Package 'chandwich'

May 8, 2026

Title Chandler-Bate Sandwich Loglikelihood Adjustment

Version 1.1.6

Date 2023-08-25

Description Performs adjustments of a user-supplied independence loglikelihood function using 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. Functions for profiling the adjusted loglikelihoods are also provided, as are functions for calculating and plotting confidence intervals, for single model parameters, and confidence regions, for pairs of model parameters. Nested models can be compared using an adjusted likelihood ratio test.

Imports graphics, methods, numDeriv, stats, utils

License GPL (>= 2)

LazyData TRUE

Encoding UTF-8

Depends R (>= 3.3.0)

RoxygenNote 7.2.3

Suggests knitr, rmarkdown, sandwich, testthat

VignetteBuilder knitr

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

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

Config/testthat/edition 3

NeedsCompilation no

Author Paul J. Northrop [aut, cre, cph], Richard E. Chandler [aut, cph]

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

Repository CRAN

Date/Publication 2023-08-25 21:40:02 UTC

Contents

`chandwich-package` *chandwich: Chandler-Bate Sandwich Loglikelihood Adjustment*

Description

Performs adjustments of an independence loglikelihood using 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. Functions for profiling the adjusted loglikelihoods are also provided, as are functions for calculating and plotting confidence intervals, for single model parameters, and confidence regions, for pairs of model parameters.

Details

The main function in the chandwich package is `adjust_loglik`. It finds the maximum likelihood estimate (MLE) of model parameters based on an independence loglikelihood in which cluster dependence in the data is ignored. The independence loglikelihood is adjusted in a way that ensures that the Hessian of the adjusted loglikelihood coincides with a robust sandwich estimate of the parameter covariance at the MLE. Three adjustments are available: one in which the independence loglikelihood itself is scaled (vertical scaling) and two others where the scaling is in the parameter vector (horizontal scaling).

See Chandler and Bate (2007) for full details and `vignette("chandwich-vignette", package = "chandwich")` for an overview of the package.

Author(s)

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

Authors:

- Richard E. Chandler [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

See Also

`adjust_loglik` to adjust a user-supplied loglikelihood.

`compare_models` to compare nested models using an adjusted loglikelihood ratio test. See also the S3 method `anova.chandwich`.

`conf_intervals` to calculate confidence intervals for individual model parameters. See also the S3 method `confint.chandwich`.

`conf_region` to calculate a confidence region for a pair of model parameters.

`adjust_loglik` *Loglikelihood adjustment using the sandwich estimator*

Description

Performs adjustments of a user-supplied independence loglikelihood for the presence of cluster dependence, following Chandler and Bate (2007). The user provides a function that returns a vector of observation-specific loglikelihood contributions and a vector that indicates cluster membership. The loglikelihood of a sub-model can be adjusted by fixing a set of parameters at particular values.

Usage

```
adjust_loglik(
  loglik = NULL,
  ...,
  cluster = NULL,
  p = NULL,
  init = NULL,
  par_names = NULL,
  fixed_pars = NULL,
  fixed_at = 0,
  name = NULL,
  larger = NULL,
  alg_deriv = NULL,
  alg_hess = NULL,
  mle = NULL,
  H = NULL,
  V = NULL
)
```

Arguments

`loglik`	A named function. Returns a vector of the loglikelihood contributions of individual observations. The first argument must be the vector of model parameter(s). If any of the model parameters are out-of-bounds then `loglik` should return either `-Inf` or a vector with at least one element equal to `-Inf`. The number of parameters in the **full** model must be specified using (at least) one of the arguments `p`, `init` or `par_names`.
`...`	Further arguments to be passed either to `loglik` (and to `alg_deriv` and `alg_hess` if these are supplied) or to `optim`. The latter may include `gr`, `method`, `lower`, `upper` or `control`. In the call to `optim`, `hessian = TRUE` will be used regardless of any value supplied. The function `loglik` must *not* have arguments with names that match any of these arguments to `optim`.
`cluster`	A vector or factor indicating from which cluster the respective loglikelihood contributions from `loglik` originate. Must have the same length as the vector returned by `loglik`. If `cluster` is not supplied then it is set inside `adjust_loglik` under the assumption that each observation forms its own cluster.
`p`	A numeric scalar. The dimension of the **full** parameter vector, i.e. the number of parameters in the full model. Must be consistent with the lengths of `init` and `par_names`, if these are also supplied.
`init`	A numeric vector of initial values. Must have length equal to the number of parameters in the **full** model. If `init` is supplied then `p` is set to `length(init)`, provided that this is consistent with the the value given by `p` or implied by `length(par_names)`. If `fixed_pars` is not `NULL` then `init[-fixed_pars]` is used in the search for the MLE. If `init` is not supplied then `rep(0.1, p)` is used.
`par_names`	A character vector. Names of the `p` parameters in the **full** model. Must be consistent with the lengths of `init` and `p`, if these are also supplied.

`fixed_pars` A vector specifying which parameters are to be restricted to be equal to the value(s) in `fixed_at`. Can be either a numeric vector, specifying indices of the components of the **full** parameter vector, or a character vector of parameter names, which must be a subset of those supplied in `par_names` or stored in the object `larger`.

`fixed_at` A numeric vector of length 1 or `length(fixed_pars)`. If `length(fixed_at) = 1` then the components `fixed_pars` of the parameter vector are all fixed at `fixed_at`. If `length(fixed_at) = length(fixed_pars)` then the component `fixed_pars[i]` is fixed at `fixed_at[i]` for each `i`.

`name` A character scalar. A name for the model that gives rise to `loglik`. If this is not supplied then the name in `larger` is used, if this has been supplied, and the name of the function `loglik` otherwise.

`larger` Only relevant if `fixed_pars` is not `NULL`. If `larger` is supplied but `fixed_pars` is not then an error will result. if `larger` is supplied then information about the model in `larger`, e.g. about `p` and `par_names` will override any attempt to set these arguments in the call to `adjust_loglik`.

An object of class `"chandwich"` returned by `adjust_loglik`, corresponding to a model in which the smaller model implied by `fixed_pars` is nested. If `larger` is supplied then all the arguments to `adjust_loglik` apart from `fixed_pars` and `fixed_at` are extracted from `larger`. If `init` is not supplied in the current call to `adjust_loglik` then `init` is set to `attr(larger, "MLE")`, with the elements in `fixed_pars` set to `fixed_at`.

`alg_deriv` A function with the vector of model parameter(s) as its first argument. Returns a `length(cluster)` by `p` numeric matrix. Column i contains the derivatives of each of the loglikelihood contributions in `loglik` with respect to model parameter i.

`alg_hess` A function with the vector of model parameter(s) as its first argument. Returns a `p` by `p` numeric matrix equal to the Hessian of `loglik`, i.e. the matrix of second derivatives of the function `loglik`.

Supplying both `V` and `alg_deriv` or both `H` and `alg_hess` will produce an error.

`mle` A numeric vector. Can only be used if `fixed_pars = NULL`. Provides the maximum likelihood estimate of the model parameters, that is, the value of the parameter vector at which the independence loglikelihood `loglik` is maximized. Must have length equal to the number of parameters in the **full** model. If `mle` is supplied then `p` is set to `length(mle)`, provided that this is consistent with the the value given by `p` or implied by `length(par_names)`. If `mle` is supplied then it overrides `init`.

`H, V` p by p numeric matrices. Only used if `mle` is supplied. Provide estimates of the Hessian of the independence loglikelihood (H) and the variance of the vector of cluster-specific contributions to the score vector (first derivatives with respect to the parameters) of the independence loglikelihood, each evaluated at the MLE `mle`. See the *Introducing chandwich* vignette and/or Chandler and Bate (2007).

Supplying both `V` and `alg_deriv` or both `H` and `alg_hess` will produce an error.

Details

Three adjustments to the independence loglikelihood described in Chandler and Bate (2007) are available. The vertical adjustment is described in Section 6 and two horizontal adjustments are described in Sections 3.2 to 3.4. See the descriptions of `type` and, for the horizontal adjustments, the descriptions of `C_cholesky` and `C_spectral`, in **Value**.

The adjustments involve first and second derivatives of the loglikelihood with respect to the model parameters. These are estimated using `jacobian` and `optimHess` unless `alg_deriv` and/or `alg_hess` are supplied.

