Package: nlreg (via r-universe)

September 10, 2024

Version 1.2-2.2

Date 2019-01-30

Title Higher Order Inference for Nonlinear Heteroscedastic Models

Author S original by Alessandra R. Brazzale

<alessandra.brazzale@unipd.it> and Ruggero Bellio

<ruggero.bellio@uniud.it>. R port by Alessandra R. Brazzale

<alessandra.brazzale@unipd.it>, following earlier work by Douglas Bates.

Maintainer Alessandra R. Brazzale <alessandra.brazzale@unipd.it>

URL https://www.r-project.org, http://statwww.epfl.ch/AA/

Description Likelihood inference based on higher order approximations for nonlinear models with possibly non constant variance.

Depends R (>= 3.5.0), statmod, survival

Suggests boot, cond, csampling, marg

License GPL (>= 2) | file LICENCE

LazyLoad yes

LazyData yes

NeedsCompilation no

Date/Publication 2019-01-30 14:50:03 UTC

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

RemoteUrl https://github.com/cran/nlreg

RemoteRef HEAD

RemoteSha 4c8d7769cff46cf350f07e56e5bc6966c66e8d64

Contents

`nlreg-package` *Higher Order Inference for Nonlinear Heteroscedastic Models*

Description

Likelihood inference based on higher order approximations for nonlinear models with possibly non constant variance.

Details

The DESCRIPTION file:

Package:	nlreg
Version:	1.2-2.2
Date:	2019-01-30
Title:	Higher Order Inference for Nonlinear Heteroscedastic Models
Author:	S original by Alessandra R. Brazzale <alessandra.brazzale@unipd.it> and Ruggero Bellio <ruggero.bellio@un

Maintainer:	Alessandra R. Brazzale <alessandra.brazzale@unipd.it>
URL:	https://www.r-project.org, http://statwww.epfl.ch/AA/
Description:	Likelihood inference based on higher order approximations for nonlinear models with possibly non constant va
Depends:	R (>= 3.5.0), statmod, survival
Suggests:	boot, cond, csampling, marg
License:	GPL (>= 2) | file LICENCE
LazyLoad:	yes
LazyData:	yes

Index of help topics:

```
                        'all.profiles.nlreg' object
profile.nlreg           Profile Method for 'nlreg' Objects
ria                     Radioimmunoassay Data
summary.all.nlreg.profiles
                        Summary Method for Objects of Class
                        'all.nlreg.profiles'
summary.mpl             Summary Method for 'mpl' Objects
summary.nlreg           Summary Method for Nonlinear Heteroscedastic
                        Models
summary.nlreg.profile   Summary Method for Objects of Class
                        'nlreg.profile'
var2cor                 Convert Covariance Matrix to Correlation Matrix
                        - Generic Function
```

Likelihood inference based on higher order approximations for nonlinear models with possibly non constant variance

Package:	nlreg
Version:	1.2-0
Date:	2009-10-03
URL:	http://www.r-project.org, http://statwww.epfl.ch/AA/
Depends:	R (>= 2.6.0), statmod, survival
Suggests:	boot, cond, csampling, marg
License:	GPL (>= 2)
LazyLoad:	yes
LazyData:	yes

Index:

```
Functions:
=========
Dmean                   Differentiate the Mean Function of a Nonlinear
                        Model
Dvar                    Differentiate the Variance Function of a
                        Nonlinear Model
contour.all.nlreg.profiles
                        Contour Method for 'nlreg' Objects
expInfo                 Returns the Expected Information Matrix --
                        Generic Function
expInfo.nlreg           Expected Information Matrix for 'nlreg'
                        Objects
logLik.nlreg            Compute the Log Likelihood for Nonlinear
                        Heteroscedastic Models
mpl                     Maximum Adjusted Profile Likelihood Estimation
                        -- Generic Function
mpl.nlreg               Maximum Adjusted Profile Likelihood Estimates
                        for a 'nlreg' Object
mpl.object              Maximum Adjusted Profile Likelihood Object
```

```
nlreg                   Fit a Nonlinear Heteroscedastic Model via
                        Maximum Likelihood
nlreg.diag              Nonlinear Heteroscedastic Model Diagnostics
nlreg.object            Nonlinear Heteroscedastic Model Object
obsInfo                 Returns the Observed Information Matrix --
                        Generic Function
obsInfo.nlreg           Observed Information Matrix for 'nlreg'
                        Objects
param                   Extract All Parameters from a Model -- Generic
                        Function
plot.nlreg.contours     Use plot() on a 'nlreg.contours' object
plot.nlreg.diag         Diagnostic Plots for Nonlinear Heteroscedastic
                        Models
plot.nlreg.profile      Use plot() on a 'profile.nlreg' and
                        'all.profiles.nlreg' object
profile.nlreg           Profile Method for 'nlreg' Objects
summary.all.nlreg.profiles
                        Summary Method for Objects of Class
                        'all.nlreg.profiles'
summary.mpl             Summary Method for 'mpl' Objects
summary.nlreg           Summary Method for Nonlinear Heteroscedastic
                        Models
summary.nlreg.profile   Summary Method for Objects of Class
                        'nlreg.profile'
var2cor                 Convert Covariance Matrix to Correlation
                        Matrix -- Generic Function


Datasets:
========
C1                      Herbicide Data (Chlorsulfuron)
C2                      Herbicide Data (Chlorsulfuron)
C3                      Herbicide Data (Chlorsulfuron)
C4                      Herbicide Data (Chlorsulfuron)
M2                      Herbicide Data (Metsulfuron Methyl)
M4                      Herbicide Data (Metsulfuron Methyl)
chlorsulfuron           Chlorsulfuron Data
daphnia                 'Daphnia Magna' Data
helicopter              Paper Helicopter Data
metsulfuron             Metsulfuron Methyl Data
ria                     Radioimmunoassay Data
```

Further information is available in the following vignettes:

```
Rnews-paper    hoa: An R Package Bundle for Higher Order Likelihood Inference (source, pdf)
```

Author(s)

S original by Alessandra R. Brazzale <alessandra.brazzale@unipd.it> and Ruggero Bellio <ruggero.bellio@uniud.it>. R port by Alessandra R. Brazzale <alessandra.brazzale@unipd.it>, following earlier work by Douglas Bates.

Maintainer: Alessandra R. Brazzale <alessandra.brazzale@unipd.it>

References

Brazzale, A.R. (2005). hoa: An R package bundle for higher order likelihood inference. Rnews, 5/1, May 2005, 20-27. ISSN 609-3631. URL: https://www.r-project.org/doc/Rnews/Rnews_2005-1.pdf

Examples of applications, and generally of the use of likelihood asymptotics, are given in: Brazzale, A.R., Davison, A.C. and Reid, N. (2007). Applied Asymptotics: Case Studies in Small-Sample Statistics. Cambridge University Press, Cambridge. URL: http://statwww.epfl.ch/AA/

See Also

`cond` `marg` `csampling`

C1 *Six Herbicide Data Sets*

Description

The `C1`–`C4`, `M2` and `M4` data frames have 40 to 72 rows and three columns.

Six bioassay on the action of the herbicides chlorsulfuron and metsulfuron methyl on the callus area of colonies of *Brassica napus L.* The experiments consist of measurements for different dose levels and can be balanced (`C4`, `M2`) or unbalanced (`C1`, `C2`, `C3`, `M4`).

Usage

```
data(C1)
data(C2)
data(C3)
data(C4)
data(M2)
data(M4)
```

Format

These data frame contain the following columns:

`group` indicator variable for each tested dose;

`dose` the tested dose (nmol/l);

`area` the callus area (mm^2).

Note

Data sets C3 and chlorsulfuron are the same. Data sets M2 and metsulfuron are the same.

Source

The data were obtained from

Seiden, P., Kappel, D. and Streibig, J. C. (1998) Response of *Brassica napus L.* tissue culture to metsulfuron methyl and chlorsulfuron. *Weed Research*, **38**, 221–228.

References

Bellio, R., Jensen, J.E. and Seiden, P. (2000). Applications of likelihood asymptotics for nonlinear regression in herbicide bioassays. *Biometrics*, **56**, 1204–1212.

Brazzale, A. R. (2000) *Practical Small-Sample Parametric Inference*. Ph.D. Thesis N. 2230, Department of Mathematics, Swiss Federal Institute of Technology Lausanne. Section 5.3, Example 8.

See Also

chlorsulfuron, metsulfuron

Examples

```
data(C3)
attach(C3)
plot(dose, area, xlab = "tested dose (nmol/l)",
     ylab = "log callus area (mm^2)", log = "y")
detach()
```

---

chlorsulfuron *Chlorsulfuron Data*

Description

The chlorsulfuron data frame has 51 rows and 3 columns.

Bioassay on the action of the herbicide chlorsulfuron on the callus area of colonies of *Brassica napus L.* The experiment consists of 51 measurements for 10 different dose levels. The design is unbalanced: the number of replicates per dose varies from a minimum of 5 to a maximum of 8.

Usage

```
data(chlorsulfuron)
```

Format

This data frame contains the following columns:

`group` indicator variable for each tested dose;

`dose` the tested dose (nmol/l);

`area` the callus area (mm^2).

Source

The data were obtained from

Seiden, P., Kappel, D. and Streibig, J. C. (1998) Response of *Brassica napus L.* tissue culture to metsulfuron methyl and chlorsulfuron. *Weed Research*, **38**, 221–228. Dataset C3.

References

Bellio, R., Jensen, J.E. and Seiden, P. (2000). Applications of likelihood asymptotics for nonlinear regression in herbicide bioassays. *Biometrics*, **56**, 1204–1212.

Brazzale, A. R. (2000) *Practical Small-Sample Parametric Inference*. Ph.D. Thesis N. 2230, Department of Mathematics, Swiss Federal Institute of Technology Lausanne. Section 5.3, Example 8.

Examples

```
data(chlorsulfuron)
attach(chlorsulfuron)
plot(dose, area, xlab = "tested dose (nmol/l)",
     ylab = "log callus area (mm^2)", log = "y")
detach()
```

---

contour.all.nlreg.profiles

Contour Method for 'nlreg' Objects

Description

Draws the approximate bivariate contour plots for two or all parameters of a nonlinear heteroscedastic model and, on request, returns the list of elements used.

