Package ‘profileCI’

January 27, 2026

Type Package

Title Profiling a Log-Likelihood to Calculate Confidence Intervals

Version 1.1.1

Date 2026-01-27

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

Description Provides tools for profiling a user-supplied log-likelihood function to calculate confidence intervals for model parameters. Speed of computation can be improved by adjusting the step sizes in the profiling and/or starting the profiling from limits based on the approximate large sample normal distribution for the maximum likelihood estimator of a parameter. The accuracy of the limits can be set by the user. A plot method visualises the log-likelihood and confidence interval. Cases where the profile log-likelihood flattens above the value at which a confidence limit is defined can be handled, leading to a limit at plus or minus infinity. Disjoint confidence intervals will not be found.

Imports graphics, itp, stats, utils

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 4.1.0)

Suggests testthat (>= 3.0.0)

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

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

Config/testthat/edition 3

RoxygenNote 7.3.3

NeedsCompilation no

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

Repository CRAN

Date/Publication 2026-01-27 00:10:08 UTC

Contents

`profileCI-package` *profileCI: Profiling a Log-Likelihood to Calculate Confidence Intervals*

Description

Provides tools for profiling a user-supplied log-likelihood function to calculate confidence intervals for model parameters. Speed of computation can be improved by adjusting the step sizes in the profiling and/or starting the profiling from limits based on the approximate large sample normal distribution for the maximum likelihood estimator of a parameter. The accuracy of the limits can be set by the user. A plot method visualises the log-likelihood and confidence interval. Cases where the profile log-likelihood flattens above the value at which a confidence limit is defined can be handled, leading to a limit at plus or minus infinity. Disjoint confidence intervals will not be found.

Details

The main function is `profileCI`, which profiles the log-likelihood, with respect to one parameter at a time.

Author(s)

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

See Also

`profileCI` and `plot.profileCI`.

`logLikFn` *Calculate Log-Likelihood*

Description

This is a generic function for calculating a log-likelihood for an object at input parameter values.

Usage

```
logLikFn(object, pars, ...)

## S3 method for class 'glm'
logLikFn(object, pars, ...)

## S3 method for class 'nls'
logLikFn(object, pars, ...)
```

Arguments

`object`	A fitted model object.
`pars`	A numeric vector of parameters of the model.
`...`	Further arguments. None are used in the `logLikFn.glm` and `logLikFn.nls` generics.

Details

This generic function has been created to enable a function that calculates the log-likelihood for a parametric model at a given set of parameter values in `pars` to be available to the function `profileCI`.

Value

A numeric scalar. The value of the log-likelihood function for the fitted model object `object` for parameter values `pars`.

The `logLikFn.glm` generic is specifically for the unweighted Poisson log-linear GLM case.

The `logLikFn.nls` generic is more general and should work for all model objects returned by `stats::nls`.

See Also

`profileCI`, `stats::glm`, `stats::nls`.

`profileCI` *Confidence Intervals using Profile Likelihood*

Description

Calculates confidence intervals for one or more parameters in a fitted model object. A function that returns the log-likelihood must be supplied, either directly via the argument `loglik` or using a `logLikFn` S3 generic.

Usage

```
profileCI(
  object,
  loglik,
  ...,
  parm = "all",
  level = 0.95,
  profile = TRUE,
  mult = 32,
  faster = TRUE,
  epsilon = -1,
  flat = 1e-06,
  lb,
  ub,
  optim_args = list()
)
```