Value

A function of class `"chandwich"` to evaluate an adjusted loglikelihood, or the independence loglikelihood, at one or more sets of model parameters, with arguments

`x`	A numeric vector or matrix giving values of the `p_current` (see below) parameters in the model to which the returned adjusted loglikelihood applies. If `p_current = 1` this may be a numeric vector or a matrix with 1 column. If `p_current > 1` this may be a numeric vector of length `p` (one set of model parameters) or a numeric matrix with `p` columns (`nrow(x)` sets of model parameters), one set in each row of `x`.
`type`	A character scalar. The type of adjustment to use. One of `"vertical"`, `"cholesky"`, `"spectral"` or `"none"`.

The latter results in the evaluation of the (unadjusted) independence loglikelihood. The function has (additional) attributes

`p_full`, `p_current`	The number of parameters in the full model and current models, respectively.
`free_pars`	A numeric vector giving the indices of the free parameters in the current model, with names inferred from `par_names` if this was supplied.
`MLE`, `res_MLE`	Numeric vectors, with names inferred from `par_names` if this was supplied. Maximum likelihood estimates of free parameters under the current model (`mle`) and all parameters in the full model, including any parameters with fixed values (`res_MLE`).
`SE`, `adjSE`	The unadjusted and adjusted estimated standard errors, of the free parameters, respectively.
`VC`, `adjVC`	The unadjusted and adjusted estimated variance-covariance matrix of the free parameters, respectively.
`HI`, `HA`	The Hessians of the independence and adjusted loglikelihood, respectively.
`C_cholesky`, `C_spectral`	The matrix C in equation (14) of Chandler and Bate (2007), calculated using Cholesky decomposition and spectral decomposition, respectively.
`full_par_names`, `par_names`	The names of the parameters in the full and current models, respectively, if these were supplied in this call or a previous call.
`max_loglik`	The common maximised value of the independence and adjusted loglikelihoods.

loglik, cluster	The arguments loglik and cluster supplied in this call, or a previous call.
loglik_args	A list containing the further arguments passed to loglik via ... in this call, or a previous call.
loglikVecMLE	a vector containing the contributions of individual observations to the independence log-likelihood evaluated at the MLE.
name	The argument name, or the name of the function loglik if name isn't supplied.
nobs	The number of observations.
call	The call to adjust_loglik.

If fixed_pars is not NULL then there are further attributes

fixed_pars	The argument fixed_pars, with names inferred from par_names if this was supplied.
fixed_at	The argument fixed_at, with names inferred from par_names if this was supplied.

If alg_deriv and/or alg_hess were supplied then these are returned as further attributes.

To view an individual attribute called att_name use attr(x, "att_name") or attributes(x)$att_name.

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

summary.chandwich for maximum likelihood estimates and unadjusted and adjusted standard errors.

plot.chandwich for plots of one-dimensional adjusted loglikelihoods.

confint.chandwich, anova.chandwich, coef.chandwich, vcov.chandwich and logLik.chandwich for other chandwich methods.

conf_intervals for confidence intervals for individual parameters.

conf_region for a confidence region for a pair of parameters.

compare_models to compare nested models using an (adjusted) likelihood ratio test.

Examples

```
# ------------------------ Binomial model, rats data ----------------------

# Contributions to the independence loglikelihood
binom_loglik <- function(prob, data) {
  if (prob < 0 || prob > 1) {
    return(-Inf)
  }
  return(dbinom(data[, "y"], data[, "n"], prob, log = TRUE))
}
rat_res <- adjust_loglik(loglik = binom_loglik, data = rats, par_names = "p")
```

```
# Plot the loglikelihoods
plot(rat_res, type = 1:4, legend_pos = "bottom", lwd = 2, col = 1:4)
# MLE, SEs and adjusted SEs
summary(rat_res)

# -------------------------- GEV model, owtemps data -----------------------
# ------------ following Section 5.2 of Chandler and Bate (2007) -----------

# Contributions to the independence loglikelihood
gev_loglik <- function(pars, data) {
  o_pars <- pars[c(1, 3, 5)] + pars[c(2, 4, 6)]
  w_pars <- pars[c(1, 3, 5)] - pars[c(2, 4, 6)]
  if (isTRUE(o_pars[2] <= 0 | w_pars[2] <= 0)) return(-Inf)
  o_data <- data[, "Oxford"]
  w_data <- data[, "Worthing"]
  check <- 1 + o_pars[3] * (o_data - o_pars[1]) / o_pars[2]
  if (isTRUE(any(check <= 0))) return(-Inf)
  check <- 1 + w_pars[3] * (w_data - w_pars[1]) / w_pars[2]
  if (isTRUE(any(check <= 0))) return(-Inf)
  o_loglik <- log_gev(o_data, o_pars[1], o_pars[2], o_pars[3])
  w_loglik <- log_gev(w_data, w_pars[1], w_pars[2], w_pars[3])
  return(o_loglik + w_loglik)
}

# Initial estimates (method of moments for the Gumbel case)
sigma <- as.numeric(sqrt(6 * diag(var(owtemps))) / pi)
mu <- as.numeric(colMeans(owtemps) - 0.57722 * sigma)
init <- c(mean(mu), -diff(mu) / 2, mean(sigma), -diff(sigma) / 2, 0, 0)

# Loglikelihood adjustment for the full model
par_names <- c("mu[0]", "mu[1]", "sigma[0]", "sigma[1]", "xi[0]", "xi[1]")
large <- adjust_loglik(gev_loglik, data = owtemps, init = init,
                       par_names = par_names)
# Rows 1, 3 and 4 of Table 2 of Chandler and Bate (2007)
t(summary(large))

# Loglikelihood adjustment of some smaller models: xi[1] = 0 etc

# Starting from a larger model
medium <- adjust_loglik(larger = large, fixed_pars = "xi[1]")
small <- adjust_loglik(larger = large, fixed_pars = c("sigma[1]", "xi[1]"))
small <- adjust_loglik(larger = medium, fixed_pars = c("sigma[1]", "xi[1]"))

# Starting from scratch
medium <- adjust_loglik(gev_loglik, data = owtemps, init = init,
          par_names = par_names, fixed_pars = "xi[1]")
small <- adjust_loglik(gev_loglik, data = owtemps, init = init,
         par_names = par_names, fixed_pars = c("sigma[1]", "xi[1]"))

# --------- Misspecified Poisson model for negative binomial data ----------

# ... following Section 5.1 of the "Object-Oriented Computation of Sandwich
```

```
# Estimators" vignette of the sandwich package
# https://cran.r-project.org/web/packages/sandwich/vignettes/sandwich-OOP.pdf

# Simulate data
set.seed(123)
x <- rnorm(250)
y <- rnbinom(250, mu = exp(1 + x), size = 1)
# Fit misspecified Poisson model
fm_pois <- glm(y ~ x + I(x^2), family = poisson)
summary(fm_pois)$coefficients

# Contributions to the independence loglikelihood
pois_glm_loglik <- function(pars, y, x) {
  log_mu <- pars[1] + pars[2] * x + pars[3] * x ^ 2
  return(dpois(y, lambda = exp(log_mu), log = TRUE))
}
pars <- c("alpha", "beta", "gamma")
pois_quad <- adjust_loglik(pois_glm_loglik, y = y, x = x, par_names = pars)
summary(pois_quad)

# Providing algebraic derivatives and Hessian
pois_alg_deriv <- function(pars, y, x) {
  mu <- exp(pars[1] + pars[2] * x + pars[3] * x ^ 2)
  return(cbind(y - mu, x * (y - mu), x ^2 * (y - mu)))
}
pois_alg_hess <- function(pars, y, x) {
  mu <- exp(pars[1] + pars[2] * x + pars[3] * x ^ 2)
  alg_hess <- matrix(0, 3, 3)
  alg_hess[1, ] <- -c(sum(mu), sum(x * mu), sum(x ^ 2 * mu))
  alg_hess[2, ] <- -c(sum(x * mu), sum(x ^ 2 * mu), sum(x ^ 3 * mu))
  alg_hess[3, ] <- -c(sum(x ^ 2 * mu), sum(x ^ 3 * mu), sum(x ^ 4 * mu))
  return(alg_hess)
}
pois_quad <- adjust_loglik(pois_glm_loglik, y = y, x = x, p = 3,
                           alg_deriv = pois_alg_deriv, alg_hess = pois_alg_hess)
summary(pois_quad)

got_sandwich <- requireNamespace("sandwich", quietly = TRUE)
if (got_sandwich) {
  # Providing MLE, H and V
  # H and V calculated using bread() and meat() from sandwich package
  n_obs <- stats::nobs(fm_pois)
  pois_quad <- adjust_loglik(pois_glm_loglik, y = y, x = x, p = 3,
                             mle = fm_pois$coefficients,
                             H = -solve(sandwich::bread(fm_pois) / n_obs),
                             V = sandwich::meat(fm_pois) * n_obs)
}
```