Usage

```
## S3 method for class 'all.nlreg.profiles'
contour(x, offset1, offset2, alpha = c(0.1, 0.05),
        stats = c("sk", "fr"), ret = FALSE, plotit = TRUE,
        drawlabels = FALSE, lwd1 = 1, lwd2 = 1, lty1 = "solid",
        lty2 = "solid", cl1 = "blue", cl2 = "red", col = "black",
        pch1 = 1, pch2 = 16, cex = 0.5, ...)
```

Arguments

x	an all.nlreg.profiles object, that is, the result of a call to profile.nlreg with offset = "all".
offset1, offset2	the two parameters to consider in the approximate bivariate contour plots.
alpha	a numerical vector defining the levels of the contours; the default is c(0.1, 0.05), that is, $1-\alpha = 0.9$ and $1-\alpha = 0.95$.
stats	character value indicating which higher order statistics to plot. Admissible values are "sk" for *Skovgaard's (1996)* proposal and "fr" for *Fraser, Reid and Wu's (1999)* approach. The default is "sk".
ret	logical value; if TRUE, a list containing the elements needed to draw the approximate contour plots is returned. Default is FALSE.
plotit	logical value indicating whether to draw the contours. Default is TRUE.
drawlabels	logical value. Contours are labelled if TRUE.
lwd1, lwd2	the line widths used to compare different curves in the same plot; default is lwd2 = 2 for higher order solutions and lwd1 = 1 for first order solutions.
lty1, lty2	line types used to compare different curves in the same plot; default is "solid" for all statistics.
cl1, cl2, col	colors used to compare different curves in the same plot; default is cl2 = "red" for higher order solutions, and cl1 = "blue" for the remaining first order statistics. The default color of the plot is col = "black".
pch1, pch2	character types used to compare different values in the same plot; default is pch2 = 16 for higher order solutions, and pch1 = 1 for the remaining first order statistics.
cex	the character expansions relative to the standard size of the device to be used for printing text. The default is cex = 0.5.
...	absorbs additional arguments such as graphics parameters.

Details

The function contour.all.nlreg.profiles calculates all elements needed to draw the profile and approximate bivariate contour plots for respectively two parameters of interest and all parameters in the model, depending on whether the offset1 and offset2 arguments are used.

Contour plots represent the bivariate extension of profile plots. Given two parameters of interst, they plot the corresponding joint confidence regions of levels $1-\alpha$ obtained from the likelihood ratio statistic and the Wald statistic (*Bates and Watts, 1988, Section 6.1.2*). The closer the two curves are, the more the likelihood surface is quadratic. Usually profile traces are added, that is, the curves showing the constrained maximum likelihood estimates of one parameter as a function of the other, as they provide useful information on how the estimates affect each other. If the asymptotic correlation is zero, the angle between the traces is close to $\pi/2$. The calculation of exact contour plots is computationally very intesive, as the model has to be refitted several times to obtain the constrained estimates. *Bates and Watts (1988, Appendix A.6)* present an approximate solution, which only requires the computation of the parameter profiles and which gives rise to the so-called profile pair sketches.

The function `contour.all.nlreg.profiles` extends the classical profile plots and profile pair sketches by including the higher order solutions r^* (*Barndorff-Nielsen, 1991*) and w^* (*Skovgaard, 2001*). The idea is to provide insight into the behaviour of first order methods such as detecting possible bias of the estimates or the influence of the model curvature. More precisely, the sample space derivatives in *Barndorff-Nielsens' (1991)* r^* statistic are replaced by respectively the approximations proposed in *Skovgaard (1996)* and *Fraser, Reid and Wu (1999)* depending on the value of the `stats` argument. The r^* statistic is used to calculate an approximation to *Skovgaard's (2001)* w^* statistic adopting the method by *Bates and Watts (1988, Appendix A.6)*. This method can break down, if the two parameter estimates are strongly correlated. The approximate contours of w^* are then missing in the corresponding panels; four bullets indicate where they intersect the profile traces.

All necessary quantities are retrieved from the `all.nlreg.profiles` object passed through the `x` argument. The `offset1` and `offset2` arguments can be used to specifiy two parameters of interest, in which case only the profile pair sketches for these two parameters are returned, one on the original scale and one on the normal scale. On the normal scale, the units do not express the parameter values themselves, but the associated likelihood root statistics. (See *Bates and Watts, 1988, Section 6.1.2*, for explanation.) If the `offset1` and `offset2` arguments are missing, profile plots and approximate contour plots are drawn for all model parameters. The plots are organized in form of a matrix. The main diagonal contains the profile plots. The approximate bivariate contour plots in the lower triangle are plotted on the original scale, whereas the ones in the upper triangle are on the r scale.

The theory and statistics used are summarized in *Brazzale (2000, Chapters 2 and 3)*. More details of the implementation are given in *Brazzale (2000, Section 6.3.2)*.

Value

If `ret = TRUE`, a list of class `nlreg.contours` is returned which contains the elements needed to draw the profiles and approximate bivariate contours for two or all parameters in a nonlinear heteroscedastic model. Otherwise, no value is returned.

Side Effects

If `plotit = TRUE`, a plot is produced on the current graphics device.

Note

`contour.all.nlreg.profiles` is a method for the generic function `contour` for class `all.nlreg.profiles`. It can be invoked by calling `contour` for an object of the appropriate class, or directly by calling `contour.all.nlreg.profiles`.

References

Barndorff-Nielsen, O. E. (1991) Modified signed log likelihood ratio. *Biometrika*, **78**, 557–564.

Bates, D. M. and Watts, D. G. (1988) *Nonlinear Regression Analysis and Its Applications*. New York: Wiley.

Brazzale, A. R. (2000) *Practical Small-Sample Parametric Inference*. Ph.D. Thesis N. 2230, Department of Mathematics, Swiss Federal Institute of Technology Lausanne.

Fraser, D.A.S., Reid, N. and Wu, J. (1999). A simple general formula for tail probabilities for frequentist and Bayesian inference. *Biometrika*, **86**, 249–264.

Skovgaard, I. M (1996) An explicit large-deviation approximation to one-parameter tests. *Bernoulli*, **2**, 145–165.

Skovgaard, I. M. (2001) Likelihood asymptotics. *Scandinavian Journal of Statistics*, **28**, 3–32.

See Also

nlreg.profile.objects, plot.nlreg.contours, contour

Examples

```
## Not run:
data(metsulfuron)
metsulfuron.nl <-
    nlreg( formula = log(area) ~ log( b1+(b2-b1) / (1+(dose/b4)^b3) ),
           weights = ~ ( 1+dose^exp(g) )^2, data = metsulfuron,
           start = c(b1 = 138, b2 = 2470, b3 = 2, b4 = 0.07, g = log(0.3)),
           hoa = TRUE )
##
metsulfuron.prof <- profile( metsulfuron.nl, trace = TRUE )
par( mai = rep(0.2, 4) )
contour( metsulfuron.prof )
## End(Not run)
```

---

daphnia *'Daphnia Magna' Data*

Description

The daphnia data frame has 136 rows and 2 columns.

Ecotoxicity study to assess the impact of the herbicide dinoseb on the survival of *Daphnia magna* Strauss, 1820, a micro-crustacean widely used as test organism in aquatic ecotoxicological assays. The design of the experiment includes 35 irregularly spaced concentrations ranging from 0.006 to 11.3 mg/l and a control group. The upper endpoint of 11.3 mg/l is the highest concentration at which the test substance is soluble in the test medium. The number of replicates per concentration varies from 1 to 11 experimental units. The survival time is measured in days.

Usage

```
data(daphnia)
```

Format

This data frame contains the following columns:

`conc` the tested concentration (mg/l);

`time` the survival time in days.

Source

The data were obtained from

Ch\'evre, N. (2000) *Etude et mod\'elisation des effets \'ecotoxiques d'un micropolluant organique sur* Daphnia magna *et* Pseudokirchneriella subcapitata (in French). Ph.D. Thesis N. 2117, Department of Rural Engineering, Swiss Federal Institute of Technology Lausanne.

References

Brazzale, A.R. (2000) *Practical Small-Sample Parametric Inference*. Ph.D. Thesis N. 2230, Department of Mathematics, Swiss Federal Institute of Technology Lausanne. Section 5.3, Example 5.

Ch\'evre, N., Becker-van Slooten, K., Tarradellas, J., Brazzale, A. R., Behra, R. and Guettinger, H. (2001) Effects of dinoseb on the entire life-cycle of *Daphnia magna*. Part II: Modelling of survival and proposal of an alternative to No-Observed-Effect-Concentration (NOEC). *Environmental Toxicology and Chemistry*, **21**, 828–833.

Examples

```
data(daphnia)
attach(daphnia)
plot(conc, time, xlab = "test concentration (mg/l)",
     ylab = "survival time (d)", log = "y")
detach()
```

Dmean *Differentiate the Mean Function of a Nonlinear Model*

Description

Calculates the gradient and Hessian of the mean function of a nonlinear heteroscedastic model.

Usage

```
Dmean(nlregObj, hessian = TRUE)
```

Arguments

nlregObj	a nonlinear heteroscedastic model fit as obtained from a call to nlreg.
hessian	logical value indicating whether the Hessian should be computed. The default is TRUE.

Details

The mean function is differentiated with respect to the regression coefficients as specified in the `coef` component of the `nlreg` object. The returned function definition, however, includes all parameters — regression coefficients and variance parameters — as arguments. When evaluated, it implicitly refers to the data to whom the model was fitted and which must be on the search list. The gradient and Hessian are calculated for each data point: the `gradient` attribute is a $n \times p$ matrix and the `hessian` attribute is a $n \times p \times p$ array, where n and p are respectively the number of data points and the number of regression coefficients.

Value

a function whose arguments are named according to the parameters of the nonlinear model `nlregObj`. When evaluated, it returns the value of the mean function along with attributes called `gradient` and `hessian`, the latter if requested. These are the gradient and Hessian of the mean function with respect to the regression coefficients.

Note

`Dmean` and `Dvar` are the two workhorse functions of the `nlreg` library. The details are given in *Brazzale (2000, Section 6.1.2).*

The symbolic differentiation algorithm is based upon the `D` function. As this algorithm is highly recursive, the `hessian = TRUE` argument should only be used if the Hessian matrix is needed. Whenever possible, derivatives should be stored so as to be re-used in further calculations. This is, for instance, achieved by the nonlinear heteroscedastic model fitting routine `nlreg` through the argument `hoa = TRUE`.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language: A Programming Environment for Data Analysis and Graphics*. London: Chapman \& Hall. Section 9.6.

Brazzale, A. R. (2000) *Practical Small-Sample Parametric Inference*. Ph.D. Thesis N. 2230, Department of Mathematics, Swiss Federal Institute of Technology Lausanne.

