FALSE`), where the `cluster` may either contain a value for each observation in the raw data, or for each threshold exceedance in the data.

If `cluster` is not supplied (is `NULL`) then it is assumed that each observation forms its own cluster. See **Details** for further details.

`use_vcov` A logical scalar. Should we use the `vcov` S3 method for `x` (if this exists) to estimate the Hessian of the independence loglikelihood to be passed as the argument `H` to `adjust_loglik`? Otherwise, `H` is estimated inside `adjust_loglik` using `optimHess`.

`...` Further arguments to be passed to the functions in the sandwich package `meat` (if `cluster = NULL`), or `meatCL` (if `cluster` is not `NULL`).

Details

See `alogLik` for details.

If `pot` was used then `x` does not contain the raw data that `alogLik` needs. The model will need to be re-fitted using `pot_refit` and the user will be prompted to do this by an error message produced by `alogLik`.

Value

An object inheriting from class `"chandwich"`. See `adjust_loglik`. `class(x)` is a vector of length 5. The first 3 components are `c("lax", "chandwich", "evir")`. The remaining 2 components depend on the model that was fitted. If `gev` was used then these components are `c("gev", "stat")`. If `gpd` was used then these components are `c("gpd", "stat")`. If `pot_refit` was used then these components are `c("potd", "stat")`.

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

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

Zeileis (2006) Object-Oriented Computation and Sandwich Estimators. *Journal of Statistical Software*, **16**, 1-16. doi:10.18637/jss.v016.i09

See Also

`alogLik`: loglikelihood adjustment for model fits.

Examples

```
# We need the evir package
got_evir <- requireNamespace("evir", quietly = TRUE)
if (got_evir) {
  library(evir)
  # An example from the evir::gev documentation
  data(bmw)
  out <- gev(bmw, "month")
  adj_out <- alogLik(out)
  summary(adj_out)

  # An example from the evir::gpd documentation
  data(danish)
  out <- gpd(danish, 10)
  adj_out <- alogLik(out)
  summary(adj_out)

  # An example from the evir::pot documentation
  # We use lax::pot_refit() to return the input data
  out <- pot_refit(danish, 10)
  adj_out <- alogLik(out)
  summary(adj_out)
}
```

`extRemes` *Loglikelihood adjustment for extRemes fits*

Description

S3 `alogLik` method to perform loglikelihood adjustment for fitted extreme value model objects returned from the function `fevd` in the `extRemes` package. The model must have been fitted using maximum likelihood estimation.

Usage

```
## S3 method for class 'fevd'
alogLik(x, cluster = NULL, use_vcov = TRUE, ...)
```

Arguments

`x`	A fitted model object with certain associated S3 methods. See **Details**.
`cluster`	A vector or factor indicating from which cluster the respective log-likelihood contributions from `loglik` originate. The length of `cluster` must be consistent with the `estfun` method to be used in the estimation of the 'meat' `V` of the sandwich estimator of the covariance matrix of the parameters to be passed to `adjust_loglik`. In most cases, `cluster` must have length equal to the number of observations in data. The exception is the GP (only) model (`binom` = `FALSE`),

where the cluster may either contain a value for each observation in the raw data, or for each threshold exceedance in the data.

If cluster is not supplied (is NULL) then it is assumed that each observation forms its own cluster. See **Details** for further details.

use_vcov — A logical scalar. Should we use the vcov S3 method for x (if this exists) to estimate the Hessian of the independence loglikelihood to be passed as the argument H to adjust_loglik? Otherwise, H is estimated inside adjust_loglik using optimHess.

... — Further arguments to be passed to the functions in the sandwich package meat (if cluster = NULL), or meatCL (if cluster is not NULL).

Details

See alogLik for details.

Value

An object inheriting from class "chandwich". See adjust_loglik. class(x) is a vector of length 5. The first 3 components are c("lax", "chandwich", "extRemes"). The remaining 2 components depend on the model that was fitted. The 4th component is: "gev" if x$type = "GEV" or x$type = "Gumbel"; "gp" if x$type = "GP" or x$type = "Exponential"; "pp" if x$type = "PP". The 5th component is "stat" if is.fixedfevd = TRUE and "nonstat" if is.fixedfevd = FALSE.

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

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

Zeileis (2006) Object-Oriented Computation and Sandwich Estimators. *Journal of Statistical Software*, **16**, 1-16. doi:10.18637/jss.v016.i09

See Also

alogLik: loglikelihood adjustment for model fits.