---

anova.chandwich *Comparison of nested models*

Description

anova method for objects of class "chandwich". 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 'chandwich'
anova(object, object2, ...)
```

Arguments

object	An object of class "chandwich", returned by adjust_loglik.
object2	An object of class "chandwich", returned by adjust_loglik.
...	Further objects of class "chandwich" and/or arguments to be passed to compare_models. The name of any object of class "chandwich" passed via ... must not match any argument of compare_models or any argument of optim.

Details

For details the adjusted likelihood ratio test see compare_models and Chandler and Bate (2007).

The objects of class "chandwich" need not be provided in nested order: they will be ordered inside anova.chandwich 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

compare_models for an adjusted likelihood ratio test of two models.

adjust_loglik to adjust a user-supplied loglikelihood function.

conf_intervals for confidence intervals for individual parameters.

conf_region for a confidence region for pairs of parameters.

Examples

```
# -------------------------- GEV model, owtemps data -----------------------
# ------------ following Section 5.2 of Chandler and Bate (2007) -----------

gev_loglik <- function(pars, data) {
  o_pars <- pars[c(1, 3, 5)] + pars[c(2, 4, 6)]
  w_pars <- pars[c(1, 3, 5)] - pars[c(2, 4, 6)]
  if (isTRUE(o_pars[2] <= 0 | w_pars[2] <= 0)) return(-Inf)
  o_data <- data[, "Oxford"]
  w_data <- data[, "Worthing"]
  check <- 1 + o_pars[3] * (o_data - o_pars[1]) / o_pars[2]
  if (isTRUE(any(check <= 0))) return(-Inf)
  check <- 1 + w_pars[3] * (w_data - w_pars[1]) / w_pars[2]
  if (isTRUE(any(check <= 0))) return(-Inf)
  o_loglik <- log_gev(o_data, o_pars[1], o_pars[2], o_pars[3])
  w_loglik <- log_gev(w_data, w_pars[1], w_pars[2], w_pars[3])
  return(o_loglik + w_loglik)
}

# Initial estimates (method of moments for the Gumbel case)
sigma <- as.numeric(sqrt(6 * diag(var(owtemps))) / pi)
mu <- as.numeric(colMeans(owtemps) - 0.57722 * sigma)
init <- c(mean(mu), -diff(mu) / 2, mean(sigma), -diff(sigma) / 2, 0, 0)

# Log-likelihood adjustment of the full model
par_names <- c("mu[0]", "mu[1]", "sigma[0]", "sigma[1]", "xi[0]", "xi[1]")
large <- adjust_loglik(gev_loglik, data = owtemps, init = init,
         par_names = par_names)

# Log-likelihood adjustment of some smaller models: xi[1] = 0 etc

medium <- adjust_loglik(larger = large, fixed_pars = "xi[1]")
small <- adjust_loglik(larger = medium, fixed_pars = c("sigma[1]", "xi[1]"))
tiny <- adjust_loglik(larger = small,
                      fixed_pars = c("mu[1]", "sigma[1]", "xi[1]"))

anova(large, medium, small, tiny)
```

`coef.chandwich` *Extract model coefficients method for objects of class "chandwich"*

Description

`coef` method for class "chandwich".

Usage

```
## S3 method for class 'chandwich'
coef(object, complete = FALSE, ...)
```

Arguments

object	an object of class "chandwich", a result of a call to adjust_loglik.
complete	A logical scalar. If complete = TRUE then the full vector of parameter estimates is returned, including any fixed using fixed_pars and fixed_at in the call to adjust_loglik. Otherwise, only the vector of estimates of free parameters is returned. The default is complete = FALSE, which is in sync with vcov.chandwich.
...	Additional optional arguments. At present no optional arguments are used.

Details

The full vector of estimates is taken from attributes(object)$res_MLE and the reduced vector from attributes(object)$MLE.

Value

A numeric vector of estimated parameters, which will be named if names were provided in the call to adjust_loglik.

See Also

vcov.chandwich: vcov method for class "chandwich".

summary.chandwich: summary method for class "chandwich".

adjust_loglik to adjust a user-supplied loglikelihood.

---

compare_models *Comparison of nested models*

Description

Compares 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

```
compare_models(
  larger,
  smaller = NULL,
  approx = FALSE,
  type = c("vertical", "cholesky", "spectral", "none"),
  fixed_pars = NULL,
  fixed_at = rep_len(0, length(fixed_pars)),
  init = NULL,
  ...
)
```

Arguments

larger	An object of class "chandwich" returned by adjust_loglik. The larger of the two models.
smaller	An object of class "chandwich" returned by adjust_loglik. The smaller of the two models. If smaller is supplied then the arguments fixed_pars and fixed_at described below are ignored.
approx	A logical scalar. If approx = TRUE then the approximation detailed by equations (18)-(20) of Chandler and Bate (2007) is used. This option is available only if smaller is supplied. If smaller is not supplied then approx = TRUE is used, with no warning. The approximation doesn't make sense if type = "none". If type = "none" and approx = TRUE then approx is set to FALSE with no warning. If approx = FALSE then the adjusted likelihood is maximised under the restriction imposed by delta, that is, equation (17) of Chandler and Bate (2007) is used.
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.
fixed_pars	A numeric vector. Indices of the components of the **full** parameter vector that are restricted to be equal to the value(s) in fixed_at.
fixed_at	A numeric vector of length 1 or length(fixed_pars). If length(fixed_at) = 1 then the components fixed_pars of the parameter vector are all fixed at fixed_at. If length(fixed_at) = length(fixed_pars) then the component fixed_pars[i] is fixed at fixed_at[i] for each i.
init	(Only relevant if approx = FALSE). A numeric vector of initial values for use in the search for the MLE under the smaller model. Must have length equal to the number of parameters in the smaller of the two models being compared. If init is not supplied, or it is of the wrong length, then attr(smaller, "MLE") is used if smaller is supplied and attr(larger, "MLE")[-fixed_pars] otherwise.
...	Further arguments to be passed to optim. These may include gr, method, lower, upper or control.

Details

The smaller of the two models is specified either by supplying smaller or fixed_pars. If both are supplied then smaller takes precedence.

For full details see Section 3.5 of Chandler and Bate (2007). If approx = FALSE then the a likelihood ratio test of the null hypothesis that the smaller model is a valid simplification of the larger model is carried out directly using equation (17) of Chandler and Bate (2007) based on the adjusted loglikelihood under the larger model, returned by adjust_loglik. This adjusted loglikelihood is maximised subject to the constraint that a subset of the parameters in the larger model are fixed. If smaller is supplied then this maximisation can be avoided using an approximation detailed by equations (18)-(20) of Chandler and Bate (2007), which uses the MLE under the smaller model. The same null distribution (chi-squared with degrees of freedom equal to the number of parameters that are fixed) is used in both cases.

Value

An object of class "compmod", a list with components

- `alrts` the adjusted likelihood ratio test statistic.
- `df` under the null hypothesis that the smaller model is a valid simplification of the larger model the adjusted likelihood ratio test statistic has a chi-squared distribution with `df` degrees of freedom.
- `p_value` the p-value associated with the test of the null hypothesis.
- `larger_mle` the MLE of the parameters under the larger model.
- `smaller_mle` the MLE of the parameters under the smaller model.
- `larger_fixed_pars`, `smaller_fixed_pars` Numeric vectors of the indices of parameters fixed in the larger and smaller models, respectively.
- `larger_fixed_at`, `smaller_fixed_at` Numeric vectors of the values at which the parameters in `larger_fixed_pars` and `smaller_fixed_pars` are fixed.
- `approx` the argument `approx`.

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

`adjust_loglik` to adjust a user-supplied loglikelihood function.

`conf_intervals` for confidence intervals for individual parameters.

`conf_region` for a confidence region for pairs of parameters.

`print.compmod`.

Examples