See Also

`Dvar`, `nlreg.object`, `deriv3`, `D`

Examples

```
library(boot)
data(calcium)
calcium.nl <- nlreg( cal ~ b0*(1-exp(-b1*time)),
                     start = c(b0 = 4, b1 = 0.1),  data = calcium )
Dmean( calcium.nl )
##function (b0, b1, logs)
##{
##    .expr3 <- exp(-b1 * time)
##    .expr4 <- 1 - .expr3
##    .expr6 <- .expr3 * time
```

```
##    .value <- b0 * .expr4
##    .grad <- array(0, c(length(.value), 2), list(NULL, c("b0",
##        "b1")))
##    .hessian <- array(0, c(length(.value), 2, 2), list(NULL,
##        c("b0", "b1"), c("b0", "b1")))
##    .grad[, "b0"] <- .expr4
##    .hessian[, "b0", "b0"] <- 0
##    .hessian[, "b0", "b1"] <- .hessian[, "b1", "b0"] <- .expr6
##    .grad[, "b1"] <- b0 * .expr6
##    .hessian[, "b1", "b1"] <- -(b0 * (.expr6 * time))
##    attr(.value, "gradient") <- .grad
##    attr(.value, "hessian") <- .hessian
##    .value
##}
##
param( calcium.nl )
##        b0         b1       logs
## 4.3093653  0.2084780 -1.2856765
##
attach( calcium )
calcium.md <- Dmean( calcium.nl )
attr( calcium.md( 4.31, 0.208, -1.29 ), "gradient" )
##              b0       b1
## [1,] 0.08935305 1.766200
## [2,] 0.08935305 1.766200
## [3,] 0.08935305 1.766200
## [4,] 0.23692580 4.275505
## \dots
detach()
```

Dvar *Differentiate the Variance Function of a Nonlinear Model*

Description

Calculates the gradient and Hessian of the variance function of a nonlinear heteroscedastic model.

Usage

```
Dvar(nlregObj, hessian = TRUE)
```

Arguments

nlregObj	a nonlinear heteroscedastic model fit as obtained from a call to nlreg.
hessian	logical value indicating whether the Hessian should be computed. The default is TRUE.

Details

The variance function is differentiated with respect to the variance parameters specified in the `varPar` component of the `nlregObj` object and, if the variance function depends on them, with respect to the regression coefficients specified in the `coef` component. The returned function definition includes all parameters. When evaluated, it implicitly refers to the data to whom the `nlreg` object was fitted and which must be on the search list. The gradient and Hessian are calculated for each data point: the `gradient` attribute is a $n \times p$ matrix, and the `hessian` attribute is a $n \times p \times p$ array, where n and p are respectively the number of data points and the number of regression coefficients.

Value

a function whose arguments are named according to the parameters of the nonlinear model `nlregObj`. When evaluated, it returns the value of the variance function along with attributes called `gradient` and `hessian`, the latter if requested. These are the gradient and Hessian of the variance function with respect to the model parameters.

Note

`Dmean` and `Dvar` are the two workhorse functions of the `nlreg` library. The details are given in *Brazzale (2000, Section 6.1.2)*.

The symbolic differentiation algorithm is based upon the `D` function. As this algorithm is highly recursive, the `hessian = TRUE` argument should only be used if the Hessian matrix is needed. Whenever possible, derivatives should be stored so as to be re-used in further calculations. This is, for instance, achieved for the nonlinear heteroscedastic model fitting routine `nlreg` through the argument `hoa = TRUE`.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language: A Programming Environment for Data Analysis and Graphics*. London: Chapman \& Hall. Section 9.6.

Brazzale, A. R. (2000) *Practical Small-Sample Parametric Inference*. Ph.D. Thesis N. 2230, Department of Mathematics, Swiss Federal Institute of Technology Lausanne.

See Also

`Dmean`, `nlreg.object`, `deriv3`, `D`

Examples

```
library(boot)
data(calcium)
calcium.nl <- nlreg( cal ~ b0*(1-exp(-b1*time)),
                     start = c(b0 = 4, b1 = 0.1), data = calcium )
Dvar( calcium.nl )
##function (b0, b1, logs)
##{
##    .expr1 <- exp(logs)
##    .value <- .expr1
```

```
##    .grad <- array(0, c(length(.value), 1), list(NULL, c("logs")))
##    .hessian <- array(0, c(length(.value), 1, 1), list(NULL,
##        c("logs"), c("logs")))
##    .grad[, "logs"] <- .expr1
##    .hessian[, "logs", "logs"] <- .expr1
##    attr(.value, "gradient") <- .grad
##    attr(.value, "hessian") <- .hessian
##    .value
##}
##
attach( calcium )
calcium.vd <- Dvar( calcium.nl )
param( calcium.nl )
##        b0         b1       logs
## 4.3093653  0.2084780 -1.2856765
##
attr( calcium.vd( 4.31, 0.208, -1.29 ), "gradient" )
##          logs
##[1,] 0.2752708
##
calcium.nl <- update( calcium.nl, weights = ~ ( 1+time^g )^2,
                      start = c(b0 = 4, b1 = 0.1, g = 1))
Dvar( calcium.nl )
##function (b0, b1, g, logs)
##{
##    .expr1 <- time^g
##    .expr2 <- 1 + .expr1
##    .expr4 <- exp(logs)
##    .expr5 <- .expr2^2 * .expr4
##    .expr6 <- log(time)
##    .expr7 <- .expr1 * .expr6
##    .expr10 <- 2 * (.expr7 * .expr2) * .expr4
##    .value <- .expr5
##    .grad <- array(0, c(length(.value), 2), list(NULL, c("g",
##        "logs")))
##    .hessian <- array(0, c(length(.value), 2, 2), list(NULL,
##        c("g", "logs"), c("g", "logs")))
##    .grad[, "g"] <- .expr10
##    .hessian[, "g", "g"] <- 2 * (.expr7 * .expr6 * .expr2 + .expr7 *
##        .expr7) * .expr4
##    .hessian[, "g", "logs"] <- .hessian[, "logs", "g"] <- .expr10
##    .grad[, "logs"] <- .expr5
##    .hessian[, "logs", "logs"] <- .expr5
##    attr(.value, "gradient") <- .grad
##    attr(.value, "hessian") <- .hessian
##    .value
##}
##
calcium.vd <- Dvar( calcium.nl )
param( calcium.nl )
##        b0         b1          g       logs
## 4.3160408  0.2075937  0.3300134 -3.3447585
##
```

```
attr( calcium.vd(4.32, 0.208, 0.600, -2.66 ), "gradient" )
##                g      logs
## [1,] -0.11203422 0.1834220
## [2,] -0.11203422 0.1834220
## [3,] -0.11203422 0.1834220
## [4,]  0.09324687 0.3295266
## \dots
##
detach()
```

---

expInfo *Returns the Expected Information Matrix — Generic Function*

Description

Returns the expected information matrix from a fitted model object.

Usage

```
expInfo(object, ...)
```

Arguments

object	any fitted model object for which the observed information can be calculated.
...	absorbs any additional argument.

Details

This function is generic (see methods); method functions can be written to handle specific classes of data. Classes which already have methods for this function include: nlreg.

Value

the expected information matrix for a fitted regression model.

See Also

expInfo.nlreg, nlreg.object, obsInfo

Examples

```
data(metsulfuron)
metsulfuron.nl <-
    nlreg( log(area) ~ log( b1+(b2-b1) / (1+(dose/b4)^b3) ),
           weights = ~ ( 1+dose^exp(g) )^2, data = metsulfuron,
           start = c(b1 = 138, b2 = 2470, b3 = 2, b4 = 0.07, g = log(0.3)),
           hoa = TRUE)
expInfo( metsulfuron.nl )
```

expInfo.nlreg *Expected Information Matrix for 'nlreg' Objects*

Description

Returns the expected information matrix for a fitted nlreg model.

Usage

```
## S3 method for class 'nlreg'
expInfo(object, par, mu, v, m1 = NULL, v1 = NULL, ...)
```

Arguments

object	a fitted nlreg object such as returned by a call to nlreg.
par	a vector of parameter values where each element is named after the parameter it represents. If missing, the values in the ws$allPar component of object are used.
mu	numerical vector containing the mean function evaluated at each data point. If missing, the fitted values saved in object are used.
v	numerical vector containing the variance function evaluated at each data point. If missing, the values of the weights component of object are used.
m1	a matrix whose rows represent the gradients of the mean function evaluated at each data point. If NULL, the gradient attribute of the object returned by a call to Dmean is used.
v1	a matrix whose rows represent the gradient of the variance function evaluated at each data point. If NULL, the gradient attribute of the object returned by a call to Dvar is used.
...	absorbs any additional argument.

Details

This function is a method for the generic function expInfo for objects inheriting from class nlreg.

Value

the expected information matrix of the fitted nonlinear model passed through the object argument.

Note

This function is mostly intended for internal use. It is called by functions such as nlreg.diag, summary.nlreg and profile.nlreg. To extract the expected information matrix from a fitted nlreg object, the generic method expInfo should be used.

See Also

expInfo, nlreg.object, obsInfo

`helicopter` *Helicopter Data*

Description

The `helicopter` data frame has 9 rows and 6 columns.

Experimental design for studying the influences of the factors wing length and wing width on a paper helicopter's flight time. The goal is to find the factor setting that maximizes flight time when the paper helicopter is dropped from a fixed height of 15.5 feet

Usage

```
data(helicopter)
```

Format

A data frame with 9 observations on the following 6 variables:

`L` wing length in inches;

`W` wing width in inches;

`B` base length (always set to 3in);

`H` base height (always set to 2in);

`Order` run order;

`Time` flight time in seconds.

Source

The data were obtained from

Annis, D. H. (2006) Rethinking the paper helicopter: Combining statistical and engineering knowledge. *The American Statistician*, **59**, 320–326.

References

Box, G. E. P. (1992) Teaching engineers experimental design with a paper helicopter. *Quality Engineering*, **4**, 453–459.

Examples

```
data(helicopter)
##
## fit model (5) of Annis (2005)
## -----------------------------
heli <- helicopter
##
heli$LW <- heli$L * heli$W
heli$S <- heli$B * heli$H + ( 2 * heli$L + 1 ) * heli$W
```

```
heli$logTime <- log( heli$Time )
heli$Y <- heli$logTime + log( heli$S ) / 2
#
heli.nlreg <- nlreg( Y ~ b0 + b1 * log( b2^2 / LW + LW ), data = heli,
                     start = c( b0 = 6, b1 = -1, b2 = 20 ) )
```

`logLik.nlreg` *Compute the Log Likelihood for Nonlinear Heteroscedastic Models*

Description

Computes the log likelihood for a nonlinear model with possibly non constant variance.

Usage

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

Arguments

`object`	an object inheriting from class `nlreg` representing a fitted nonlinear heteroscedastic model.
`...`	absorbs any additional argument.

Details

This is a method for the function `logLik()` for objects inheriting from class `nlreg`.

Value

Returns an object class `logLik` which is a number with attributes `nobs`, `npar` and `df` giving respectively the number of observations, the number of parameters (regression coefficients plus variance parameters) and the degrees of freedom in the model.

Note

The default `print` method for `logLik` objects is used.

See Also

`rsm.object`, `logLik`

Examples

```
library(boot)
data(calcium)
calcium.nl <- nlreg( cal ~ b0*(1-exp(-b1*time)),
                     start = c(b0 = 4, b1 = 0.1), data = calcium )
logLik( calcium.nl )
##
data(metsulfuron)
metsulfuron.nl <-
    nlreg( log(area) ~ log( b1+(b2-b1) / (1+(dose/b4)^b3) ),
           weights = ~ ( 1+dose^exp(g) )^2, data = metsulfuron,
           start = c(b1 = 138, b2 = 2470, b3 = 2, b4 = 0.07, g = log(0.3)),
           hoa = TRUE )
logLik( metsulfuron.nl )
```

`metsulfuron` *Metsulfuron Methyl Data*

Description

The `metsulfuron` data frame has 40 rows and 3 columns.