Examples

```
# We need the extRemes and distillery packages
got_extRemes <- requireNamespace("extRemes", quietly = TRUE)
got_distillery <- requireNamespace("distillery", quietly = TRUE)

if (got_extRemes & got_distillery) {
  library(extRemes)
  library(distillery)
  # Examples from the extRemes::fevd documentation
  data(PORTw)

  # GEV
```

```
fit0 <- fevd(TMX1, PORTw, units = "deg C", use.phi = TRUE)
adj_fit0 <- alogLik(fit0)
summary(adj_fit0)

# GEV regression
fitPORTstdmax <- fevd(TMX1, PORTw, scale.fun = ~STDTMAX, use.phi = TRUE)
adj_fit1 <- alogLik(fitPORTstdmax)
summary(adj_fit1)
fitPORTstdmax2 <- fevd(TMX1, PORTw, location.fun = ~STDTMAX,
                       scale.fun = ~STDTMAX, use.phi = TRUE)
adj_fit2 <- alogLik(fitPORTstdmax2)
summary(adj_fit2)
anova(adj_fit0, adj_fit1)
anova(adj_fit1, adj_fit2)
anova(adj_fit0, adj_fit2)
anova(adj_fit0, adj_fit1, adj_fit2)

# Gumbel
fit0 <- fevd(TMX1, PORTw, type = "Gumbel", units = "deg C")
adj_fit0 <- alogLik(fit0)
summary(adj_fit0)

# GP
data(damage)
fit1 <- fevd(Dam, damage, threshold = 6, type = "GP",
             time.units = "2.05/year")
adj_fit1 <- alogLik(fit1)
summary(adj_fit1)

# Exponential
fit0 <- fevd(Dam, damage, threshold = 6, type="Exponential",
             time.units = "2.05/year")
adj_fit0 <- alogLik(fit0)
summary(adj_fit0)

# GP non-constant threshold
data(Fort)
fit <- fevd(Prec, Fort, threshold = 0.475,
            threshold.fun = ~I(-0.15 * cos(2 * pi * month / 12)),
            type = "GP")
adj_fit <- alogLik(fit)
summary(adj_fit)

# Exponential non-constant threshold
fit <- fevd(Prec, Fort, threshold = 0.475,
            threshold.fun = ~I(-0.15 * cos(2 * pi * month / 12)),
            type = "Exponential")
adj_fit <- alogLik(fit)
summary(adj_fit)

# PP model
fit <- fevd(Prec, Fort, threshold = 0.475, type = "PP", units = "inches")
adj_fit <- alogLik(fit)
```

```
  summary(adj_fit)

  # PP non-constant threshold
  fit <- fevd(Prec, Fort, threshold = 0.475,
              threshold.fun=~I(-0.15 * cos(2 * pi * month / 12)),
              type = "PP")
  adj_fit <- alogLik(fit)
  summary(adj_fit)
}
```

---

fExtremes *Loglikelihood adjustment for fExtremes fits*

Description

S3 alogLik method to perform loglikelihood adjustment for fitted extreme value model objects returned from the functions gevFit, gumbelFit and gpdFit in the fExtremes package. The model must have been fitted using maximum likelihood estimation.

Usage

```
## S3 method for class 'fGEVFIT'
alogLik(x, cluster = NULL, use_vcov = TRUE, ...)

## S3 method for class 'fGPDFIT'
alogLik(x, cluster = NULL, use_vcov = TRUE, ...)
```

Arguments

x	A fitted model object with certain associated S3 methods. See **Details**.
cluster	A vector or factor indicating from which cluster the respective log-likelihood contributions from loglik originate. The length of cluster must be consistent with the estfun method to be used in the estimation of the 'meat' V of the sandwich estimator of the covariance matrix of the parameters to be passed to adjust_loglik. In most cases, cluster must have length equal to the number of observations in data. The exception is the GP (only) model (binom = FALSE), where the cluster may either contain a value for each observation in the raw data, or for each threshold exceedance in the data. If cluster is not supplied (is NULL) then it is assumed that each observation forms its own cluster. See **Details** for further details.
use_vcov	A logical scalar. Should we use the vcov S3 method for x (if this exists) to estimate the Hessian of the independence loglikelihood to be passed as the argument H to adjust_loglik? Otherwise, H is estimated inside adjust_loglik using optimHess.
...	Further arguments to be passed to the functions in the sandwich package meat (if cluster = NULL), or meatCL (if cluster is not NULL).

Details

See `alogLik` for details.

Value

An object inheriting from class `"chandwich"`. See `adjust_loglik`. `class(x)` is a vector of length 5. The first 3 components are `c("lax", "chandwich", "fExtremes")`. The remaining 2 components depend on the model that was fitted. If `gevFit` or `gumbelFit` was used then these components are `c("gev", "stat")`. If `gpdFit` was used then these components are `c("gpd", "stat")`.

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

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

Zeileis (2006) Object-Oriented Computation and Sandwich Estimators. *Journal of Statistical Software*, **16**, 1-16. doi:10.18637/jss.v016.i09

See Also

`alogLik`: loglikelihood adjustment for model fits.

Examples

```
# We need the fExtremes package
got_fExtremes <- requireNamespace("fExtremes", quietly = TRUE)
if (got_fExtremes) {
  library(fExtremes)

  # GEV
  # An example from the fExtremes::gevFit documentation
  set.seed(4082019)
  x <- fExtremes::gevSim(model = list(xi=0.25, mu=0, beta=1), n = 1000)
  # Fit GEV distribution by maximum likelihood estimation
  fit <- fExtremes::gevFit(x)
  adj_fit <- alogLik(fit)
  summary(adj_fit)

  # GP
  # An example from the fExtremes::gpdFit documentation
  # Simulate GP data
  x <- fExtremes::gpdSim(model = list(xi = 0.25, mu = 0, beta = 1), n = 1000)
  # Fit GP distribution by maximum likelihood estimation
  fit <- fExtremes::gpdFit(x, u = min(x))
  adj_fit <- alogLik(fit)
  summary(adj_fit)
}
```

ismev *Loglikelihood adjustment for ismev fits*

Description

S3 alogLik method to perform loglikelihood adjustment for fitted extreme value model objects returned from the functions gev.fit, gpd.fit, pp.fit and rlarg.fit in the ismev package. If regression modelling is used then the model will need to be re-fitted, see ismev_refits.