```
# -------------------------- GEV model, owtemps data ----------------------
# ------------ following Section 5.2 of Chandler and Bate (2007) -----------

gev_loglik <- function(pars, data) {
  o_pars <- pars[c(1, 3, 5)] + pars[c(2, 4, 6)]
  w_pars <- pars[c(1, 3, 5)] - pars[c(2, 4, 6)]
  if (isTRUE(o_pars[2] <= 0 | w_pars[2] <= 0)) return(-Inf)
  o_data <- data[, "Oxford"]
  w_data <- data[, "Worthing"]
  check <- 1 + o_pars[3] * (o_data - o_pars[1]) / o_pars[2]
  if (isTRUE(any(check <= 0))) return(-Inf)
  check <- 1 + w_pars[3] * (w_data - w_pars[1]) / w_pars[2]
  if (isTRUE(any(check <= 0))) return(-Inf)
  o_loglik <- log_gev(o_data, o_pars[1], o_pars[2], o_pars[3])
  w_loglik <- log_gev(w_data, w_pars[1], w_pars[2], w_pars[3])
```

```
  return(o_loglik + w_loglik)
}

# Initial estimates (method of moments for the Gumbel case)
sigma <- as.numeric(sqrt(6 * diag(var(owtemps))) / pi)
mu <- as.numeric(colMeans(owtemps) - 0.57722 * sigma)
init <- c(mean(mu), -diff(mu) / 2, mean(sigma), -diff(sigma) / 2, 0, 0)

# Log-likelihood adjustment of the full model
par_names <- c("mu[0]", "mu[1]", "sigma[0]", "sigma[1]", "xi[0]", "xi[1]")
large <- adjust_loglik(gev_loglik, data = owtemps, init = init,
         par_names = par_names)

# Log-likelihood adjustment of some smaller models: xi[1] = 0 etc

medium <- adjust_loglik(larger = large, fixed_pars = "xi[1]")
small <- adjust_loglik(larger = medium, fixed_pars = c("sigma[1]", "xi[1]"))

# Tests

# Test xi1 = 0 (2 equivalent ways), vertical adjustment
compare_models(large, fixed_pars = "xi[1]")
compare_models(large, medium)
# Test xi1 = 0, using approximation
compare_models(large, medium, approx = TRUE)

# Horizontal adjustments
compare_models(large, medium, type = "cholesky")$p_value
compare_models(large, medium, type = "spectral")$p_value
# No adjustment (independence loglikelihood)
compare_models(large, medium, type = "none")$p_value

# Test sigma1 = 0 for model with xi1 = 0
compare_models(medium, small)
# Test sigma1 = xi1 = 0
compare_models(large, small)

# --------- Misspecified Poisson model for negative binomial data ----------

# ... following Section 5.1 of the "Object-Oriented Computation of Sandwich
# Estimators" vignette of the sandwich package
# https://cran.r-project.org/web/packages/sandwich/vignettes/sandwich-OOP.pdf

# Simulate data
set.seed(123)
x <- rnorm(250)
y <- rnbinom(250, mu = exp(1 + x), size = 1)
# Fit misspecified Poisson model
fm_pois <- glm(y ~ x + I(x^2), family = poisson)
summary(fm_pois)$coefficients

# Contributions to the independence loglikelihood
pois_glm_loglik <- function(pars, y, x) {
```

```
  log_mu <- pars[1] + pars[2] * x + pars[3] * x ^ 2
  return(dpois(y, lambda = exp(log_mu), log = TRUE))
}
pars <- c("alpha", "beta", "gamma")
pois_quad <- adjust_loglik(pois_glm_loglik, y = y, x = x, par_names = pars)
summary(pois_quad)

pois_lin <- adjust_loglik(larger = pois_quad, fixed_pars = "gamma")

# Test the significance of the quadratic term
compare_models(pois_quad, pois_lin)$p_value
compare_models(pois_quad, pois_lin, approx = TRUE)$p_value
```

`confint.chandwich` *Confidence intervals for model parameters*

Description

`confint` method for objects of class `"chandwich"`. Computes confidence intervals for one or more model parameters based on an object returned from `adjust_loglik`.

Usage

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

Arguments

`object`	An object of class `"chandwich"`, returned by `adjust_loglik`.
`parm`	A vector specifying the (unfixed) parameters for which confidence intervals are required. If missing, all parameters are included. Can be either a numeric vector, specifying indices of the components of the **full** parameter vector, or a character vector of parameter names, which must be a subset of those supplied in `par_names` in the call to `adjust_loglik` that produced `object`. `parm` must not have any parameters in common with `attr(object, "fixed_pars")`. `parm` must not contain all of the unfixed parameters, i.e. there is no point in profiling over all the unfixed parameters.
`level`	The confidence level required. A numeric scalar in (0, 1).

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.
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.
...	Further arguments to be passed to conf_intervals.

Details

For details see the documentation for the function conf_intervals, on which confint.chandwich is based.

Value

A matrix with columns giving lower and upper confidence limits for each parameter. These will be labelled as (1 - level)/2 and 1 - (1 - level)/2 in % (by default 2.5% and 97.5%). The row names are the names of the model parameters, if these are available.

See Also

The underlying function conf_intervals. If you would like to plot the profile loglikelihood function for a parameter then call conf_intervals directly and then use the associated plot method. Note that in conf_intervals() a parameter choice is specified using an argument called which_pars, not parm.

conf_region for a confidence region for pairs of parameters.

compare_models for an adjusted likelihood ratio test of two models.

adjust_loglik to adjust a user-supplied loglikelihood function.

Examples

```
# -------------------------- GEV model, owtemps data -----------------------
# ------------ following Section 5.2 of Chandler and Bate (2007) -----------

gev_loglik <- function(pars, data) {
  o_pars <- pars[c(1, 3, 5)] + pars[c(2, 4, 6)]
  w_pars <- pars[c(1, 3, 5)] - pars[c(2, 4, 6)]
  if (isTRUE(o_pars[2] <= 0 | w_pars[2] <= 0)) return(-Inf)
  o_data <- data[, "Oxford"]
  w_data <- data[, "Worthing"]
  check <- 1 + o_pars[3] * (o_data - o_pars[1]) / o_pars[2]
  if (isTRUE(any(check <= 0))) return(-Inf)
  check <- 1 + w_pars[3] * (w_data - w_pars[1]) / w_pars[2]
  if (isTRUE(any(check <= 0))) return(-Inf)
  o_loglik <- log_gev(o_data, o_pars[1], o_pars[2], o_pars[3])
  w_loglik <- log_gev(w_data, w_pars[1], w_pars[2], w_pars[3])
  return(o_loglik + w_loglik)
}

# Initial estimates (method of moments for the Gumbel case)
```

```
sigma <- as.numeric(sqrt(6 * diag(var(owtemps))) / pi)
mu <- as.numeric(colMeans(owtemps) - 0.57722 * sigma)
init <- c(mean(mu), -diff(mu) / 2, mean(sigma), -diff(sigma) / 2, 0, 0)

# Log-likelihood adjustment of the full model
par_names <- c("mu[0]", "mu[1]", "sigma[0]", "sigma[1]", "xi[0]", "xi[1]")
large <- adjust_loglik(gev_loglik, data = owtemps, init = init,
                       par_names = par_names)
confint(large)
confint(large, profile = FALSE)
```

`conf_intervals` *Confidence intervals*

Description

Calculates confidence intervals for individual parameters.

Usage