Bioassay on the action of metsulfuron methyl, a sulfunylurea herbicide, on a tissue culture of *Brassica napus L.* The experiment consists of 8 doses and 5 replications at each level.

Usage

```
data(metsulfuron)
```

Format

This data frame contains the following columns:

`group` indicator variable for each tested dose;

`dose` the tested dose (nmol/l);

`area` the callus area (mm^2).

Source

The data were obtained from

Seiden, P., Kappel, D. and Streibig, J. C. (1998) Response of *Brassica napus L.* tissue culture to metsulfuron methyl and chlorsulfuron. *Weed Research*, **38**, 221–228. Dataset M2.

References

Bellio, R., Jensen, J.E. and Seiden, P. (2000). Applications of likelihood asymptotics for nonlinear regression in herbicide bioassays. *Biometrics*, **56**, 1204–1212.

Brazzale, A. R. (2000) *Practical Small-Sample Parametric Inference*. Ph.D. Thesis N. 2230, Department of Mathematics, Swiss Federal Institute of Technology Lausanne. Section 5.3, Example 7.

Examples

```
data(metsulfuron)
attach(metsulfuron)
plot(dose, area, xlab = "tested dose (nmol/l)",
     ylab = "log callus area (mm^2)", log = "y")
detach()
```

`mpl` *Maximum Adjusted Profile Likelihood Estimation — Generic Function*

Description

Calculates the maximum adjusted profile likelihood estimates.

Usage

```
mpl(fitted, ...)
```

Arguments

`fitted`	any fitted model object for which the maximum adjusted profile likelihood estimates can be calculated.
`...`	absorbs any additional argument.

Details

This function is generic (see `methods`); method functions can be written to handle specific classes of data. Classes which already have methods for this function include: `nlreg`.

Value

the maximum adjusted profile likelihood estimates for all parameters of a regression model or for a subset of them.

See Also

`mpl.nlreg`, `nlreg.object`, `methods`

Examples

```
data(metsulfuron)
metsulfuron.nl <-
    nlreg( formula = log(area) ~ log( b1+(b2-b1) / (1+(dose/b4)^b3) ),
           weights = ~ ( 1+dose^exp(g) )^2, data = metsulfuron, hoa = TRUE,
           start = c(b1 = 138, b2 = 2470, b3 = 2, b4 = 0.07, g = log(0.3)) )
mpl( metsulfuron.nl, trace = TRUE )
##
options( object.size = 10000000 )
```

```
data(chlorsulfuron)
chlorsulfuron.nl <-
    nlreg( log(area) ~ log( b1+(b2-b1) / (1+(dose/b4)^b3) ),
        weights = ~ ( 1+k*dose^g*(b2-b1)^2/(1+(dose/b4)^b3)^4*b3^2*dose^(2*b3-2)/
                    b4^(2*b3)/(b1+(b2-b1)/(1+(dose/b4)^b3))^2 ),
        start = c(b1 = 2.2, b2 = 1700, b3 = 2.8, b4 = 0.28, g = 2.7, k = 1),
        data = chlorsulfuron, hoa = TRUE, trace = TRUE,
        control = list(x.tol = 10^-12, rel.tol = 10^-12, step.min = 10^-12) )
mpl( chlorsulfuron.nl, trace = TRUE )
```

`mpl.nlreg` *Maximum Adjusted Profile Likelihood Estimates for a 'nlreg' Object*

Description

Calculates the maximum adjusted profile likelihood estimates of the variance parameters for a nonlinear heteroscedastic model.

Usage

```
## S3 method for class 'nlreg'
mpl(fitted, offset = NULL, stats = c("sk", "fr"),
    control = list(x.tol = 1e-6, rel.tol = 1e-6, step.min = 1/2048,
    maxit = 100), trace = FALSE, ... )
```

Arguments

`fitted`	a `nlreg` object, that is, the result of a call to `nlreg` with non-constant variance function.
`offset`	a numerical vector whose elements are named after the variance parameters appearing in the nonlinear model. These will be fixed to the values specified in `offset`. The name `logs` is used to identify the constant term $\log(\sigma^2)$ which is included by default in the variance function (see the `weights` argument in `nlreg`). The default is `NULL`.
`stats`	character value indicating which correction term to use. Admissible values are `"sk"` for *Skovgaard's (1996)* proposal and `"fr"` for *Fraser, Reid and Wu's (1999)* approach. The default is `"sk"`.
`control`	a list of iteration and algorithmic constants. See the **Details** section below for their definition.
`trace`	logical flag. If `TRUE`, details of the iterations are printed. Default is `FALSE`.
`...`	absorbs any additional argument.

Details

The `mpl.nlreg` routine returns nearly unbiased estimates of the variance parameters of a nonlinear heteroscedastic regression model by maximizing the corresponding adjusted profile likelihood (*Barndorff-Nielsen, 1983*). More precisely, it implements two approximations derived from the theories developed respectively by *Skovgaard (1996)* and *Fraser, Reid and Wu (1999)*. The core algorithm alternates minimization of minus the adjusted profile log likelihood with respect to the variance parameters, and minimization of minus the profile log likelihood with respect to the regression coefficients. The first step is omitted if the `offset` argument is used in which case `mpl.nlreg` returns the constrained maximum likelihood estimates of the regression coefficients. The quasi-Newton optimizer `optim` is used in both steps. Starting values are retrieved from the `nlreg` object passed through the `fitted` argument.

The algorithm iterates until convergence or until the maximum number of iterations is reached. The stopping rule considers the relative difference between successive estimates of the variance parameters and the relative increment of the adjusted profile log likelihood. These are governed by the parameters `x.tol` and `rel.tol`/`step.min`, respectively. If the `offset` argument is used, the relative difference between successive estimates of the regression coefficients and the relative increment of the profile log likelihood are considered instead. If convergence has been reached, the results are saved in an object of class `mpl`. The output can be examined by `print` and `summary`. Components can be extracted using `coef` and `param`.

The theory is outlined in *Brazzale (2000, Sections 3.1 and 3.2.3)*. Details of the implementation are given in *Brazzale (2000, Section 6.3.1)*.

Value

an object of class `mpl` which inherits from `nlreg`. See `mpl.object` for details.

Side Effects

If `trace = TRUE` and `offset = NULL`, the iteration number and the corresponding adjusted profile log likelihood are printed.

Note

The argument `control` which controls the convergence criteria plays an important role. Fine-tuning of this argument helps surrounding a well-known problem in nonlinear regression, that is, convergence failure in cases where the likelihood and/or the adjusted profile likelihood are very flat.

References

Barndorff-Nielsen, O. E. (1983) On a formula for the distribution of the maximum likelihood estimator. *Biometrika*, **70**, 343–365.

Brazzale, A. R. (2000) *Practical Small-Sample Parametric Inference*. Ph.D. Thesis N. 2230, Department of Mathematics, Swiss Federal Institute of Technology Lausanne.

Fraser, D.A.S., Reid, N. and Wu, J. (1999). A simple general formula for tail probabilities for frequentist and Bayesian inference. *Biometrika*, **86**, 249–264.

Skovgaard, I. (1996) An explicit large-deviation approximation to one-parameter tests. *Bernoulli*, **2**, 145–165.

See Also

mpl, mpl.object, nlreg.object, optim

Examples

```
data(metsulfuron)
metsulfuron.nl <-
    nlreg( formula = log(area) ~ log( b1+(b2-b1) / (1+(dose/b4)^b3) ),
           weights = ~ ( 1+dose^exp(g) )^2, data = metsulfuron, hoa = TRUE,
          start = c(b1 = 138, b2 = 2470, b3 = 2, b4 = 0.07, g = log(0.3)) )
##
## MMPLE of the variance parameters
##
metsulfuron.mpl <- mpl( metsulfuron.nl, trace = TRUE )
summary( metsulfuron.mpl, corr = FALSE )
##
## constrained MLEs of the regression coefficients
##
metsulfuron.mpl <- mpl( metsulfuron.nl, offset = metsulfuron.nl$varPar,
                        trace = TRUE )
summary( metsulfuron.mpl, corr = FALSE )
```

---

mpl.object *Maximum Adjusted Profile Likelihood Object*

Description

Class of objects returned when calculating the maximum adjusted profile likelihood estimates of the variance parameters of a nonlinear heteroscedastic model.

Arguments

The following components must be included in a mpl object:

varPar	the maximum adjusted profile likelihood estimates of the variance parameters.
coefficients	the constrained MLEs of the regression coefficients given the maximum adjusted profile likelihood estimates of the variance parameters.
offset	the values passed through the offset argument in the call to mpl.nlreg that generated the mpl object and to which the variance parameters were fixed.
varParMLE	the MLEs of the variance parameters.
coefMLE	the MLEs of the regression coefficients.
varParCov	the (asymptotic) covariance matrix of the variance parameters, that is, the corresponding block in the inverse of the observed information matrix.
coefCov	the (asymptotic) covariance matrix of the regression coefficients, that is, the corresponding block in the inverse of the observed information matrix.
lmp	the adjusted profile log likelihood from the fit.

`lp` the profile log likelihood from the fit.

`stats` the indicator of which higher order solution was used.

`formula` the model formula.

`meanFun` the formula expression of the mean function.

`varFun` the formula expression of the variance function.

`data` a list representing a summary of the original data with the following components.

`'offset name'` the predictor variable with no duplicated value.

`repl` the number of replicates available for each value of the predictor.

`dupl` a vector of the same length than the predictor variable indicating the position of each data point in the *offset name* component.

`t1` the sum of the reponses for each design point in the *offset name* component.

`t2` the sum of the squared responses for each design point in the *offset name* component.

`nobs` the number of observations.

`iter` the number of interations needed for convergence; only if `offset` is not `NULL`.

`call` an image of the call to `mpl.nlreg`, but with all the arguments explicitly named.

`ws` a list containing information that is used in subsequent calculations, that is:

`allPar` the MLEs of all parameters.

`homVar` a logical value indicating whether the variance function is constant.

`xVar` a logical value indicating whether the variance function depends on the predictor variable.

`hoa` the value of the `hoa` argument in the call that generated the `nlreg` object passed through the `fitted` argument.

`missingData` a logical value indicating whether the `data` argument was missing in the call that generated the `nlreg` object passed through the `fitted` argument.

`frame` the name of the data frame if specified in the call to `nlreg` that generated the `fitted` argument.

`iter` the number of iteration required until convergence (only for non constant variance function).

`md` a function definition that returns the first two derivatives of the mean function if `hoa = TRUE` in the function call that generated the `nlreg` object passed through the `fitted` argument.