Arguments

`object`	A fitted model object. This object must have a `coef` S3 method that returns a **named** vector of parameter values. If `faster = TRUE` or `profile = FALSE` then it must also have a `vcov` S3 method that returns a matrix with `dimnames` that match those in the vector returned by the `coef` method. If necessary, these may be created using `.S3method()`. For example, if `object` is a list inheriting from class `"foo"`, with coefficients in `object$coefficients` and variance-covariance matrix in `object$vcov`, then use `.S3method("coef", "foo", function(x) x$coefficients)` and `.S3method("vcov", "foo", function(x) x$vcov)`.
`loglik`	A named function that returns the log-likelihood based on input parameter values and data. The first argument must be the vector of model parameters. If the likelihood is zero for any observation in the data then the function should return `-Inf`. Alternatively, `loglik` does not need to be supplied if a `logLikFn` S3 method has been created for `object`. The `profileCI` package provides `logLikFn.glm`, which is used in an example in **Examples**.
`...`	Further arguments to be passed to `loglik`.
`parm`	A vector specifying the parameters for which confidence intervals are calculated, either a vector of numbers or a vector of names. The default, `which = "all"`, produces confidence intervals for all the parameters.
`level`	The confidence level required. A numeric scalar in (0, 1).
`profile`	A logical scalar. If `TRUE` then confidence intervals based on a profile log-likelihood are returned. If `FALSE` then intervals based on approximate large sample normal theory, which are symmetric about the MLE, are returned.
`mult`	A positive numeric scalar. Controls the increment by which the parameter of interest is increased/decreased when profiling above/below its MLE. The increment is `mult * se / 100` where `se` is the estimated standard error of the estimator of the parameter. Decreasing `mult` profiles at more points but will be slower.

`faster`	A logical scalar. If `faster = TRUE` then the profiling of the log-likelihood in search of a lower (upper) confidence limit is started at the corresponding symmetric lower (upper) confidence limit. However, `faster = FALSE` is more robust, particularly if a symmetric limit is close to, or beyond, the edge of the parameter space.
`epsilon`	Only relevant if `profile = TRUE`. A numeric vector of values that determine the accuracy of the confidence limits. `epsilon` is recycled to the length of the parameter vector `parm`. • If `epsilon[i] > 0` then this value is passed as the argument `epsilon` to the `itp::itp` function, which estimates the parameter values for which the profile log-likelihood for parameter `i` drops to the value that defines the confidence limits, once profiling has been successful in finding an interval within which this value lies. • If `epsilon[i] < 0` monotonic cubic spline interpolation (`stats::splinefun` with `method = "hyman"`) is used, which will tend to be faster. • If `epsilon[i] = 0` then linear interpolation is used, which will be faster still.
`flat`	A positive numeric scalar used to avoid continuing a search for a confidence limits in cases where the profile log-likelihood becomes flat. If a reduction in profile log-likelihood is less than `flat * mult / 100` then the search is stopped. The value of the returned confidence limit is `Inf` for an upper limit and `-Inf` for a lower limit.
`lb`, `ub`	Optional numeric vectors of length `length(parm)`. If supplied, `lb[i]` and/or `ub[i]` place respective lower and upper bounds on the interval over which profiling takes place for parameter `parm[i]`. If a bound is reached before a confidence limit is determined or before the profile log-likelihood is determined to have become flat, then the relevant limit is returned as `NA`. Elementwise, `lb` must be smaller than, and `ub` larger than, `coef(object)`.
`optim_args`	A list of further arguments (other than `par` and `fn`) to pass to `stats::optim`. For example, `optim_args = list(method = "BFGS", control = list(trace = 1))` changes the method used from `"Nelder-Mead"` to `"BFGS"` and sets `trace` to provide the lowest level of tracing information.

Details

The default, `epsilon = -1`, should work well enough in most circumstances, but to achieve a specific accuracy set `epsilon` to be a small positive value, for example, `epsilon = 1e-4`.

The defaults `mult = 32` and `faster = TRUE` are designed to calculate confidence intervals fairly quickly. If the object returned from `profileCI` is plotted, using `plot.profileCI`, then we will not obtain a smooth plot of a profile log-likelihood. Setting `faster = FALSE` and reducing `mult`, perhaps to `8` or `16` should produce a smoother plot.

The arguments `flat1, lb` and `ub` are provided to prevent a call to `profileCI` hanging in a search for a confidence limit that will never be found.

Value

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

If `profile = TRUE` then the returned object has extra attributes `crit`, `level` and `for_plot`. The latter is a named list of length equal to the number of parameters. Each component is a 2-column numeric matrix. The first column contains values of the parameter and the second column the corresponding values of the profile log-likelihood. The profile log-likelihood is equal to the attribute `crit` at the limits of the confidence interval. The attribute `level` is the input argument `level`. If `faster = FALSE` or `epsilon > 0` then the attributes `lower_pars` and `upper_pars` are lists that provide, for each profiling, the values of the parameters for the last maximisation of the log-likelihood.