```
conf_intervals(
  object,
  which_pars = NULL,
  init = NULL,
  conf = 95,
  mult = 1.5,
  num = 10,
  type = c("vertical", "cholesky", "spectral", "none"),
  profile = TRUE,
  ...
)
```

Arguments

`object`	An object of class `"chandwich"` returned by `adjust_loglik`.
`which_pars`	A vector specifying the (unfixed) parameters for which confidence intervals are required. Can be either a numeric vector, specifying indices of the components of the **full** parameter vector, or a character vector of parameter names, which must be a subset of those supplied in `par_names` in the call to `adjust_loglik` that produced `object`. `which_pars` must not have any parameters in common with `attr(object, "fixed_pars")`. `which_pars` must not contain all of the unfixed parameters, i.e. there is no point in profiling over all the unfixed parameters. If missing, all parameters are included.
`init`	A numeric vector of initial estimates of the values of the parameters that are not fixed and are not in `which_pars`. Should have length `attr(object, "p_current") - length(which_pars)`. If `init` is `NULL` or is of the wrong length then the relevant components from the MLE stored in `object` are used.

conf	A numeric scalar in (0, 100). Confidence level for the intervals.
mult	A numeric vector of length 1 or the same length as which_pars. The search for the profile loglikelihood-based confidence limits is conducted over the corresponding symmetric confidence intervals (based on approximate normal theory), extended by a factor of the corresponding component of mult.
num	A numeric scalar. The number of values at which to evaluate the profile loglikelihood either side of the MLE. Increasing num increases the accuracy of the confidence limits, but the code will take longer to run.
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.
profile	A logical scalar. If FALSE then only intervals based on approximate large sample normal theory, which are symmetric about the MLE, are returned (in sym_CI) and prof_CI in the returned object will contain NAs.
...	Further arguments to be passed to optim. These may include gr, method, lower, upper or control.

Details

Calculates (profile, if necessary) likelihood-based confidence intervals for individual parameters, and also provides symmetric intervals based on a normal approximation to the sampling distribution of the estimator. See also the S3 confint method confint.chandwich.

Value

An object of class "confint", a list with components

conf	The argument conf.
cutoff	A numeric scalar. For values inside the confidence interval the profile loglikelihood lies above cutoff.
parameter_vals, prof_loglik_vals	2 * num + 1 by length(which_pars) numeric matrices. Column i of parameter_vals contains the profiled values of parameter which_par[i]. Column i of prof_loglik_vals contains the corresponding values of the profile loglikelihood.
sym_CI, prof_CI	length(which_pars) by 2 numeric matrices. Row i of sym_CI (prof_CI) contains the symmetric (profile loglikelihood-based) confidence intervals for parameter which_pars[i].

If a value in prof_CI is NA then this means that the search for the confidence limit did no extend far enough. A remedy is to increase the value of mult, or the relevant component of mult, and perhaps also increase num.

max_loglik	The value of the adjusted loglikelihood at its maximum, stored in object$max_loglik.
type	The argument type supplied in the call to conf_intervals, i.e. the type of loglikelihood adjustment.
which_pars	The argument which_pars.
name	A character scalar. The name of the model, stored in attr(object, "name").
p_current	The number of free parameters in the current model.

`fixed_pars`, `fixed_at`
: `attr(object, "fixed_pars")` and `attr(object, "fixed_at")`, the arguments `fixed_pars` and `fixed_at` to `adjust_loglik`, if these were supplied.

See Also

`confint.chandwich` S3 confint method for objects of class `"chandwich"` returned from `adjust_loglik`.

`adjust_loglik` to adjust a user-supplied loglikelihood function.

`summary.chandwich` for maximum likelihood estimates and unadjusted and adjusted standard errors.

`plot.chandwich` for plots of one-dimensional adjusted loglikelihoods.

`conf_region` for a confidence region for a pair of parameters.

`compare_models` to compare nested models using an (adjusted) likelihood ratio test.

`plot.confint`, `print.confint`.

Examples

```
# ------------------------- Binomial model, rats data ----------------------

# Contributions to the independence loglikelihood
binom_loglik <- function(prob, data) {
  if (prob < 0 || prob > 1) {
    return(-Inf)
  }
  return(dbinom(data[, "y"], data[, "n"], prob, log = TRUE))
}
rat_res <- adjust_loglik(loglik = binom_loglik, data = rats, par_names = "p")

# 95% likelihood-based confidence intervals, vertically adjusted
ci <- conf_intervals(rat_res)
plot(ci)
# Unadjusted
conf_intervals(rat_res, type = "none")

# ------------------------- GEV model, owtemps data ------------------------
# ------------ following Section 5.2 of Chandler and Bate (2007) -----------

gev_loglik <- function(pars, data) {
  o_pars <- pars[c(1, 3, 5)] + pars[c(2, 4, 6)]
  w_pars <- pars[c(1, 3, 5)] - pars[c(2, 4, 6)]
  if (isTRUE(o_pars[2] <= 0 | w_pars[2] <= 0)) return(-Inf)
  o_data <- data[, "Oxford"]
  w_data <- data[, "Worthing"]
  check <- 1 + o_pars[3] * (o_data - o_pars[1]) / o_pars[2]
  if (isTRUE(any(check <= 0))) return(-Inf)
  check <- 1 + w_pars[3] * (w_data - w_pars[1]) / w_pars[2]
  if (isTRUE(any(check <= 0))) return(-Inf)
  o_loglik <- log_gev(o_data, o_pars[1], o_pars[2], o_pars[3])
  w_loglik <- log_gev(w_data, w_pars[1], w_pars[2], w_pars[3])
  return(o_loglik + w_loglik)
```

```
}

# Initial estimates (method of moments for the Gumbel case)
sigma <- as.numeric(sqrt(6 * diag(var(owtemps))) / pi)
mu <- as.numeric(colMeans(owtemps) - 0.57722 * sigma)
init <- c(mean(mu), -diff(mu) / 2, mean(sigma), -diff(sigma) / 2, 0, 0)

# Log-likelihood adjustment of the full model
par_names <- c("mu[0]", "mu[1]", "sigma[0]", "sigma[1]", "xi[0]", "xi[1]")
large <- adjust_loglik(gev_loglik, data = owtemps, init = init,
                       par_names = par_names)

# 95% likelihood-based confidence intervals, vertically adjusted
large_v <- conf_intervals(large, which_pars = c("xi[0]", "xi[1]"))
large_v
plot(large_v)
plot(large_v, which_par = "xi[1]")

# Unadjusted
large_none <- conf_intervals(large, which_pars = c("xi[0]", "xi[1]"),
                             type = "none")
large_none
plot(large_v, large_none)
plot(large_v, large_none, which_par = "xi[1]")


# --------- Misspecified Poisson model for negative binomial data ----------

# ... following Section 5.1 of the "Object-Oriented Computation of Sandwich
# Estimators" vignette of the sandwich package
# https://cran.r-project.org/web/packages/sandwich/vignettes/sandwich-OOP.pdf

# Simulate data
set.seed(123)
x <- rnorm(250)
y <- rnbinom(250, mu = exp(1 + x), size = 1)
# Fit misspecified Poisson model
fm_pois <- glm(y ~ x + I(x^2), family = poisson)
summary(fm_pois)$coefficients

# Contributions to the independence loglikelihood
pois_glm_loglik <- function(pars, y, x) {
  log_mu <- pars[1] + pars[2] * x + pars[3] * x ^ 2
  return(dpois(y, lambda = exp(log_mu), log = TRUE))
}
pars <- c("alpha", "beta", "gamma")
pois_quad <- adjust_loglik(pois_glm_loglik, y = y, x = x, par_names = pars)
conf_intervals(pois_quad)
```

---

conf_region *Two-dimensional confidence regions*

Description

Calculates the (profile, if necessary) loglikelihood for a pair of parameters from which confidence regions can be plotted using plot.confreg.

Usage