`vd` a function definition that returns the first two derivatives of the variance function if `hoa = TRUE` in the function call that generated the `nlreg` object passed through the `fitted` argument.

Generation

This class of objects is returned by the `mpl.nlreg` function. Class `mpl` inherits from class `nlreg`.

Methods

Objects of this class have methods for the functions `print`, `summary`, `coef` and `param`.

Note

The coefficients and variance parameters should be extracted by the generic functions of the same name, rather than by the $ operator.

The data and ws components are not intended to be directly used by users, but rather contain information used by functions such as summary.

See Also

mpl.nlreg, mpl, nlreg.object

---

nlreg *Fit a Nonlinear Heteroscedastic Model via Maximum Likelihood*

Description

Returns an object of class nlreg which represents a nonlinear heteroscedastic model fit of the data obtained by maximizing the corresponding likelihood function.

Usage

```
nlreg(formula, weights = NULL, data = sys.frame(sys.parent()), start,
      offset = NULL, subset = NULL,
      control = list(x.tol = 1e-06, rel.tol = 1e-06,
                     step.min = 1/2048, maxit = 100), trace = FALSE,
      hoa = FALSE)
```

Arguments

formula	a formula expression as for other nonlinear regression models, of the form response ~ f(predictor) where f is a nonlinear function of the predictor involving a number of regression coefficients. Only one predictor variable can be included in the model formula. Missing values are not allowed.
weights	a formula expression of the form ~ V(predictor) where V is a nonlinear variance function involving the predictor or some transformation of it, variance parameters and/or regression coefficients. The error variance nlreg works with is Var(error) = s^2 V(predictor) where the constant term σ^2 is included by default and must not be specified in the weights argument. The nlreg routine treats it on the logarithmic scale and assigns to it the parameter name logs. By default, the error variance is assumed to be constant.
data	an optional data frame in which to interpret the variables occurring in the model formula. Missing values are not allowed.

start	a numerical vector containing the starting values that initialize the iterative estimating procedure. Each component of the vector must be named and represents one of the parameters included in the mean and, if defined, variance function. Starting values have to be supplied for every model parameter, except for the constant term in the variance function which is included by default in the model. See the weights argument above.
offset	a numerical vector with a single named element. The name indicates the parameter of interest which will be fixed to the value specified. logs is used to identify the constant term σ^2 which is included by default in the variance function.
subset	expression saying which subset of the rows of the data should be used in the fit. This can be a logical vector or a numeric vector indicating which observation numbers are to be included. All observations are included by default.
control	a list of iteration and algorithmic constants. See the **Details** section below for their definition.
trace	logical flag. If TRUE, details of the iterations are printed.
hoa	logical flag. If TRUE, the first and second derivatives of the mean and, if defined, variance functions are stored in the fitted model object. The default is FALSE.

Details

A nonlinear heteroscedastic model representing the relationship between two scalar quantities is fitted. The response is specified on the left-hand side of the formula argument. The predictor appears in the right-hand side of the formula and, if specified, weights arguments. Only one predictor variable can be included. Missing values in the data are not allowed.

The fitting criterion is maximum likelihood. The core algorithm implemented in nlreg alternates minimization of minus the log likelihood with respect to the regression coefficients and the variance parameters. The quasi-Newton optimizer optim is used in both steps. The constant term σ^2 in Var(error) = s^2 V(predictor) is included by default. In order to work with a real value, σ^2 is estimated on the logarithmic scale, that is, the model is reparametrized into $\log(\sigma^2)$ which gives rise to the parameter name logs. If the errors are homoscedastic, the second step is omitted and the algorithm switches automatically to nls. If the weights argument is omitted, homoscedasticity of the errors is assumed.

Starting values for all parameters have to be passed through the start argument except for σ^2 for which the maximum likelihood estimate is available in closed form. Starting values should be chosen carefully in order to avoid convergence to a local maximum.

The algorithm iterates until convergence or until the maximum number of iterations defined by maxit is reached. The stopping rule considers the relative difference between successive estimates and the relative increment of the log likelihood. These are governed by the parameters x.tol and rel.tol/step.min, respectively.

If convergence has been reached, the results are saved in an object of class nlreg. The output can be examined by print and summary. Components can be extracted using coef, param, fitted and residuals. The model fit can be updated using update. Profile plots and profile pair sketches are provided by profile, and contour. Diagnostic plots are obtained from nlreg.diag.plots or simply plot.

The details are given in *Brazzale (2000, Section 6.3.1)*.

Value

An object of class `nlreg` is returned which inherits from `nls`. See `nlreg.object` for details.

Side Effects

If `trace = TRUE`, the iteration number and the corresponding log likelihood are printed.

Note

The arguments `hoa` and `control` play an important role. The first forces the algorithm to save the derivatives of the mean and variance functions in the fitted model object. This is imperative if one wants to save execution time, especially for complex models. Fine-tuning of the `control` argument which controls the convergence criteria helps surrounding a well-known problem in nonlinear regression, that is, convergence failure in cases where the likelihood is very flat.

If the errors are homoscedastic, the `nlreg` routine switches automatically to `nls` which, although rarely, dumps because of convergence problems. To avoid this, either reparametrize the model (see *Bates and Watts, 1988*) or choose starting values that are more realistic. This advice also holds in case of convergence problems for models with non constant variance function. Use the `trace = TRUE` argument to gain insight into what goes on at the different iteration steps.

The `weights` argument has a different meaning than in other model fitting routines such as `lm` and `glm`. It defines the variance function of the nonlinear model and not a vector of observation weights that are multiplied into the squared residuals.

References

Bates, D. M. and Watts, D. G. (1988) *Nonlinear Regression Analysis and Its Applications*. New York: Wiley.

Brazzale, A. R. (2000) *Practical Small-Sample Parametric Inference*. Ph.D. Thesis N. 2230, Department of Mathematics, Swiss Federal Institute of Technology Lausanne.

Seber, G. A. F. and Wild, C. J. (1989) *Nonlinear Regression*. New York: Wiley.

See Also

`nlreg.object`, `nls`

Examples

```
library(boot)
data(calcium)
##
## Homoscedastic model fit
calcium.nl <- nlreg( cal ~ b0*(1-exp(-b1*time)), start = c(b0 = 4, b1 = 0.1),
                     data = calcium )
##
## Heteroscedastic model fit
calcium.nl <- nlreg( cal ~ b0*(1-exp(-b1*time)), weights = ~ ( 1+time^g )^2,
                     start = c(b0 = 4, b1 = 0.1, g = 1), data = calcium,
                     hoa = TRUE)
## or
```

```
calcium.nl <- update(calcium.nl, weights = ~ (1+time^g)^2,
                     start = c(b0 = 4, b1 = 0.1, g = 1), hoa = TRUE )
##
##
## Power-of-X (POX) variance function
##
data(metsulfuron)
metsulfuron.nl <-
    nlreg( log(area) ~ log( b1+(b2-b1) / (1+(dose/b4)^b3) ),
           weights = ~ ( 1+dose^exp(g) )^2, data = metsulfuron,
           start = c(b1 = 138, b2 = 2470, b3 = 2, b4 = 0.07, g = log(0.3)),
           hoa = TRUE )
##
##
## Power-of-mean (POM) variance function
##
data(ria)
ria.nl <- nlreg( count ~ b1+(b2-b1) / (1+(conc/b4)^b3),
                 weights = ~ ( b1+(b2-b1) / (1+(conc/b4)^b3) )^g, data = ria,
                 start = c(b1 = 1.6, b2 = 20, b3 = 2, b4 = 300, g = 2),
                 hoa = TRUE, trace = TRUE )
##
##
## Error-in-variables (EIV) variance function
##
data(chlorsulfuron)
options( object.size = 10000000 )
chlorsulfuron.nl <-
    nlreg( log(area) ~ log( b1+(b2-b1) / (1+(dose/b4)^b3) ),
        weights = ~ ( 1+k*dose^g*(b2-b1)^2/(1+(dose/b4)^b3)^4*b3^2*dose^(2*b3-2)/
                    b4^(2*b3)/(b1+(b2-b1)/(1+(dose/b4)^b3))^2 ),
        start = c(b1 = 2.2, b2 = 1700, b3 = 2.8, b4 = 0.28, g = 2.7, k = 1),
        data = chlorsulfuron, hoa = TRUE, trace = TRUE,
        control = list(x.tol = 10^-12, rel.tol = 10^-12, step.min = 10^-12) )
```

`nlreg.diag` *Nonlinear Heteroscedastic Model Diagnostics*

Description

Calculates different types of residuals, influence measures and leverages for a nonlinear heteroscedastic model.

Usage

```
nlreg.diag(fitted, hoa = TRUE, infl = TRUE, trace = FALSE)
```

Arguments

fitted	a nlreg object, that is, the result of a call to nlreg.
hoa	logical value indicating whether higher order asymptotics should be used for calculating the regression diagnostics. Default is TRUE.
infl	logical value indicating whether influence measures should be calculated on the basis of a leave-one-out analysis. Default is TRUE.
trace	logical value. If TRUE, details of the iterations are printed. Default is FALSE.

Details

The regression diagnostics implemented in the nlreg.diag routine follow two approaches. The first exploits, where possible, the analogy with linear models, that is, it applies the classical definitions of residuals, leverages and Cook's distance after having linearized the nonlinear model through Taylor series expansion (*Carroll and Ruppert, 1988, Section 2.8*). The second approach uses the mean shift outlier model (*Cook and Weisberg, 1982, Section 2.2.2*), where a dummy variable is included for each observation at a time, the model refitted and the significance of the corresponding coefficient assessed.

The leverages are defined in analogy to the linear case (*Brazzale, 2000, Appendix A.2.2*). Two versions are available. In the first case the sub-block of the inverse of the expected information matrix corresponding to the regression coefficients is used in the definition. In the second case, this matrix is replaced by the inverse of $M'WM$, where M is the $n \times p$ matrix whose ith row is the gradient of the mean function evaluated at the ith data point and W is a diagonal matrix whose elements are the inverses of the variance function evaluated at each data point.

If the model is correctly specified, all residuals follow the standard normal distribution. The second kind of leverages described above are used to calculate the approximate studentized residuals, whereas the generalized Pearson residuals use the first kind. The ith generalized Pearson residual can also be obtained as the score statistic for testing the significance of the dummy coefficient in the mean shift outlier model for observation i. Accordingly, the ith deletion and r^*-type residuals are defined as respectively the likelihood root and modified likelihood root statistics (r and r^*) for the same situation (*Bellio, 2000, Section 2.6.1*).

Different influence measures were implemented in nlreg.diag. If infl = TRUE, the global measure (*Cook and Weisberg, 1982, Section 5.2*) and two partial ones (*Bellio, 2000, Section 2.6.2*), the first measuring the influence of each observation on the regression coefficients and the second on the variance parameters, are returned. They are calculated through a leave-one-out analysis, where one observation at a time is deleted and the model refitted. In order to avoid a further model fit, the constrained maximum likelihood estimates that would be needed are approximated by means of a Taylor series expansion centered at the MLEs. If infl = FALSE, only an approximation to Cook's distance, obtained from a first order Taylor series expansion of the partial influence measure for the regression coefficients, is returned.