Usage

```
## S3 method for class 'gev.fit'
alogLik(x, cluster = NULL, use_vcov = TRUE, ...)

## S3 method for class 'pp.fit'
alogLik(x, cluster = NULL, use_vcov = TRUE, ...)

## S3 method for class 'gpd.fit'
alogLik(
  x,
  cluster = NULL,
  use_vcov = TRUE,
  binom = FALSE,
  k,
  inc_cens = TRUE,
  ...
)

## S3 method for class 'rlarg.fit'
alogLik(x, cluster = NULL, use_vcov = TRUE, ...)
```

Arguments

x	A fitted model object with certain associated S3 methods. See **Details**.
cluster	A vector or factor indicating from which cluster the respective log-likelihood contributions from loglik originate. The length of cluster must be consistent with the estfun method to be used in the estimation of the 'meat' V of the sandwich estimator of the covariance matrix of the parameters to be passed to adjust_loglik. In most cases, cluster must have length equal to the number of observations in data. The exception is the GP (only) model (binom = FALSE), where the cluster may either contain a value for each observation in the raw data, or for each threshold exceedance in the data. If cluster is not supplied (is NULL) then it is assumed that each observation forms its own cluster. See **Details** for further details.

use_vcov — A logical scalar. Should we use the vcov S3 method for x (if this exists) to estimate the Hessian of the independence loglikelihood to be passed as the argument H to adjust_loglik? Otherwise, H is estimated inside adjust_loglik using optimHess.

... — Further arguments to be passed to the functions in the sandwich package meat (if cluster = NULL), or meatCL (if cluster is not NULL).

binom — A logical scalar. This option is only relevant to **GP models** and is only available in the **stationary** (no covariates) case. If binom = FALSE then loglikelihood adjustment is only performed using the GP model. If binom = TRUE then loglikelihood adjustment is also performed for inferences about the probability of threshold exceedance, using a Bernoulli model for the instances of threshold exceedance.

k — A non-negative integer scalar. This option is only relevant to **GP models** and is only available in the **stationary** (no covariates) case. If k is supplied then it is passed as the run parameter K to kgaps for making inferences about the extremal index θ using the K-gaps model of Suveges and Davison (2010).

inc_cens — A logical scalar. This argument is only relevant if k is supplied. Passed to kgaps to indicate whether or not to include censored inter-exceedance times, relating to the first and last observations.

Details

See alogLik for details.

If regression modelling is used then the ismev functions gev.fit, gpd.fit, pp.fit and rlarg.fit return residuals but alogLik needs the raw data. The model will need to be re-fitted, using one of the functions in ismev_refits, and the user will be prompted to do this by an error message produced by alogLik.

Value

An object inheriting from class "chandwich". See adjust_loglik.

class(x) is a vector of length 5. The first 3 components are c("lax", "chandwich", "ismev"). The remaining 2 components depend on the model that was fitted. The 4th component is: "gev" if gev.fit (or gev_refit) was used; "gpd" if gpd.fit (or gpd_refit) was used; "pp" pp.fit (or pp_refit) was used; "rlarg" rlarg.fit (or rlarg_refit) was used. The 5th component is "stat" if x$trans = FALSE and "nonstat" if x$trans = TRUE.

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

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

Zeileis (2006) Object-Oriented Computation and Sandwich Estimators. *Journal of Statistical Software*, **16**, 1-16. doi:10.18637/jss.v016.i09

See Also

alogLik: loglikelihood adjustment for model fits.

Examples

```
# We need the ismev package
got_ismev <- requireNamespace("ismev", quietly = TRUE)

if (got_ismev) {
  library(ismev)

  # GEV model -----

  # An example from the ismev::gev.fit documentation
  gev_fit <- gev.fit(revdbayes::portpirie, show = FALSE)
  adj_gev_fit <- alogLik(gev_fit)
  summary(adj_gev_fit)

  # An example from chapter 6 of Coles (2001)
  data(fremantle)
  xdat <- fremantle[, "SeaLevel"]
  # Set year 1897 to 1 for consistency with page 113 of Coles (2001)
  ydat <- cbind(fremantle[, "Year"] - 1896, fremantle[, "SOI"])
  gev_fit <- gev_refit(xdat, ydat, mul = 1:2, show = FALSE)
  adj_gev_fit <- alogLik(gev_fit)
  summary(adj_gev_fit)

  # An example from Chandler and Bate (2007)
  gev_fit <- gev_refit(ow$temp, ow, mul = 4, sigl = 4, shl = 4,
                       show = FALSE)
  adj_gev_fit <- alogLik(gev_fit, cluster = ow$year)
  summary(adj_gev_fit)
  # Get closer to the values reported in Table 2 of Chandler and Bate (2007)
  gev_fit <- gev_refit(ow$temp, ow, mul = 4, sigl = 4, shl = 4,
                       show = FALSE, method = "BFGS")
  # Call sandwich::meatCL() with cadjust = FALSE
  adj_gev_fit <- alogLik(gev_fit, cluster = ow$year, cadjust = FALSE)
  summary(adj_gev_fit)

  # GP model -----

  # An example from the ismev::gpd.fit documentation

  data(rain)
  rain_fit <- gpd.fit(rain, 10, show = FALSE)
  adj_rain_fit <- alogLik(rain_fit)
  summary(adj_rain_fit)
  # Continuing to the regression example on page 119 of Coles (2001)
  ydat <- as.matrix((1:length(rain)) / length(rain))
  reg_rain_fit <- gpd_refit(rain, 30, ydat = ydat, sigl = 1, siglink = exp,
                            show = FALSE)
  adj_reg_rain_fit <- alogLik(reg_rain_fit)
```

```
summary(adj_reg_rain_fit)

# Binomial-GP model -----

# Use Newlyn seas surges data from the exdex package
surges <- exdex::newlyn
u <- quantile(surges, probs = 0.9)
newlyn_fit <- gpd.fit(surges, u, show = FALSE)
# Create 5 clusters each corresponding approximately to 1 year of data
cluster <- rep(1:5, each = 579)[-1]
adj_newlyn_fit <- alogLik(newlyn_fit, cluster = cluster, binom = TRUE,
                          cadjust = FALSE)
summary(adj_newlyn_fit)
summary(attr(adj_newlyn_fit, "pu_aloglik"))

# Add inference about the extremal index theta, using K = 1
adj_newlyn_theta <- alogLik(newlyn_fit, cluster = cluster, binom = TRUE,
                            k = 1, cadjust = FALSE)
summary(attr(adj_newlyn_theta, "theta"))

# PP model -----

# An example from the ismev::pp.fit documentation
data(rain)
# Start from the mle to save time
init <- c(40.55755732, 8.99195409, 0.05088103)
muinit <- init[1]
siginit <- init[2]
shinit <- init[3]
rain_fit <- pp_refit(rain, 10, muinit = muinit, siginit = siginit,
                     shinit = shinit, show = FALSE)
adj_rain_fit <- alogLik(rain_fit)
summary(adj_rain_fit)

# An example from chapter 7 of Coles (2001).
# Code from demo ismev::wooster.temps
data(wooster)
x <- seq(along = wooster)
usin <- function(x, a, b, d) {
  return(a + b * sin(((x - d) * 2 * pi) / 365.25))
}
wu <- usin(x, -30, 25, -75)
ydat <- cbind(sin(2 * pi * x / 365.25), cos(2 * pi *x / 365.25))
# Start from the mle to save time
init <- c(-15.3454188, 9.6001844, 28.5493828, 0.5067104, 0.1023488,
          0.5129783, -0.3504231)
muinit <- init[1:3]
siginit <- init[4:6]
shinit <- init[7]
wooster.pp <- pp_refit(-wooster, threshold = wu, ydat = ydat, mul = 1:2,
                       sigl = 1:2, siglink = exp, method = "BFGS",
                       muinit = muinit, siginit = siginit, shinit = shinit,
                       show = FALSE)
```

```
  adj_pp_fit <- alogLik(wooster.pp)
  summary(adj_pp_fit)

  # r-largest order statistics model -----

  # An example based on the ismev::rlarg.fit() documentation
  vdata <- revdbayes::venice
  rfit <- rlarg.fit(vdata, muinit = 120.54, siginit = 12.78,
                    shinit = -0.1129, show = FALSE)
  adj_rfit <- alogLik(rfit)
  summary(adj_rfit)


  # Adapt this example to add a covariate
  set.seed(30102019)
  ydat <- matrix(runif(nrow(vdata)), nrow(vdata), 1)
  rfit2 <- rlarg_refit(vdata, ydat = ydat, mul = 1,
                       muinit = c(120.54, 0), siginit = 12.78,
                       shinit = -0.1129, show = FALSE)
  adj_rfit2 <- alogLik(rfit2)
  summary(adj_rfit2)

}
```