```
conf_region(
  object,
  which_pars = NULL,
  range1 = c(NA, NA),
  range2 = c(NA, NA),
  conf = 95,
  mult = 2,
  num = c(10, 10),
  type = c("vertical", "cholesky", "spectral", "none"),
  ...
)
```

Arguments

object	An object of class "chandwich" returned by adjust_loglik.
which_pars	A vector of length 2 specifying the 2 (unfixed) parameters for which confidence region is required. Can be either a numeric vector, specifying indices of the components of the **full** parameter vector, or a character vector of parameter names, which must be a subset of those supplied in par_names in the call to adjust_loglik that produced object. which_pars must not have any parameters in common with attr(object, "fixed_pars"). which_pars must not contain all of the unfixed parameters, i.e. there is no point in profiling over all the unfixed parameters. If which_pars is not supplied but the current model has exactly two free parameters, i.e. attr(object, "p_current") = 2 then which_pars is set to attr(object, "free_pars") = 2.
range1, range2	Numeric vectors of length 2. Respective ranges (of the form c(lower, upper)) of values of which_pars[1] and which_pars[2] over which to profile. Missing values in range1 and/or range2 are filled in using conf and mult. See below for details.
conf	A numeric scalar in (0, 100). The highest confidence level of interest. This is only relevant if range1 and/or range2 are not completely specified. In that event conf is used, in combination with mult, to try to set up the grid of parameter values to include the largest confidence region of interest.
mult	A numeric vector of length 1 or the same length as which_pars. The search for the profile loglikelihood-based confidence limits is conducted over the corresponding symmetric confidence intervals (based on approximate normal theory), extended by a factor of the corresponding component of mult.
num	A numeric vector of length 1 or 2. The numbers of values at which to evaluate the profile loglikelihood either side of the MLE. num[i] relates to which_pars[i]. If num has length 1 then num is replicated to have length 2.

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.
...	Further arguments to be passed to optim. These may include gr, method, lower, upper or control. Any arguments that are not appropriate for optim, i.e. not in methods::formalArgs(stats::optim), will be removed without warning.

Value

An object of class "confreg", a list with components

grid1, grid2	Numeric vectors. Respective values of which_pars[1] and which_pars[2] in the grid over which the (profile) loglikelihood is evaluated.
max_loglik	A numeric scalar. The value value of the loglikelihood at its maximum.
prof_loglik	An 2 num + 1 by 2 num + 1 numeric matrix containing the values of the (profile) loglikelihood.
type	A character scalar. The input type.
which_pars	A numeric or character vector. The input which_pars. If the which_pars was numeric then it is supplemented by the parameter names, if these are available in object.
name	A character scalar. The name of the model, stored in attr(object, "name").

See Also

adjust_loglik to adjust a user-supplied loglikelihood function.

conf_intervals for confidence intervals for individual parameters.

compare_models to compare nested models using an (adjusted) likelihood ratio test.

plot.confreg.

Examples

```
# -------------------------- GEV model, owtemps data -----------------------
# ------------ following Section 5.2 of Chandler and Bate (2007) -----------

gev_loglik <- function(pars, data) {
  o_pars <- pars[c(1, 3, 5)] + pars[c(2, 4, 6)]
  w_pars <- pars[c(1, 3, 5)] - pars[c(2, 4, 6)]
  if (isTRUE(o_pars[2] <= 0 | w_pars[2] <= 0)) return(-Inf)
  o_data <- data[, "Oxford"]
  w_data <- data[, "Worthing"]
  check <- 1 + o_pars[3] * (o_data - o_pars[1]) / o_pars[2]
  if (isTRUE(any(check <= 0))) return(-Inf)
  check <- 1 + w_pars[3] * (w_data - w_pars[1]) / w_pars[2]
  if (isTRUE(any(check <= 0))) return(-Inf)
  o_loglik <- log_gev(o_data, o_pars[1], o_pars[2], o_pars[3])
  w_loglik <- log_gev(w_data, w_pars[1], w_pars[2], w_pars[3])
  return(o_loglik + w_loglik)
}
```

```
# Initial estimates (method of moments for the Gumbel case)
sigma <- as.numeric(sqrt(6 * diag(var(owtemps))) / pi)
mu <- as.numeric(colMeans(owtemps) - 0.57722 * sigma)
init <- c(mean(mu), -diff(mu) / 2, mean(sigma), -diff(sigma) / 2, 0, 0)

# Log-likelihood adjustment of the full model
par_names <- c("mu[0]", "mu[1]", "sigma[0]", "sigma[1]", "xi[0]", "xi[1]")
large <- adjust_loglik(gev_loglik, data = owtemps, init = init,
                       par_names = par_names)


# Plots like those in Figure 4 of Chandler and Bate (2007)
# (a)
which_pars <- c("mu[0]", "mu[1]")
reg_1 <- conf_region(large, which_pars = which_pars)
reg_none_1 <- conf_region(large, which_pars = which_pars, type = "none")
plot(reg_1, reg_none_1)
# (b)
which_pars <- c("sigma[0]", "sigma[1]")
reg_2 <- conf_region(large, which_pars = which_pars)
reg_none_2 <- conf_region(large, which_pars = which_pars, type = "none")
plot(reg_2, reg_none_2)
# (c)
# Note: the naive and bivariate model contours are the reversed in the paper
which_pars <- c("sigma[0]", "xi[0]")
reg_3 <- conf_region(large, which_pars = which_pars)
reg_none_3 <- conf_region(large, which_pars = which_pars, type = "none")
plot(reg_3, reg_none_3)


# --------- Misspecified Poisson model for negative binomial data ----------

# ... following Section 5.1 of the "Object-Oriented Computation of Sandwich
# Estimators" vignette of the sandwich package
# https://cran.r-project.org/web/packages/sandwich/vignettes/sandwich-OOP.pdf

# Simulate data
set.seed(123)
x <- rnorm(250)
y <- rnbinom(250, mu = exp(1 + x), size = 1)
# Fit misspecified Poisson model
fm_pois <- glm(y ~ x + I(x^2), family = poisson)
summary(fm_pois)$coefficients

# Contributions to the independence loglikelihood
pois_glm_loglik <- function(pars, y, x) {
  log_mu <- pars[1] + pars[2] * x + pars[3] * x ^ 2
  return(dpois(y, lambda = exp(log_mu), log = TRUE))
}
pars <- c("alpha", "beta", "gamma")
# Linear model (gamma fixed at 0)
pois_lin <- adjust_loglik(pois_glm_loglik, y = y, x = x, par_names = pars,
                          fixed_pars = "gamma")
```

```
pois_vertical <- conf_region(pois_lin)
pois_none <- conf_region(pois_lin, type = "none")
plot(pois_none, pois_vertical, conf = c(50, 75, 95, 99), col = 2:1, lwd = 2,
     lty = 1)
```

`logLik.chandwich` *Extract log-likelihood for objects of class "chandwich"*

Description

`logLik` method for class "chandwich".

Usage

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

Arguments

`object`	an object of class "chandwich", a result of a call to `adjust_loglik`.
`...`	Additional optional arguments. At present no optional arguments are used.

Details

The value of the maximised (independence) loglikelihood is extracted from `attr(object, "max_loglik")`. It is also equal to `object(attr(object, "MLE"))`.

Value

An object of class `"logLik"`: a numeric scalar with value equal to the maximised (independence) loglikelihood, that is, the value of the independence loglikelihood at the MLE, and the attribute `"df"`, which gives the number of free parameters estimated.

See Also

`coef.chandwich`: `coef` method for class "chandwich".

`vcov.chandwich`: `vcov` method for class "chandwich".

`summary.chandwich`: `summary` method for class "chandwich".

`adjust_loglik` to adjust a user-supplied loglikelihood.

log_gev *The Generalised Extreme Value Log-Density Function*

Description

Log-Density function of the generalised extreme value (GEV) distribution

Usage

```
log_gev(x, loc = 0, scale = 1, shape = 0)
```

Arguments

x	Numeric vectors of quantiles.
loc, scale, shape	Numeric scalars. Location, scale and shape parameters. scale must be positive.

Details

It is assumed that x**,** loc **=** μ**,** scale **=** σ **and** shape **=** ξ **are such that the GEV density is non-zero, i.e. that** $1 + \xi(x - \mu)/\sigma > 0$**. No check of this, or that** scale **> 0 is performed in this function.**

The distribution function of a GEV distribution with parameters loc = μ, scale = σ (>0) and shape = ξ is

$$F(x) = exp-[1 + \xi(x - \mu)/\sigma]^{(} - 1/\xi)$$

for $1 + \xi(x - \mu)/\sigma > 0$. If $\xi = 0$ the distribution function is defined as the limit as ξ tends to zero. The support of the distribution depends on ξ: it is $x <= \mu - \sigma/\xi$ for $\xi < 0$; $x >= \mu - \sigma/\xi$ for $\xi > 0$; and x is unbounded for $\xi = 0$. Note that if $\xi < -1$ the GEV density function becomes infinite as x approaches $\mu - \sigma/\xi$ from below.

See https://en.wikipedia.org/wiki/Generalized_extreme_value_distribution for further information.

Value

A numeric vector of value(s) of the log-density of the GEV distribution.

References

Jenkinson, A. F. (1955) The frequency distribution of the annual maximum (or minimum) of meteorological elements. *Quart. J. R. Met. Soc.*, **81**, 158-171. Chapter 3: doi:10.1002/qj.49708134804

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

Examples