A detailed account of regression diagnostics can be found in *Davison and Snell (1991)* and *Davison and Tsai (1992)*. The details and in particular the definitions of the above residuals and diagnostics are given in *Brazzale (2000, Section 6.3.1 and Appendix A.2.2)*.

Value

Returns an object of class nlreg.diag with the following components:

`fitted`	the fitted values, that is, the mean function evaluated at each data point.
`resid`	the response (or standardized) residuals from the fit.
`rp`	the generalized Pearson residuals from the fit.
`rs`	the approximate studentized residuals from the fit.
`rj`	the deletion residuals from the fit; only if `hoa = TRUE`.
`rsj`	the r^*-type residuals from the fit; only if `hoa = TRUE`.
`h`	the leverages of the observations.
`ha`	the approximate leverages of the observations.
`cook`	an approximation to Cook's distance for the regression coefficients.
`ld`	the global influence of each observation; only for heteroscedastic errors and if `infl = TRUE`.
`ld.rc`	the partial influence of each observation on the estimates of the regression coefficients; only for heteroscedastic errors and if `infl = TRUE`.
`ld.vp`	the partial influence of each observation on the estimates of the variance parameters; only for heteroscedastic errors and if `infl = TRUE`.
`npar`	the number of regression coefficients.

Side Effects

If `trace = TRUE`, the number of the observation currently considered in the mean shift outlier model or omitted in the leave-one-out analysis (see **Details** section above) is printed; only if `hoa = TRUE` or `infl = TRUE`.

Acknowledgments

This function is based on A. J. Canty's function `glm.diag` contained in library `boot`.

Note

The calculation of the deletion and r^*-type residuals and of the influence measures can be time-consuming. In the first case, the mean shift outlier model has to be refitted as many times as the total number of observations. In the second case, the original model is refitted the same amount of times, where one observation at a time is deleted. Furthermore, the definition of the r^*-type residuals requires differentiation of the mean function of the mean shift outlier model. These calculations can be avoided by changing the default setting of the arguments `hoa` and `infl` to `FALSE`.

To obtain some of the regression diagnostics (typically those based on higher order statistics), the model is repeatedly refitted for different values of the mean shift outlier model parameter. Although rarely, convergence problems may occur as the starting values are chosen in an automatic way. A `try` construct is used to prevent the `nlreg.diag` method from breaking down. Hence, the values of the diagnostics are not available where a convergence problem was encountered. A warning is issued whenever this occurs.

References

Bellio, R. (2000) *Likelihood Asymptotics: Applications in Biostatistics*. Ph.D. Thesis, Department of Statistics, University of Padova.

Brazzale, A. R. (2000) *Practical Small-Sample Parametric Inference*. Ph.D. Thesis N. 2230, Department of Mathematics, Swiss Federal Institute of Technology Lausanne.

Carroll, R. J. and Ruppert, D. (1988) *Transformation and Weighting in Regression*. London: Chapman & Hall.

Cook, R. D. and Weisberg, S. (1982) *Residuals and Influence in Regression*. New York: Chapman & Hall.

Davison, A. C. and Snell, E. J. (1991) Residuals and diagnostics. In *Statistical Theory and Modelling: In Honour of Sir David Cox* (eds. D. V. Hinkley, N. Reid, and E. J. Snell), 83–106. London: Chapman & Hall.

Davison, A. C. and Tsai, C.-L. (1992) Regression model diagnostics. *Int. Stat. Rev.*, **60**, 337–353.

See Also

nlreg.diag.plots, nlreg.object

Examples

```
library(boot)
data(calcium)
calcium.nl <- nlreg( cal ~ b0*(1-exp(-b1*time)), weights = ~ ( 1+time^g )^2,
                     data=calcium, start = c(b0 = 4, b1 = 0.1, g = 1),
                     hoa = TRUE )
##
calcium.diag <- nlreg.diag( calcium.nl )
plot( calcium.diag, which = 9 )
##
calcium.diag <- nlreg.diag( calcium.nl, hoa = FALSE, infl = FALSE)
plot(calcium.diag, which = 9)
## Not available
```

---

nlreg.object *Nonlinear Heteroscedastic Model Object*

Description

Class of objects returned when fitting a nonlinear heteroscedastic model.

Arguments

The following components must be included in a nlreg object:

coef	the MLEs of the regression coefficients, that is, of the parameters appearing in the right-hand side of the formula argument in the call that generated the nlreg object.

`varPar` the MLEs of the variance parameters appearing in the `weights` argument of the call that generated the `nlreg` object. If this argument was missing, the MLE of $\log(\sigma^2)$, the logarithm of the constant variance, is returned.

`offset` a numerical vector with a single named element indicating the parameter of interest and the value to which it was fixed while fitting the nonlinear model.

`logLik` the log likelihood from the fit.

`meanFun` the formula expression of the mean function.

`varFun` the formula expression of the variance function.

`data` a list representing a summary of the original data with the following components:

`'offset name'` the predictor variable with no duplicated value.

`repl` the number of replicates available for each value of the predictor.

`dupl` a vector of the same length than the predictor variable indicating the position of each data point in the *offset name* component.

`t1` the sum of the reponses for each design point in the *offset name* component.

`t2` the sum of the squared responses for each design point in the *offset name* component.

`fitted` the fitted values, that is, the mean function evaluated at each data point.

`weights` the variance function evaluated at each data point.

`residuals` the response/standardized residuals from the fit.

`start` the starting values used to initialize the fitting routine.

`call` an image of the call to `nlreg`, but with all the arguments explicitly named.

`ws` a list containing information that is used in subsequent calculations with the following components:

`allPar` the MLEs of all parameters.

`homVar` a logical value indicating whether the variance function is constant.

`xVar` a logical value indicating whether the variance function depends on the predictor variable.

`hoa` the value of the `hoa` argument in the call that generated the `nlreg` object.

`missingData` a logical value indicating whether the `data` argument was missing in the call that generated the `nlreg` object.

`frame` the name of the data frame if specified in the call to `nlreg`.

`iter` the number of iteration required until convergence (only for non constant variance function).

`md` a function definition that returns the first two derivatives of the mean function if `hoa = TRUE` in the function call that generated the `nlreg` object.

`vd` a function definition that returns the first two derivatives of the variance function if `hoa = TRUE` in the function call that generated the `nlreg` object.

Generation

This class of objects is returned by the `nlreg` function to represent a fitted nonlinear heteroscedastic model. Class `nlreg` inherits from class `nls`, which represents a homoscedastic nonlinear model fit.

Methods

Objects of this class have methods for the functions `print`, `summary`, `fitted` among others.

Note

The residuals, fitted values and coefficients should be extracted by the generic functions of the same name, rather than by the `$` operator.

The `data` and `ws` components are not intended to be directly accessed by users, but rather contain information invoked by functions such as `profile` and `nlreg.diag`.

See Also

`nlreg`, `nls`

---

obsInfo *Returns the Observed Information Matrix — Generic Function*

Description

Returns the observed information matrix from a fitted model object.

Usage

```
obsInfo(object, ...)
```

Arguments

`object`	any fitted model object for which the observed information matrix can be calculated.
`...`	absorbs any additional argument.

Details

This function is generic (see `methods`); method functions can be written to handle specific classes of data. Classes which already have methods for this function include: `nlreg`.

Value

the observed information matrix for a fitted regression model.

See Also

`obsInfo.nlreg`, `nlreg.object`, `expInfo`

Examples

```
data(metsulfuron)
metsulfuron.nl <-
    nlreg( log(area) ~ log( b1+(b2-b1) / (1+(dose/b4)^b3) ),
           weights = ~ ( 1+dose^exp(g) )^2, data = metsulfuron,
           start = c(b1 = 138, b2 = 2470, b3 = 2, b4 = 0.07, g = log(0.3)),
           hoa = TRUE)
obsInfo( metsulfuron.nl )
```

`obsInfo.nlreg` *Observed Information Matrix for 'nlreg' Objects*

Description

Returns the observed information matrix from a fitted `nlreg` model.

Usage

```
## S3 method for class 'nlreg'
obsInfo(object, par, mu, v, m1 = NULL, m2 = NULL, v1 = NULL,
        v2 = NULL, ...)
```

Arguments

`object`	a fitted `nlreg` object such as returned by a call to `nlreg`.
`par`	a vector of parameter values where each element is named after the parameter it represents. If missing, the values in the `ws$allPar` component of `object` are used.
`mu`	numerical vector containing the mean function evaluated at each data point. If missing, the fitted values saved in `object` are used.
`v`	numerical vector containing the variance function evaluated at each data point. If missing, the values of the `weights` component of `object` are used.
`m1`	a matrix whose rows represent the gradients of the mean function evaluated at each data point. If `NULL`, the `gradient` attribute of the object returned by a call to `Dmean` is used.
`m2`	a three-way array whose rows represent the Hessian of the mean function evaluated at each data point. If `NULL`, the `hessian` attribute of the object returned by a call to `Dmean` is used.
`v1`	a matrix whose rows represent the gradient of the variance function evaluated at each data point. If `NULL`, the `gradient` attribute of the object returned by a call to `Dvar` is used.
`v2`	a three-way array whose rows represent the Hessian of the variance function evaluated at each data point. If `NULL`, the `hessian` attribute of the object returned by a call to `Dvar` is used.
`...`	absorbs any additional argument.

Details

This function is a method for the generic function obsInfo for objects inheriting from class nlreg.

Value

the observed information matrix of the fitted nonlinear model passed through the object argument.

Note

This function is mostly intended for internal use. It is called by functions such as summary.nlreg and profile.nlreg. To extract the observed information matrix from a fitted nlreg object, the generic method obsInfo should be used.

See Also

obsInfo, nlreg.object, expInfo

---

param *Extract All Parameters from a Model — Generic Function*

Description

This function extracts all parameters (regression coefficients, variance parameters etc.) from a fitted model.

Usage

```
param(object, ...)
```

Arguments

object	any fitted model object from which parameters may be extracted.
...	additional arguments like digits to control how many digits should be printed.

Details

This function is generic (see methods); method functions can be written to handle specific classes of data. Classes which already have methods for this function include: nlreg.

Value

all parameters (regression coefficients, variance parameters etc.) of a fitted model.

See Also

param.nlreg, methods

Examples

```
data(metsulfuron)
metsulfuron.nl <-
    nlreg( log(area) ~ log( b1+(b2-b1) / (1+(dose/b4)^b3) ),
           weights = ~( 1+dose^exp(g) )^2, data = metsulfuron,
           start = c(b1 = 138, b2 = 2470, b3 = 2, b4 = 0.07, g = log(0.3)),
           hoa = TRUE )
param( metsulfuron.nl )
##          b1           b2           b3           b4            g         logs
## 139.0395322 2471.5097481    1.7091297    0.0772535   -1.2582597   -3.8198406
```

`plot.nlreg.contours` *Use plot() on a 'nlreg.contours' object*

Description

This is a method for the function `plot` for objects inheriting from class `nlreg.contours`.