`ismev_refits` *Maximum-likelihood (Re-)Fitting using the ismev package*

Description

These are a slightly modified versions of the `gev.fit`, `gpd.fit`, `pp.fit` and `rlarg.fit` functions in the `ismev` package. The modification is to add to the returned object regression design matrices for the parameters of the model. That is, `xdat`, `ydat`, `mulink`, `siglink`, `shlink` and matrices `mumat`, `sigmat`, `shmat` for the location, scale and shape parameters `gev.fit`, `pp.fit` and `rlarg.fit`, and `xdat`, `ydat`, `siglink`, `shlink` and matrices `sigmat`, `shmat` for the scale and shape parameters for `gpd.fit`.

Usage

```
gev_refit(
  xdat,
  ydat = NULL,
  mul = NULL,
  sigl = NULL,
  shl = NULL,
  mulink = identity,
  siglink = identity,
  shlink = identity,
  muinit = NULL,
  siginit = NULL,
```

```
  shinit = NULL,
  show = TRUE,
  method = "Nelder-Mead",
  maxit = 10000,
  ...
)

gpd_refit(
  xdat,
  threshold,
  npy = 365,
  ydat = NULL,
  sigl = NULL,
  shl = NULL,
  siglink = identity,
  shlink = identity,
  siginit = NULL,
  shinit = NULL,
  show = TRUE,
  method = "Nelder-Mead",
  maxit = 10000,
  ...
)

pp_refit(
  xdat,
  threshold,
  npy = 365,
  ydat = NULL,
  mul = NULL,
  sigl = NULL,
  shl = NULL,
  mulink = identity,
  siglink = identity,
  shlink = identity,
  muinit = NULL,
  siginit = NULL,
  shinit = NULL,
  show = TRUE,
  method = "Nelder-Mead",
  maxit = 10000,
  ...
)

rlarg_refit(
  xdat,
  r = dim(xdat)[2],
  ydat = NULL,
```

```
  mul = NULL,
  sigl = NULL,
  shl = NULL,
  mulink = identity,
  siglink = identity,
  shlink = identity,
  muinit = NULL,
  siginit = NULL,
  shinit = NULL,
  show = TRUE,
  method = "Nelder-Mead",
  maxit = 10000,
  ...
)
```

Arguments

xdat	A numeric vector of data to be fitted.
ydat	A matrix of covariates for generalized linear modelling of the parameters (or NULL (the default) for stationary fitting). The number of rows should be the same as the length of xdat.
mul, sigl, shl	Numeric vectors of integers, giving the columns of ydat that contain covariates for generalized linear modelling of the location, scale and shape parameters repectively (or NULL (the default) if the corresponding parameter is stationary).
mulink, siglink, shlink	Inverse link functions for generalized linear modelling of the location, scale and shape parameters repectively.
muinit, siginit, shinit	numeric of length equal to total number of parameters used to model the location, scale or shape parameter(s), resp. See Details section for default (NULL) initial values.
show	Logical; if TRUE (the default), print details of the fit.
method	The optimization method (see optim for details).
maxit	The maximum number of iterations.
...	Other control parameters for the optimization. These are passed to components of the control argument of optim.
threshold	The threshold; a single number or a numeric vector of the same length as xdat.
npy	The number of observations per year/block.
r	The largest r order statistics are used for the fitted model.