```
log_gev(1:4, 1, 0.5, 0.8)
log_gev(1:3, 1, 0.5, -0.2)
```

owtemps *Oxford and Worthing annual maximum temperatures*

Description

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

Usage

```
owtemps
```

Format

A dataframe with 80 rows and 2 columns, named Oxford and 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.chandwich *Plot diagnostics for a chandwich object*

Description

`plot` method for class "chandwich". Only applicable to an object `x` for which `attr(x, "p_current") = 1`, i.e. a model with one free parameter.

Usage

```
## S3 method for class 'chandwich'
plot(x, y, type = 1, legend = length(type) > 1, legend_pos = "topleft", ...)
```

Arguments

x	an object of class "chandwich", a result of a call to adjust_loglik.
y	Not used.
type	An integer vector, a subset of the numbers 1:4. Indicates which loglikelihoods to plot: 1 for "vertical" adjustment; 2 for "cholesky" (horizontal adjustment); 3 for "spectral" (horizontal adjustment); 4 for no adjustment, i.e. based on the independence loglikelihood.
legend	A logical scalar or a character vector. If this is supplied then a legend is added to the plot. If legend is a character vector then it is used as the argument legend to legend. Otherwise, i.e. if legend = TRUE then the argument type is used.
legend_pos	The position of the legend (if required) specified using the argument x in legend.
...	Additional arguments passed to matplot or legend. The arguments col, lty and lwd will be used (in a consistent way) by both matplot and legend. If the argument xlim to matplot is not supplied then the MLE minus (for lower) or plus (for upper) standard errors is used. If type does not include 4 then adjusted standard errors are used. Otherwise, the larger of the adjusted and unadjusted standard errors are used.

Value

Nothing is returned.

See Also

adjust_loglik to adjust a user-supplied loglikelihood function.

summary.chandwich for maximum likelihood estimates and unadjusted and adjusted standard errors.

conf_intervals and plot.confint to plot confidence intervals for individual parameters.

conf_region and plot.confreg to plot a confidence region for a pair of parameters.

Examples

```
# ------------------------ Binomial model, rats data ----------------------

# Contributions to the independence loglikelihood
binom_loglik <- function(prob, data) {
  if (prob < 0 || prob > 1) {
    return(-Inf)
  }
  return(dbinom(data[, "y"], data[, "n"], prob, log = TRUE))
}
rat_res <- adjust_loglik(loglik = binom_loglik, data = rats, par_names = "p")

# Vertically adjusted loglikelihood only
plot(rat_res)
# Three adjusted loglikelihoods and the independence loglikelihood
plot(rat_res, type = 1:4)
```

```
# Plot over (0,1) and reposition the legend
plot(rat_res, type = 1:4, xlim = c(0, 1), legend_pos = "bottom")
```

`plot.confint` *Plot diagnostics for a confint object*

Description

`plot` method for class "confint". Plots the (profile) loglikelihood for a parameter using the values calculated by `conf_intervals`. Up to 4 different types of loglikelihood (see the argument `type` to the function returned by `adjust_loglik`) may be superimposed on the same plot. By default (`add_lines = TRUE`) the confidence limits calculated in `conf_intervals` are indicated on the plot .

Usage

```
## S3 method for class 'confint'
plot(
  x,
  y = NULL,
  y2 = NULL,
  y3 = NULL,
  which_par = x$which_pars[1],
  conf = x$conf,
  add_lines = TRUE,
  legend = any(c(!is.null(y), !is.null(y2), !is.null(y3))),
  legend_pos = "topleft",
  ...
)
```

Arguments

`x, y, y2, y3`	objects of class `"confint"`, results of calls to `conf_intervals` for a common model. A (profile) loglikelihood will be plotted for each object.
`which_par`	A scalar specifying the parameter for which the plot is produced. Can be either a numeric scalar, specifying index of the component of the **full** parameter vector, or a character scalar parameter name. The former must be in `x$which_pars`, the latter must be in `names(x$which_pars)`.
`conf`	A numeric vector of values in (0, 100). If `add_lines = TRUE` then a horizontal line is added for each value in `conf`. If `conf` is not supplied then the value stored in `x$conf` is used.
`add_lines`	A logical scalar. Whether or not to add horizontal lines to the plot to identify the confidence limits. If there is only one input `"confint"` object then the values of the confidence limits will be added to the horizontal axis.
`legend`	A logical scalar or a character vector. If this is supplied then a legend is added to the plot. If `legend` is a character vector then it is used as the argument `legend` to `legend`. Otherwise, i.e. if `legend = TRUE` then the component `type` of the input object(s) `x, y, y2, y3` are used.

legend_pos	The position of the legend (if required) specified using the argument x in legend.
...	Additional arguments passed to matplot or legend. The arguments col, lty and lwd will be used (in a consistent way) by both matplot and legend.

Value

Nothing is returned.

Examples

See the examples in conf_intervals.

See Also

adjust_loglik to adjust a user-supplied loglikelihood function.

conf_intervals for confidence intervals for individual parameters.

conf_region and plot.confreg to plot a confidence region for a pair of parameters.

---

plot.confreg *Plot diagnostics for a confreg object*

Description

plot method for class "confreg". Plots confidence regions for pairs of parameters using the profile loglikelihood values calculated by conf_region. Up to 4 different types of loglikelihood (see the argument type to the function returned by adjust_loglik) may be superimposed on the same plot.

Usage

```
## S3 method for class 'confreg'
plot(
  x,
  y = NULL,
  y2 = NULL,
  y3 = NULL,
  conf = 95,
  legend = any(c(!is.null(y), !is.null(y2), !is.null(y3))),
  legend_pos = "topleft",
  ...
)
```

Arguments

x, y, y2, y3	objects of class "confreg", results of calls to conf_region for a common model and a common value of which_pars. Contours are plotted for each object.
conf	A numeric vector of confidence levels, i.e. numbers in (0, 100). A confidence region contour is plotted for each value in conf.
legend	A logical scalar or a character vector. If this is supplied then a legend is added to the plot. If legend is a character vector then it is used as the argument legend to legend. Otherwise, i.e. if legend = TRUE then the component type of the input object(s) x, y, y2, y3 are used.
legend_pos	The position of the legend (if required) specified using the argument x in legend.
...	Additional arguments passed to contour or legend. The arguments col, lty and lwd will be used (in a consistent way) by both contour and legend.

Value

Nothing is returned.

Examples

See the examples in conf_region.

See Also

adjust_loglik to adjust a user-supplied loglikelihood function.

conf_region for a confidence region for a pair of parameters.

conf_intervals and plot.confint to plot confidence intervals for individual parameters.

---

print.chandwich *Print method for objects of class "chandwich"*

Description

print method for class "chandwich".

Usage

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

Arguments

x	an object of class "chandwich", a result of a call to adjust_loglik.
...	Additional optional arguments. At present no optional arguments are used.

Details

Just prints the original call to adjust_loglik and a character vector giving the names of the attributes (produced using ls(attributes(x))) to the function returned from adjust_loglik. To view an individual attribute called att_name use attr(x, "att_name") or attributes(x)$att_name.

Value

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

See Also

summary.chandwich: summary method for class "chandwich".

adjust_loglik to adjust a user-supplied loglikelihood.

---

print.compmod *Print method for objects of class "compmod"*

Description

print method for class "compmod".

Usage

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

Arguments

x	an object of class "compmod", a result of a call to compare_models.
digits	An integer. The argument digits to signif.
...	Additional optional arguments. At present no optional arguments are used.

Details

Prints the name of the model, the null (H0) and alternative hypotheses (HA), the test statistic, degrees of freedom and the p-value. If the test is based on the approximation detailed by equations (18)-(20) of Chandler and Bate (2007), rather than equation (17), then this stated.

Value

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

Examples

See the examples in compare_models.

See Also

adjust_loglik to adjust a user-supplied loglikelihood function.

compare_models to compare nested models using an (adjusted) likelihood ratio test.

---

print.confint *Print method for objects of class "confint"*

Description

print method for class "confint".

Usage

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

Arguments

x	an object of class "confint", a result of a call to conf_intervals.
digits	An integer. The argument digits to print.default.
...	Additional optional arguments. At present no optional arguments are used.

Details

Prints the name of the model, details of any fixed parameters, the confidence level of the interval(s) and whether or not the loglikelihood has been adjusted, and symmetric and (profile) likelihood based intervals.

Value

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

Examples

See the examples in conf_intervals.

See Also

adjust_loglik to adjust a user-supplied loglikelihood function.

conf_intervals for confidence intervals for individual parameters.

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

Description

`print` method for an object `x` of class "summary.chandwich".

Usage

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

Arguments

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

Value

Prints a numeric matrix with 3 columns and the number of rows equal to the number of parameters in the current model, i.e. `attr(object, "p_current")`. The columns contain: the maximum likelihood estimates (MLE), unadjusted standard errors (SE) and adjusted standard errors (adjSE).

Examples

See the examples in `adjust_loglik`.

See Also

`adjust_loglik` to adjust a user-supplied loglikelihood.

`summary.chandwich` for a diagnostic plot.

profile_loglik *Profile loglikelihood*

Description

Calculates the profile loglikelihood for a subset of the model parameters. This function is provided primarily so that it can be called by `conf_intervals` and `conf_region`.

Usage