Usage

```
## S3 method for class 'nlreg.contours'
plot(x, alpha = c(0.1, 0.05), drawlabels = FALSE, lwd1 = 1, lwd2 = 1,
     lty1 = "solid", lty2 = "solid", cl1 = "blue", cl2 = "red",
     col = "black", pch1 = 1, pch2 = 16, cex = 0.5, ...)
```

Arguments

`x`	a `nlreg.contours` object, that is, the result of a call to `contour.all.nlreg.profiles`.
`alpha`	a numerical vector defining the levels of the contours; the default is `c(0.1, 0.05)`, that is, $1-\alpha = 0.9$ and $1-\alpha = 0.95$.
`drawlabels`	logical value. Contours are labelled if `TRUE`.
`lwd1, lwd2`	the line widths used to compare different curves in the same plot; default is `lwd2 = 2` for higher order solutions and `lwd1 = 1` for first order solutions.
`lty1, lty2`	line types used to compare different curves in the same plot; default is `"solid"` for all statistics.
`cl1, cl2, col`	colors used to compare different curves in the same plot; default is `cl2 = "red"` for higher order solutions, and `cl1 = "blue"` for the remaining first order statistics. The default color of the plot is `col = "black"`.
`pch1, pch2`	character types used to compare different values in the same plot; default is `pch2 = 16` for higher order solutions, and `pch1 = 1` for the remaining first order statistics.
`cex`	the character expansions relative to the standard size of the device to be used for printing text. The default is `cex = 0.5`.
`...`	additional graphics parameters.

Value

No value is returned.

Side Effects

A plot is produced on the current graphics device.

See Also

nlreg.contours.object, contour.all.nlreg.profiles

Examples

```
## Not run:
data(metsulfuron)
metsulfuron.nl <-
    nlreg( formula = log(area) ~ log( b1+(b2-b1) / (1+(dose/b4)^b3) ),
           weights = ~ ( 1+dose^exp(g) )^2, data = metsulfuron,
           start = c(b1 = 138, b2 = 2470, b3 = 2, b4 = 0.07, g = log(0.3)),
           hoa = TRUE )
##
metsulfuron.prof <- profile( metsulfuron.nl, trace = TRUE )
metsulfuron.cont <- contour( metsulfuron.prof, ret = TRUE, plotit = FALSE )
par( mai = rep(0.2, 4) )
plot( metsulfuron.cont )
## End(Not run)
```

---

plot.nlreg.diag *Diagnostic Plots for Nonlinear Heteroscedastic Models*

Description

The nlreg.diag.plots routine generates diagnostic plots for a nonlinear heteroscedastic model using different types of residuals, influence measures and leverages. This is equivalent to using the plot.nlreg.diag method for function plot for objects inheriting from class nlreg.diag.

Usage

```
nlreg.diag.plots(fitted, which = "all", subset = NULL, iden = FALSE,
                 labels = NULL, hoa = TRUE, infl = TRUE,
                 trace = FALSE, ret = FALSE, ...)
## S3 method for class 'nlreg.diag'
plot(x, which = "all", subset = NULL, iden = FALSE, labels = NULL,
     ...)
```

Arguments

fitted	either a nlreg object, that is, the result of a call to nlreg, or a nlreg.diag object obtained from a call to nlreg.diag.
x	a nlreg.diag object obtained from a call to nlreg.diag.
which	which plot to draw. Admissible values are 2 to 9 which correspond to the choices below. The default is "all", which pops up a menu that lists all available plots.
subset	the subset of the data used in the original nlreg fit. Must be the same than the subset option used in the call to nlreg that generated the nlreg object for which the diagnostic plots are to be drawn. Needed only if the subset option is used in the call to nlreg.
iden	logical argument. If TRUE, the user will be prompted after the plots are drawn. A positive integer will select a plot and invoke identify() on that plot. After exiting identify, the user is again prompted, this loop continuing until the user responds to the prompt with 0. If iden is FALSE (default) the user cannot interact with the plots.
labels	a vector of labels for use with identify if iden is TRUE. If it is not supplied, the labels are derived from the nlreg.diag object argument fitted or x.
hoa	logical value indicating whether higher order asymptotics should be used for calculating the regression diagnostics. Needed only if fitted is a nlreg object. Default is TRUE.
infl	logical value indicating whether influence measures should be calculated on the basis of a leave-one-out analysis. Needed only if fitted is a nlreg object. Default is TRUE.
trace	logical value. If TRUE details of the iterations are printed. Needed only if fitted is a nlreg object. Default is FALSE.
ret	logical argument indicating whether the nlreg.diag object should be returned; the default is FALSE.
...	additional graphics parameters.

Details

The diagnostics required for the plots are calculated by nlreg.diag, either by passing a nlreg.diag object or by applying nlreg.diag internally to the nlreg object specified through fitted. These are then used to produce the plots on the current graphics device. A menu lists all possible choices. They may be one or all of the following.

```
 Make a plot selection (or 0 to exit)

1:plot: Summary
2:plot: Studentized residuals against fitted values
3:plot: r* residuals against fitted values
4:plot: Normal QQ-plot of studentized residuals
5:plot: Normal QQ-plot of r* residuals
6:plot: Cook statistic against h/(1-h)
7:plot: Global influence against h/(1-h)
```

```
8:plot: Cook statistic against observation number
9:plot: Influence measures against observation number

Selection:
```

In the normal scores plots, the dotted line represents the expected line if the residuals are normally distributed, that is, it is the line with intercept 0 and slope 1.

In general, when plotting Cook's distance or the global influence measure against the standardized leverages, there will be two dotted lines on the plot. The horizontal line is at $8/(n-2p)$, where n is the number of observations and p is the number of regression coefficients estimated. Points above this line may be points with high influence on the model. The vertical line is at $2p/(n-2p)$ and points to the right of this line have high leverage compared to the variance of the raw residual at that point. If all points are below the horizontal line or to the left of the vertical line then the line is not shown.

Use of `iden = TRUE` is encouraged for proper exploration of these plots as a guide to how well the model fits the data and whether certain observations have an unduly large effect on parameter estimates.

Value

If `ret = TRUE`, the `nlreg.diag` object is returned. Otherwise, there is no returned value.

Side Effects

The current device is cleared. If `iden = TRUE`, interactive identification of points is enabled. All screens are closed, but not cleared, on termination of the function.

Acknowledgments

This function is based on A. J. Canty's function `glm.diag.plots` contained in library `boot`.

Note

Choices `3` and `5` are not available if `hoa = FALSE` in the call to `nlreg.diag` that generated the `x` argument. Choices `7` and `9` are not available if `infl = FALSE` in the same call. Plot number `9` is furthermore not available if the variance function is constant.

References

Brazzale, A. R. (2000) *Practical Small-Sample Parametric Inference*. Ph.D. Thesis N. 2230, Department of Mathematics, Swiss Federal Institute of Technology Lausanne. Section 6.3.1 and Appendix A.2.2.

Davison, A. C. and Snell, E. J. (1991) Residuals and diagnostics. In *Statistical Theory and Modelling: In Honour of Sir David Cox* (eds. D. V. Hinkley, N. Reid, and E. J. Snell), 83–106. London: Chapman & Hall.

Davison, A. C. and Tsai, C.-L. (1992) Regression model diagnostics. *Int. Stat. Rev.*, **60**, 337–353.

See Also

nlreg.diag, nlreg.object, identify

Examples

```
library(boot)
data(calcium)
calcium.nl <- nlreg( cal ~ b0*(1-exp(-b1*time)), weights = ~ ( 1+time^g )^2,
                     start = c(b0 = 4, b1 = 0.1, g = 1), data = calcium,
                     hoa = TRUE )
##
calcium.diag <- nlreg.diag( calcium.nl, trace = TRUE )
##
## menu-driven
## Not run:
plot( calcium.diag )
##
##  Make a plot selection (or 0 to exit)
##
## 1:plot: Summary
## 2:plot: Studentized residuals against fitted values
## 3:plot: r* residuals against fitted values
## 4:plot: Normal QQ-plot of studentized residuals
## 5:plot: Normal QQ-plot of r* residuals
## 6:plot: Cook statistic against h/(1-h)
## 7:plot: Global influence against h/(1-h)
## 8:plot: Cook statistic against observation number
## 9:plot: Influence measures against observation number
##
## Selection:
## End(Not run)
##
## plot 5: Normal QQ-plot of r* residuals
plot( calcium.diag, which = 5, las = 1 )
##
nlreg.diag.plots( calcium.nl, which = 5, las = 1 )
```

---

plot.nlreg.profiles *Use plot() on a 'profile.nlreg' and 'all.profiles.nlreg' object*

Description

These are methods for the function plot for objects inheriting from class "profile.nlreg" or "all.profiles.nlreg" .

Usage

```
## S3 method for class 'nlreg.profile'
plot(x, alpha = 0.05, add.leg = FALSE, stats = c("sk", "fr"),
```

```
     cex = 0.7, cex.lab = 1, cex.axis = 1, cex.main = 1, lwd1 = 1,
     lwd2 = 2, lty1 = "solid", lty2 = "solid", cl1 = "blue",
     cl2 = "red", col = "black", ylim = c(-3,3), ...)
## S3 method for class 'all.nlreg.profiles'
plot(x, nframe, alpha = 0.05, stats = c("sk", "fr"), cex = 0.7,
     cex.lab = 1, cex.axis = 1, cex.main = 1, lwd1 = 1, lwd2 = 2,
     lty1 = "solid", lty2 = "solid", cl1 = "blue", cl2 = "red",
     col = "black", ylim = c(-3,3), ...)
```

Arguments

x	an object of class `profile.nlreg` or `all.profiles.nlreg` such as generated by a call to `profile.nlreg`.
nframe	the number of frames into which to split the graphics device; only if `x` is an `all.profiles.nlreg` object.
alpha	numeric vector with the levels used to read off confidence intervals; the default is 5% which corresponds to a confidence level of $1 - \alpha = 0.95$.
stats	character value indicating which higher order statistics to plot. Admissible values are `"sk"` for *Skovgaard's (1996)* proposal and `"fr"` for *Fraser, Reid and Wu's (1999)* approach. The default is `"sk"`.
add.leg	logical value indicating whether a legend should be added to the plot; only if `x` is a `profile.nlreg` object. The default is `FALSE`.
cex, cex.lab, cex.axis, cex.main	the character expansions relative to the standard size of the device to be used for printing text, labels, axes and main title. See `par` for details.
lwd1, lwd2	the line widths used to compare different curves in the same plot; default is `lwd2 = 2` for higher order solutions and `lwd1 = 1` for first order solutions.
lty1, lty2	line types used to compare different curves in the same plot; default is `"solid"` for all statistics.
cl1, cl2, col	colors used to compare different curves in the same plot; default is `cl2 = "red"` for higher order solutions, and `cl1 = "blue"` for the remaining first order statistics. The default color of the plot is `col = "black"`.
ylim	a numerical vector with two elements defining the y-axis range; only if `x` is a `profile.nlreg` object.
...	additional graphics parameters.