References

Heffernan, J. E. and Stephenson, A. G. (2018). ismev: An Introduction to Statistical Modeling of Extreme Values. R package version 1.42. https://CRAN.R-project.org/package=ismev.

Examples

```
# We need the ismev package
got_ismev <- requireNamespace("ismev", quietly = TRUE)
if (got_ismev) {
  library(ismev)
  fit1 <- gev.fit(revdbayes::portpirie, show = FALSE)
  ls(fit1)
  fit2 <- gev_refit(revdbayes::portpirie, show = FALSE)
  ls(fit2)

  data(rain)
  fit1 <- gpd.fit(rain, 10)
  ls(fit1)
  fit2 <- gpd_refit(rain, 10)
  ls(fit2)

  fit1 <- pp.fit(rain, 10, show = FALSE)
  ls(fit1)
  fit2 <- pp_refit(rain, 10, show = FALSE)
  ls(fit2)

  data(venice)
  fit1 <- rlarg.fit(venice[, -1], muinit = 120.54, siginit = 12.78,
                    shinit = -0.1129, show = FALSE)
  ls(fit1)
  fit2 <- rlarg_refit(venice[, -1], muinit = 120.54, siginit = 12.78,
                    shinit = -0.1129, show = FALSE)
  ls(fit2)
}
```

`logLik.logLikVec` *Sum loglikelihood contributions from individual observations*

Description

S3 logLik method for logLikVec objects.

Usage

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

Arguments

`object`	An object of class `"logLikVec"` return from a `logLikVec` method.
`...`	Further arguments.

Details

See `alogLik`: loglikelihood adjustment for model fits.

Value

An object of class "logLik". This is the value of the loglikelihood, with attributes "df" (degrees of freedom) giving the number of free parameters, and "nobs" giving the number of observations.

Examples

See the example in bernoulli.

---

logLikVec *Evaluate loglikelihood contributions from specific observations*

Description

Generic function for calculating loglikelihood contributions from individual observations for a fitted model.

Usage

```
logLikVec(object, ...)
```

Arguments

object	A fitted model object.
...	Further arguments.

Details

See alogLik: loglikelihood adjustment for model fits.

Value

An object of class "logLikVec", a vector containing individual loglikelihood contributions.

Examples

See the example in bernoulli.

mev *Loglikelihood adjustment for mev fits*

Description

S3 `alogLik` method to perform loglikelihood adjustment for fitted extreme value model objects returned from the functions `fit.gev`, `fit.gpd`, and `fit.pp` and `fit.rlarg` in the mev package.

Usage

```
## S3 method for class 'mev_gev'
alogLik(x, cluster = NULL, use_vcov = TRUE, ...)

## S3 method for class 'mev_pp'
alogLik(x, cluster = NULL, use_vcov = TRUE, ...)

## S3 method for class 'mev_gpd'
alogLik(x, cluster = NULL, use_vcov = TRUE, ...)

## S3 method for class 'mev_egp'
alogLik(x, cluster = NULL, use_vcov = TRUE, ...)

## S3 method for class 'mev_rlarg'
alogLik(x, cluster = NULL, use_vcov = TRUE, ...)
```

Arguments

`x`	A fitted model object with certain associated S3 methods. See **Details**.
`cluster`	A vector or factor indicating from which cluster the respective log-likelihood contributions from `loglik` originate. The length of `cluster` must be consistent with the `estfun` method to be used in the estimation of the 'meat' `V` of the sandwich estimator of the covariance matrix of the parameters to be passed to `adjust_loglik`. In most cases, `cluster` must have length equal to the number of observations in data. The exception is the GP (only) model (`binom = FALSE`), where the `cluster` may either contain a value for each observation in the raw data, or for each threshold exceedance in the data. If `cluster` is not supplied (is `NULL`) then it is assumed that each observation forms its own cluster. See **Details** for further details.
`use_vcov`	A logical scalar. Should we use the `vcov` S3 method for `x` (if this exists) to estimate the Hessian of the independence loglikelihood to be passed as the argument `H` to `adjust_loglik`? Otherwise, `H` is estimated inside `adjust_loglik` using `optimHess`.
`...`	Further arguments to be passed to the functions in the sandwich package `meat` (if `cluster = NULL`), or `meatCL` (if `cluster` is not `NULL`).

Details

See alogLik for details.

If x was returned from fit.pp then the data xdat supplied to fit.pp must contain *all* the data, both threshold exceedances and non-exceedances.

Value

An object inheriting from class "chandwich". See adjust_loglik. class(x) is a vector of length 5. The first 3 components are c("lax", "chandwich", "mev"). The 4th component depends on which model was fitted. "gev" if fit.gev was used; "gpd" if fit.gpd was used; "pp" fit.pp was used; "egp" fit.egp was used; "rlarg" fit.rlarg was used; The 5th component is "stat" (for stationary).

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

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

Zeileis (2006) Object-Oriented Computation and Sandwich Estimators. *Journal of Statistical Software*, **16**, 1-16. doi:10.18637/jss.v016.i09

See Also

alogLik: loglikelihood adjustment for model fits.