```
profile_loglik(
  object,
  prof_pars = NULL,
  prof_vals = NULL,
  init = NULL,
  type = c("vertical", "cholesky", "spectral", "none"),
  ...
)
```

Arguments

object	An object of class "chandwich" returned by adjust_loglik.
prof_pars	A vector specifying the subset of the (unfixed) parameters over which to profile. Can be either a numeric vector, specifying indices of the components of the **full** parameter vector, or a character vector of parameter names, which must be a subset of those supplied in par_names in the call to adjust_loglik that produced object. prof_pars must not have any parameters in common with attr(object, "fixed_pars"). prof_pars must not contain all of the unfixed parameters, i.e. there is no point in profiling over all of the unfixed parameters.
prof_vals	A numeric vector. Values of the parameters in prof_pars. If prof_vals = NULL then the MLEs stored in object of the parameters prof_pars are used.
init	A numeric vector of initial estimates of the values of the parameters that are not fixed and are not in prof_pars. Should have length attr(object, "p_current") - length(prof_pars). If init is NULL or is of the wrong length then the relevant components from the MLE stored in object are used.
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.
...	Further arguments to be passed to optim. These may include gr, method, lower, upper or control.

Value

A numeric vector of length 1. The value of the profile loglikelihood. The returned object has the attribute "free_pars" giving the optimal values of the parameters that remain after the parameters prof_pars and attr(object, "fixed_pars") have been removed from the full parameter vector. If there are no such parameters, which happens if an attempt is made to profile over *all* non-fixed parameters, then this attribute is not present and the value returned is calculated using the function object.

See Also

adjust_loglik to adjust a user-supplied loglikelihood function.

conf_intervals for confidence intervals for individual parameters.

conf_region for a confidence region for a pair of parameters.

Examples

```
# -------------------------- GEV model, owtemps data ----------------------
# ------------ following Section 5.2 of Chandler and Bate (2007) -----------

gev_loglik <- function(pars, data) {
  o_pars <- pars[c(1, 3, 5)] + pars[c(2, 4, 6)]
  w_pars <- pars[c(1, 3, 5)] - pars[c(2, 4, 6)]
  if (isTRUE(o_pars[2] <= 0 | w_pars[2] <= 0)) return(-Inf)
  o_data <- data[, "Oxford"]
  w_data <- data[, "Worthing"]
  check <- 1 + o_pars[3] * (o_data - o_pars[1]) / o_pars[2]
  if (isTRUE(any(check <= 0))) return(-Inf)
  check <- 1 + w_pars[3] * (w_data - w_pars[1]) / w_pars[2]
  if (isTRUE(any(check <= 0))) return(-Inf)
  o_loglik <- log_gev(o_data, o_pars[1], o_pars[2], o_pars[3])
  w_loglik <- log_gev(w_data, w_pars[1], w_pars[2], w_pars[3])
  return(o_loglik + w_loglik)
}

# Initial estimates (method of moments for the Gumbel case)
sigma <- as.numeric(sqrt(6 * diag(var(owtemps))) / pi)
mu <- as.numeric(colMeans(owtemps) - 0.57722 * sigma)
init <- c(mean(mu), -diff(mu) / 2, mean(sigma), -diff(sigma) / 2, 0, 0)

# Log-likelihood adjustment of the full model
par_names <- c("mu[0]", "mu[1]", "sigma[0]", "sigma[1]", "xi[0]", "xi[1]")
large <- adjust_loglik(gev_loglik, data = owtemps, init = init,
                       par_names = par_names)

# Profile loglikelihood for xi1, evaluated at xi1 = 0
profile_loglik(large, prof_pars = "xi[1]", prof_vals = 0)

# Model with xi1 fixed at 0
medium <- adjust_loglik(larger = large, fixed_pars = "xi[1]")
# Profile loglikelihood for xi0, evaluated at xi0 = -0.1
profile_loglik(medium, prof_pars = "xi[0]", prof_vals = -0.1)
```

`rats` *Rat tumor data*

Description

Tumor incidence in 71 groups of rate from Tarone (1982). The matrix `rat` has 71 rows and 2 columns. Each row relates to a different group of rats. The first column (`y`) contains the number of rats with tumors. The second column (`n`) contains the total number of rats.

Usage

```
rats
```

Format

A matrix with 71 rows and 2 columns.

Source

Table 5.1 of Gelman, A., Carlin, J. B., Stern, H. S. Dunson, D. B., Vehtari, A. and Rubin, D. B. (2013) *Bayesian Data Analysis*, Chapman & Hall / CRC. http://www.stat.columbia.edu/~gelman/book/data/rats.asc

References

Tarone, R. E. (1982) The use of historical information in testing for a trend in proportions. *Biometrics*, **38**, 215-220.

---

summary.chandwich *Summarizing adjusted loglikelihoods*

Description

summary method for class "chandwich"

Usage

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

Arguments

object	an object of class "chandwich", a result of a call to adjust_loglik.
digits	An integer. Used for number formatting with signif.
...	Additional optional arguments. At present no optional arguments are used.

Value

Returns a numeric matrix with 3 columns and the number of rows equal to the number of parameters in the current model, i.e. attr(object, "p_current"). The columns contain: the maximum likelihood estimates (MLE), unadjusted standard errors (SE) and adjusted standard errors (adjSE).

Examples

See the examples in adjust_loglik.

See Also

adjust_loglik to adjust a user-supplied loglikelihood function.

plot.chandwich for plots of one-dimensional adjusted loglikelihoods.

vcov.chandwich *Calculate the variance-covariance matrix for an object of class "chandwich"*

Description

vcov method for class "chandwich".

Usage

```
## S3 method for class 'chandwich'
vcov(object, complete = FALSE, adjusted = TRUE, ...)
```

Arguments

object	an object of class "chandwich", a result of a call to adjust_loglik.
complete	A logical scalar. If complete = TRUE then the full variance-covariance matrix is returned, including any fixed using fixed_pars and fixed_at in the call to adjust_loglik. In this instance the corresponding elements of the matrix are NA. Otherwise, the variance-covariance matrix of only the free parameters is returned. The default is complete = FALSE, which is in sync with coef.chandwich.
adjusted	A logical scalar. If adjusted = TRUE then the variance-covariance matrix is estimated using a sandwich estimator. Otherwise, the inverse of the observed information at the MLE is used.
...	Additional optional arguments. At present no optional arguments are used.

Details

The variance-covariance matrix is based on attributes(object)$adjVC for adjusted = TRUE and attributes(object)$VC for adjusted = FALSE. These return the estimate variance-covariance matrix of only the free parameters.

Value

A numeric matrix. The dimensions will be named if names were provided in the call to adjust_loglik.

See Also

coef.chandwich: coef method for class "chandwich".

summary.chandwich: summary method for class "chandwich".

adjust_loglik to adjust a user-supplied loglikelihood.

Index