A matrix with columns giving the object `c("profileCI", "matrix", "array")`.

See Also

`plot.profileCI` and `print.profileCI`.

Examples

```
## From example(glm)
counts <- c(18, 17, 15, 20, 10, 20, 25, 13, 12)
outcome <- gl(3, 1, 9)
treatment <- gl(3, 3)
glm.D93 <- glm(counts ~ outcome + treatment, family = poisson())
confint(glm.D93)
confint.default(glm.D93)

# A Poisson log-linear GLM logLikFn.glm S3 method is provided in profileCI
# so we do not need to supply loglik explicitly
prof <- profileCI(glm.D93)
prof

# Supplying a Poisson GLM log-likelihood explicitly
poisson_loglik <- function(pars) {
  lambda <- exp(model.matrix(glm.D93) %*% pars)
  loglik <- stats::dpois(x = glm.D93$y, lambda = lambda, log = TRUE)
  return(sum(loglik))
}
# This will be a bit slower than profile.glm() because glm.fit() is fast
prof <- profileCI(glm.D93, loglik = poisson_loglik)
prof
plot(prof, parm = 1)
plot(prof, parm = "outcome2")

# Supplying a more general Poisson GLM log-likelihood
poisson_loglik_2 <- function(pars, glm_object) {
  lambda <- exp(model.matrix(glm_object) %*% pars)
  loglik <- stats::dpois(x = glm_object$y, lambda = lambda, log = TRUE)
  return(sum(loglik))
}
prof <- profileCI(glm.D93, loglik = poisson_loglik_2, glm_object = glm.D93)
prof

## Nonlinear least squares, from example(nls)
DNase1 <- subset(DNase, Run == 1)
fm1DNase1 <- nls(density ~ SSlogis(log(conc), Asym, xmid, scal), DNase1)
confint(fm1DNase1)
# profileCI() gives slightly different results because confint.nls() is
# not based on profiling the log-likelihood but rather changes in the RSS
prof <- profileCI(fm1DNase1)
prof
```

---

profileCI_methods *Methods for objects of class* "profileCI"

Description

Methods for objects of class "profileCI" returned from profileCI.

Usage

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

## S3 method for class 'profileCI'
plot(x, parm = 1:nrow(x), add = TRUE, digits = 2, ...)
```

Arguments

x	An object inheriting from class profileCI", a result of a call to profileCI.
...	Further arguments. For print."profileCI" to pass arguments to print. For plot.profileCI to pass graphical parameters to graphics::plot to create the initial plot of the profile log-likelihood.
parm	A numeric or character scalar specifying the parameter for which a profile log-likelihood is plotted. Must be a single component consistent with the argument parm to profileCI.
add	A logical scalar. If add = TRUE then the plot is annotated with a horizontal line indicating the critical value for the profile log-likelihood used to calculate the confidence limits, vertical lines indicating the values of these limits and a legend stating the confidence interval.
digits	An integer. Passed to signif to round the confidence limits in the legend, if add = TRUE. The confidence level is hard-coded to be expressed to 3 significant figures.

Details

`print.profileCI`. A numeric matrix with 2 columns giving the lower and upper confidence limits for the parameters specified by the argument `parm` in `profileCI`. These columns are labelled as `(1-level)/2` and `1-(1-level)/2`, expressed as a percentage, by default `2.5%` and `97.5%`.

`plot.profileCI`. A plot is produced of the profile log-likelihood for the parameter chosen by `parm`. Only the parameter values used to profile the log-likelihood in the call to `profileCI` are included, so if `faster = TRUE` was used then the plot will not be of a smooth curve but will be triangular in the middle.

Value

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

`plot.profileCI`: a numeric vector containing the confidence interval for the parameter chosen for the plot.

Examples

See `profileCI`.

See Also

`profileCI`.

Index