Examples

```
# We need the mev package
got_mev <- requireNamespace("mev", quietly = TRUE)

if (got_mev) {
  library(mev)
  # An example from the mev::gev.fit documentation
  gev_mev <- fit.gev(revdbayes::portpirie)
  adj_gev_mev <- alogLik(gev_mev)
  summary(adj_gev_mev)

  # Use simulated data
  set.seed(1112019)
  x <- revdbayes::rgp(365 * 10, loc = 0, scale = 1, shape = 0.1)
  pfit <- fit.pp(x, threshold = 1, npp = 365)
  adj_pfit <- alogLik(pfit)
  summary(adj_pfit)

  # An example from the mev::fit.gpd documentation
  gpd_mev <- fit.gpd(eskrain, threshold = 35, method = 'Grimshaw')
  adj_gpd_mev <- alogLik(gpd_mev)
  summary(adj_gpd_mev)
```

```
  # An example from the mev::fit.egp documentation
  # (model = "egp1" and model = "egp3" also work)
  xdat <- evd::rgpd(n = 100, loc = 0, scale = 1, shape = 0.5)
  fitted <- fit.egp(xdat = xdat, thresh = 1, model = "egp2", show = FALSE)
  adj_fitted <- alogLik(fitted)
  summary(adj_fitted)

  # An example from the mev::fit.rlarg documentation
  set.seed(31102019)
  xdat <- rrlarg(n = 10, loc = 0, scale = 1, shape = 0.1, r = 4)
  fitr <- fit.rlarg(xdat)
  adj_fitr <- alogLik(fitr)
  summary(adj_fitr)
}
```

---

ow *Oxford and Worthing annual maximum temperatures*

Description

Annual maximum temperatures at Oxford and Worthing (England), for the period 1901 to 1980.

Usage

```
ow
```

Format

A dataframe with 80 rows and 4 columns.

- Column 1, `temp`: annual maximum temperatures in degrees Fahrenheit.
- Column 2, `year`: year in which the maximum was recorded.
- Column 3, `name`: name of location, "oxford" or "worthing"
- Column 4, `loc`: location: 1 for "oxford", -1 for "worthing"

Source

Tabony, R. C. (1983) Extreme value analysis in meteorology. *The Meteorological Magazine*, **112**, 77-98.

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

plot.retlev *Plot diagnostics for a retlev object*

Description

plot method for an objects of class c("retlev", "lax").

Usage

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

Arguments

x	an object of class c("retlev", "lax"), a result of a call to return_level, using prof = TRUE.
y	Not used.
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	An integer. Passed to signif to round the values in the legend.
plot	A logical scalar. If TRUE then the plot is produced. Otherwise, it is not, but the MLE and confidence limits are returned.
...	Further arguments to be passed to plot.

Details

Plots the profile loglikelihood for a return level, provided that x returned by a call to return_level using prof = TRUE. Horizontal lines indicate the values of the maximised loglikelihood 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.

Value

A numeric vector of length 3 containing the lower 100level% confidence limit, the MLE and the upper 100level% confidence limit.

Examples

See the examples in return_level.

See Also

return_level to perform inferences about return levels.

POT *Loglikelihood adjustment for POT fits*

Description

S3 alogLik method to perform loglikelihood adjustment for fitted extreme value model objects returned from fitGPD function in the POT package. The model must have been fitted using maximum likelihood estimation.

Usage

```
## S3 method for class 'uvpot'
alogLik(x, cluster = NULL, use_vcov = TRUE, ...)
```

Arguments

x	A fitted model object with certain associated S3 methods. See **Details**.
cluster	A vector or factor indicating from which cluster the respective log-likelihood contributions from loglik originate. The length of cluster must be consistent with the estfun method to be used in the estimation of the 'meat' V of the sandwich estimator of the covariance matrix of the parameters to be passed to adjust_loglik. In most cases, cluster must have length equal to the number of observations in data. The exception is the GP (only) model (binom = FALSE), where the cluster may either contain a value for each observation in the raw data, or for each threshold exceedance in the data. If cluster is not supplied (is NULL) then it is assumed that each observation forms its own cluster. See **Details** for further details.
use_vcov	A logical scalar. Should we use the vcov S3 method for x (if this exists) to estimate the Hessian of the independence loglikelihood to be passed as the argument H to adjust_loglik? Otherwise, H is estimated inside adjust_loglik using optimHess.
...	Further arguments to be passed to the functions in the sandwich package meat (if cluster = NULL), or meatCL (if cluster is not NULL).

Details

See alogLik for details.

Value

An object inheriting from class "chandwich". See adjust_loglik.

class(x) is c("lax", "chandwich", "POT", "pot", "gpd").

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

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

Zeileis (2006) Object-Oriented Computation and Sandwich Estimators. *Journal of Statistical Software*, **16**, 1-16. doi:10.18637/jss.v016.i09

See Also

`alogLik`: loglikelihood adjustment for model fits.

Examples

```
# We need the POT package
got_POT <- requireNamespace("POT", quietly = TRUE)

if (got_POT) {
  library(POT)
  # An example from the POT::fitgpd documentation.
  set.seed(4082019)
  x <- POT::rgpd(200, 1, 2, 0.25)
  fit <- fitgpd(x, 1, "mle")
  adj_fit <- alogLik(fit)
}
```

`pot_refit` *Fits a Poisson point process to the data, an approach sometimes known as peaks over thresholds (POT), and returns an object of class "potd".*

Description

This is a slightly modified versions of the `pot` function in the `evir` package. The main modification is to add to the returned object the argument `data` supplied by the user. This is added to the returned (list) object with the name `input_data`.

Usage

```
pot_refit(data, threshold = NA, nextremes = NA, run = NA, picture = TRUE, ...)
```

Arguments

`data`	numeric vector of data, which may have a times attribute containing (in an object of class `"POSIXct"`, or an object that can be converted to that class; see `as.POSIXct`) the times/dates of each observation. If no times attribute exists, the data are assumed to be equally spaced.
`threshold`	a threshold value (either this or `nextremes` must be given but not both).

nextremes	the number of upper extremes to be used (either this or threshold must be given but not both).
run	if the data are to be declustered the run length parameter for the runs method (see decluster) should be entered here.
picture	whether or not a picture should be drawn if declustering is performed.
...	arguments passed to optim.

References

Bernhard Pfaff and Alexander McNeil (2018). evir: Extreme Values in R. R package version 1.7-4. https://CRAN.R-project.org/package=evir.

Examples

```
# We need the evir package
got_evir <- requireNamespace("evir", quietly = TRUE)
if (got_evir) {
  library(evir)
  data(danish)
  out <- pot(danish, 10)
  ls(out)
  out <- pot_refit(danish, 10)
  ls(out)
}
```

---

print.retlev *Print method for retlev object*

Description

print method for an objects of class c("retlev", "lax").

Usage

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

Arguments

x	an object of class c("retlev", "lax"), a result of a call to return_level.
digits	The argument digits to print.default.
...	Additional arguments. None are used in this function.

Details

Prints the call to return_level and the estimates and 100x$level% confidence limits for the x$m-year return level.

Value

The argument x, invisibly, as for all print methods.

Examples

See the examples in return_level.

See Also

return_level.

---

print.summary.retlev *Print method for objects of class* "summary.retlev"

Description

print method for an object x of class "summary.retlev".

Usage

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

Arguments

x	An object of class "summary.retlev", a result of a call to summary.retlev.
...	Additional arguments passed on to print.default.

Details

Prints the call and the numeric matrix x$matrix returned from summary.retlev.

Value

The argument x, invisibly, as for all print methods.

Examples

See the examples in return_level.

See Also

return_level to perform inferences about return levels.

`return_level` *Return Level Inferences for Stationary Extreme Value Models*

Description

Calculates point estimates and confidence intervals for `m`-year return levels for **stationary** extreme value fitted model objects returned from `alogLik`. 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) loglikelihood.

Usage

```
return_level(
  x,
  m = 100,
  level = 0.95,
  npy = 1,
  prof = TRUE,
  inc = NULL,
  type = c("vertical", "cholesky", "spectral", "none")
)
```

Arguments

`x`	An object inheriting from class `"lax"` returned from `alogLik`.
`m`	A numeric scalar. The return period, in years.
`level`	A numeric scalar in (0, 1). The confidence level required for the confidence interval for the `m`-year return level.
`npy`	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 loglikelihood?
`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 `adjust_loglik`, that is, the type of adjustment made to the independence loglikelihood function in creating an adjusted loglikelihood function. See **Details** and **Value** in `adjust_loglik`.

Details

At present `return_level` only supports GEV models.

Care must be taken in specifying the input value of `npy`.

- **GEV models**: it is common to have one observation per year, either because the data are annual maxima or because for each year only the maximum value over a particular season is extracted from the raw data. In this case, `npy = 1`, which is the default. If instead we extract the maximum values over the first and second halves of each year then `npy = 2`.
- **Binomial-GP models**: `npy` 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 `npy` should be set equal to the mean number of observations per year.

Supplying `npy` for binomial-GP models. The value of `npy` (or an equivalent, perhaps differently named, quantity) may have been set in the call to fit a GP model. For example, the `gpd.fit()` function in the `ismev` package has a `npy` argument and the value of `npy` is stored in the fitted model object. If `npy` is supplied by the user in the call to `return_level` 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 loglikelihood reaches the critical value `logLik(x) - 0.5 * stats::qchisq(level, 1)`. This is achieved by calculating the profile loglikelihood for a sequence of values of this return level as governed by `inc`. Once the profile loglikelihood 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 `"retlev", "lax"` 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 loglikelihood; the critical value and a matrix with return levels in the first column (`ret_levs`) and the corresponding values of the (adjusted) profile loglikelihood (`prof_loglik`).
`m`, `level`	The input values of `m` and `level`.
`call`	The call to `return_level`.

References

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

See Also

plot.retlev for plotting the profile loglikelihood for a return level.

Examples

```
# GEV model -----

got_evd <- requireNamespace("evd", quietly = TRUE)

if (got_evd) {
  library(evd)
  # An example from the evd::fgev documentation
  set.seed(4082019)
  uvdata <- evd::rgev(100, loc = 0.13, scale = 1.1, shape = 0.2)
  M1 <- fgev(uvdata)
  adj_fgev <- alogLik(M1)
  # Large inc set here for speed, sacrificing accuracy
  rl <- return_level(adj_fgev, inc = 0.5)
  summary(rl)
  rl
  plot(rl)
}

got_ismev <- requireNamespace("ismev", quietly = TRUE)

if (got_ismev) {
  library(ismev)
  # An example from the ismev::gev.fit documentation
  gev_fit <- gev.fit(revdbayes::portpirie, show = FALSE)
  adj_gev_fit <- alogLik(gev_fit)
  # Large inc set here for speed, sacrificing accuracy
  rl <- return_level(adj_gev_fit, inc = 0.05)
  summary(rl)
  rl
  plot(rl)
}

# Binomial-GP model -----

if (got_ismev) {
  library(ismev)
  data(rain)
  # An example from the ismev::gpd.fit documentation
  rain_fit <- gpd.fit(rain, 10, show = FALSE)
  adj_rain_fit <- alogLik(rain_fit, binom = TRUE)
  # Large inc set here for speed, sacrificing accuracy
  rl <- return_level(adj_rain_fit, inc = 2.5)
  summary(rl)
  rl
  plot(rl)
}
```

```
if (got_ismev) {
  # Use Newlyn seas surges data from the exdex package
  surges <- exdex::newlyn
  u <- quantile(surges, probs = 0.9)
  newlyn_fit <- gpd.fit(surges, u, show = FALSE)
  # Create 5 clusters each corresponding approximately to 1 year of data
  cluster <- rep(1:5, each = 579)[-1]
  adj_newlyn_fit <- alogLik(newlyn_fit, cluster = cluster, binom = TRUE,
                            cadjust = FALSE)
  rl <- return_level(adj_newlyn_fit, inc = 0.02)
  rl

  # Add inference about the extremal index theta, using K = 1
  adj_newlyn_theta <- alogLik(newlyn_fit, cluster = cluster, binom = TRUE,
                              k = 1, cadjust = FALSE)
  rl <- return_level(adj_newlyn_theta, inc = 0.02)
  rl
}
```

`summary.retlev` *Summary method for a* `"retlev"` *object*

Description

`summary` method for an objects of class `c("retlev", "lax")`.

Usage

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

Arguments

`object`	an object of class `c("retlev", "lax")`, a result of a call to `return_level`.
`digits`	An integer. Used 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).
`...`	Additional arguments. None are used in this function.

Value

Returns a list containing the list element `object$call` and a numeric matrix `matrix` containing the MLE and estimated SE of the return level.

Examples

See the examples in `return_level`.

See Also

return_level.

---

texmex *Loglikelihood adjustment of texmex fits*

Description

S3 alogLik method to perform loglikelihood adjustment of fitted extreme value model objects returned from the evm function in the texmex package. The model must have been fitted using maximum likelihood estimation.

Usage

```
## S3 method for class 'evmOpt'
alogLik(x, cluster = NULL, use_vcov = TRUE, ...)
```

Arguments

x	A fitted model object with certain associated S3 methods. See **Details**.
cluster	A vector or factor indicating from which cluster the respective log-likelihood contributions from loglik originate. The length of cluster must be consistent with the estfun method to be used in the estimation of the 'meat' V of the sandwich estimator of the covariance matrix of the parameters to be passed to adjust_loglik. In most cases, cluster must have length equal to the number of observations in data. The exception is the GP (only) model (binom = FALSE), where the cluster may either contain a value for each observation in the raw data, or for each threshold exceedance in the data. If cluster is not supplied (is NULL) then it is assumed that each observation forms its own cluster. See **Details** for further details.
use_vcov	A logical scalar. Should we use the vcov S3 method for x (if this exists) to estimate the Hessian of the independence loglikelihood to be passed as the argument H to adjust_loglik? Otherwise, H is estimated inside adjust_loglik using optimHess.
...	Further arguments to be passed to the functions in the sandwich package meat (if cluster = NULL), or meatCL (if cluster is not NULL).

Details

See alogLik for details.

Value

An object inheriting from class "chandwich". See adjust_loglik. class(x) is a vector of length 5. The first 3 components are c("lax", "chandwich", "texmex"). The remaining 2 components depend on the model that was fitted. The 4th component is: "gev" if x$family$name = "GEV"; "gpd" if x$family$name = "GPD"; "egp3" if x$family$name = "EGP3". The 5th component is "stat" if there are no covariates in the mode and "nonstat" otherwise.

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

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

Zeileis (2006) Object-Oriented Computation and Sandwich Estimators. *Journal of Statistical Software*, **16**, 1-16. doi:10.18637/jss.v016.i09

See Also

`alogLik`: loglikelihood adjustment for model fits.

Examples

```
## Not run:
# Not run to avoid a CRAN check error inherited from the texmex package
# We need the texmex package, and ismev for the fremantle dataset
got_texmex <- requireNamespace("texmex", quietly = TRUE)
got_ismev <- requireNamespace("ismev", quietly = TRUE)
if (got_texmex) {
  library(texmex)
  # Examples from the texmex::evm documentation

  # GEV
  mod <- evm(SeaLevel, data = texmex::portpirie, family = gev)
  adj_mod <- alogLik(mod)
  summary(adj_mod)

  # GP
  mod <- evm(rain, th = 30)
  adj_mod <- alogLik(mod)
  summary(adj_mod)
  mod <- evm(rain, th = 30, cov = "sandwich")
  mod$se
  vcov(adj_mod)
  vcov(mod)

  # EGP3
  mod <- evm(rain, th = 30, family = egp3)
  adj_mod <- alogLik(mod)
  summary(adj_mod)

  # GP regression
  # An example from page 119 of Coles (2001)
  n_rain <- length(rain)
  rain_df <- data.frame(rain = rain, time = 1:n_rain / n_rain)
  evm_fit <- evm(y = rain, data = rain_df, family = gpd, th = 30,
                 phi = ~ time)
  adj_evm_fit <- alogLik(evm_fit)
  summary(adj_evm_fit)
  evm_fit <- evm(y = rain, data = rain_df, family = gpd, th = 30,
```

```
                phi = ~ time, cov = "sandwich")
  evm_fit$se
  vcov(adj_evm_fit)
  vcov(evm_fit)

  # GEV regression
  # An example from page 113 of Coles (2001)
  if (got_ismev) {
    library(ismev)
    data(fremantle)
    new_fremantle <- fremantle
    # Set year 1897 to 1 for consistency with page 113 of Coles (2001)
    new_fremantle[, "Year"] <- new_fremantle[, "Year"] - 1896
    evm_fit <- evm(y = SeaLevel, data = new_fremantle, family = gev,
                   mu = ~ Year + SOI)
    adj_evm_fit <- alogLik(evm_fit)
    summary(adj_evm_fit)
  }

  # An example from Chandler and Bate (2007)
  # Note: evm uses phi = log(sigma)
  evm_fit <- evm(temp, ow, gev, mu = ~ loc, phi = ~ loc, xi = ~loc)
  adj_evm_fit <- alogLik(evm_fit, cluster = ow$year, cadjust = FALSE)
  summary(adj_evm_fit)
}

## End(Not run)
```

Index