Details

The function defaults to:

```
plot.nlreg.profile(x = stop("nothing to plot"), alpha = 0.05, add.leg = FALSE,
                   stats = c("sk", "fr"), cex = 0.7, cex.lab = 1, cex.axis = 1,
                cex.main = 1, lwd1 = 1, lwd2 = 2, lty1 = "solid", lty2 = "solid",
                 cl1 = "blue", cl2 = "red", col = "black", ylim = c(-3,3), \dots)
```

```
plot.all.nlreg.profiles(x = stop("nothing to plot"), nframe, alpha = 0.05,
                        stats = c("sk", "fr"), cex = 0.7, cex.lab = 1, cex.axis = 1,
                    cex.main = 1, lwd1 = 1, lwd2 = 2, lty1 = "solid", lty2 = "solid",
                    cl1 = "blue", cl2 = "red", col = "black", ylim = c(-3,3), \dots)
```

Value

No value is returned.

Side Effects

A plot is produced on the current graphics device.

References

Fraser, D.A.S., Reid, N. and Wu, J. (1999). A simple general formula for tail probabilities for frequentist and Bayesian inference. *Biometrika*, **86**, 249–264.

Skovgaard, I. (1996) An explicit large-deviation approximation to one-parameter tests. *Bernoulli*, **2**, 145–165.

See Also

profile.nlreg, nlreg.profile.objects, plot

Examples

```
## Not run:
data(metsulfuron)
metsulfuron.nl <-
    nlreg( formula = log(area) ~ log( b1+(b2-b1) / (1+(dose/b4)^b3) ),
           weights = ~ ( 1+dose^exp(g) )^2, data = metsulfuron,
           start = c(b1 = 138, b2 = 2470, b3 = 2, b4 = 0.07, g = log(0.3)),
           hoa = TRUE)
##
metsulfuron.prof <- profile( metsulfuron.nl, offset = g, trace = TRUE )
plot( metsulfuron.prof, lwd2 = 2 )
##
metsulfuron.prof <- profile( metsulfuron.nl, trace = TRUE )
plot( metsulfuron.prof, lwd2 = 2, nframe = c(2,3) )
## End(Not run)
```

---

profile.nlreg *Profile Method for 'nlreg' Objects*

Description

Returns a list of elements for profiling a nonlinear heteroscedastic model.

Usage

```
## S3 method for class 'nlreg'
profile(fitted, offset = "all", hoa = TRUE, precision = 6,
        signif = 30, n = 50, omit = 0.5, trace = FALSE, md, vd,
        all = FALSE, ...)
```

Arguments

fitted	a fitted nlreg object such as returned by a call to nlreg.
offset	a single named element representing the parameter of interest (a regression coefficient or a variance parameter), or "all" if all parameters are to be profiled, provided that the model formula contains more than one regression coefficient. The constant term $\log(\sigma^2)$ which is included by default in the variance function is referred to by logs (see the weights argument in nlreg). The default is "all".
hoa	logical value indicating whether higher order statistics should be included; the default is TRUE.
precision	numerical value defining the maximum range of values, given by MLE $\pm$ precision$*$S.E., that are profiled. The default is 6.
signif	the maximum number of output points that are calculated exactly; the default is 30.
n	the approximate number of output points produced by the spline interpolation; the default is 50.
omit	numerical value defining the range of values, given by MLE $\pm$ omit$*$S.E, which is omitted in the spline interpolation of the higher order statistics. The purpose is to avoid numerical instabilities around the maximum likelihood estimate.
trace	if TRUE, details of the iterations are printed.
md	a function definition that returns the first two derivatives of the mean function; used by allProfiles.nlreg.
vd	a function definition that returns the first two derivatives of the variance function; used by allProfiles.nlreg.
all	logical switch used by allProfiles.nlreg.
...	absorbs any additional argument.

Details

The function profile.nlreg calculates all elements necessary for profiling a scalar parameter of interest or all model parameters. The model formula must contain more than one regression coefficient.

A classical profile plot (*Bates and Watts, 1988, Section 6.1.2*) is a plot of the likelihood root statistic and of the Wald statistic against a range of values for the interest parameter. It provides a means to assess the accuracy of the normal approximation to the distribution of both statistics: the closer the corresponding curves are, the better the approximations. Confidence intervals can easily be read off for any desired level: the confidence bounds identify with the values on the x-axis at which the curves intersect the horizontal lines representing the standard normal quantiles of the desired level.

Profiling is performed by updating a fitted nonlinear heteroscedastic model. All statistics are calculated exactly for at maximum `signif` equally spaced points distributed around the MLE. To save execution time, the iterations start with a value close to the MLE and proceed in the two directions MLE $\pm$ δ, until the absolute value of all statistics exceeds the threshold 2.4. The step size δ is defined by the `signif` argument. A spline interpolation is used to extend them over the whole interval of interest. A range of values, defined by the `omit` argument is omitted to avoid numerical instabilities around the MLE. All results are stored in an object of class `nlreg.profile` or `all.nlreg.profiles` depending on the value assumed by the `offset` argument. The `summary` and `plot` method functions must be used to examine the output or represent it graphically. No `print` method is available.

If `hoa = TRUE`, `profile.nlreg` produces an enhanced version of the classical profile plots by including the third order modified likelihood root statistic r^*. More precisely, it implements two approximations to *Barndorff-Nielsens's (1991)* original formulation where the sample space derivatives are replaced by respectively the approximations proposed in *Skovgaard (1996)* and *Fraser, Reid and Wu (1999)*. The idea is to provide insight into the behaviour of first order methods, such as detecting possible bias of the estimates or the influence of the model curvature.

The theory and statistics used are summarized in *Brazzale (2000, Chapters 2 and 3)*. More details of the implementation are given in *Brazzale (1999; 2000, Section 6.3.2)*.

Value

a list of elements of class `nlreg.profile` or, if `offset = "all"`, of class `all.nlreg.profiles` for profiling a nonlinear heteroscedastic model. The `nlreg.profile` class considers a scalar parameter of interest, while the `all.nlreg.profiles` class ontains the profiles of all parameters – regression coefficients and variance parameters.

Side Effects

If `trace = TRUE`, the parameter which is currently profiled and the corresponding value are printed.

Note

`profile.nlreg` is a method for the generic function `profile` for class `nlreg`. It can be invoked by calling `profile` for an object of the appropriate class, or directly by calling `profile.nlreg`.

To obtain the profiles of the different statistics considered, the model is refitted several times while keeping the value of the parameter of interest fixed. Although rarely, convergence problems may occur as the starting values are chosen in an automatic way. A `try` construct is used to prevent the `profile.nlreg` method from breaking down. Hence, the values of the statistics are not available where a convergence problem was encountered. A warning is issued whenever this occurs.

References

Barndorff-Nielsen, O. E. (1991) Modified signed log likelihood ratio. *Biometrika*, **78**, 557–564.

Bates, D. M. and Watts, D. G. (1988) *Nonlinear Regression Analysis and Its Applications*. New York: Wiley.

Brazzale, A. R. (2000) *Practical Small-Sample Parametric Inference*. Ph.D. Thesis N. 2230, Department of Mathematics, Swiss Federal Institute of Technology Lausanne.

Fraser, D.A.S., Reid, N. and Wu, J. (1999). A simple general formula for tail probabilities for frequentist and Bayesian inference. *Biometrika*, **86**, 249–264.

Skovgaard, I. (1996) An explicit large-deviation approximation to one-parameter tests. *Bernoulli*, **2**, 145–165.

See Also

nlreg.profile.object, all.nlreg.profiles.object, profile

Examples

```
## Not run:
data(metsulfuron)
metsulfuron.nl <-
    nlreg( formula = log(area) ~ log( b1+(b2-b1) / (1+(dose/b4)^b3) ),
           weights = ~ ( 1+dose^exp(g) )^2, data = metsulfuron,
           start = c(b1 = 138, b2 = 2470, b3 = 2, b4 = 0.07, g = log(0.3)),
           hoa = TRUE )
##
metsulfuron.prof <- profile( metsulfuron.nl, offset = g, trace = TRUE )
plot( metsulfuron.prof, lwd2 = 2 )
#
metsulfuron.prof <- profile( metsulfuron.nl, trace = TRUE )
plot( metsulfuron.prof, nframe = c(2,3), lwd2 = 2 )
## End(Not run)
```

---

ria *Radioimmunoassay Data*

Description

The ria data frame has 16 rows and 2 columns.

Run of a radioimmunoassay (RIA) to estimate the concentrations of a drug in samples of porcine serum. The experiment consists of 16 observations made at 8 different drug levels with two replications at each level.

Usage

```
data(ria)
```

Format

This data frame contains the following columns:

conc the drug concentration (ng/ml);

count the observed percentage of radioactive gamma counts.

Source

The data were obtained from

Belanger, B. A., Davidian, M. and Giltinan, D. M. (1996) The effect of variance function estimation on nonlinear calibration inference in immunoassay data. *Biometrics*, **52**, 158–175. Table 1, first two columns.

References

Brazzale, A. R. (2000) *Practical Small-Sample Parametric Inference*. Ph.D. Thesis N. 2230, Department of Mathematics, Swiss Federal Institute of Technology Lausanne. Section 5.3, Example 6.

Examples

```
data(ria)
attach(ria)
plot(conc, count, xlab="drug concentration (ng/ml)", ylab="gamma counts (%)")
detach()
```

---

summary.all.nlreg.profiles

Summary Method for Objects of Class 'all.nlreg.profiles'

Description

Returns a summary list for objects of class `all.nlreg.profiles`.

Usage

```
## S3 method for class 'all.nlreg.profiles'
summary(object, alpha = 0.05, twoside = TRUE, digits = NULL, ...)
```

Arguments

`object`	an `all.nlreg.profiles` object, that is, the result from a call to `profile.nlreg` with `offset = "all"`.
`alpha`	a vector of levels for confidence intervals; the default is 95%, that is, $1 - \alpha = 0.95$.
`twoside`	a logical value. If `TRUE`, two-sided confidence intervals are returned. The default is `TRUE`.
`digits`	the number of significant digits to be printed.
`...`	absorbs any additional argument.

Details

This function is a method for the generic function summary for objects of class all.nlreg.profiles. It can be invoked by calling summary or directly summary.all.nlreg.profiles for an object of the appropriate class.

Value

A list is returned where the first components are named after the parameters of the nonlinear model that was profiled. Each component represents a matrix with $k\dim(\texttt{alpha})$ rows and 2 columns, where k equals 2 or 4 depending on whether hoa = TRUE in the call that generated the nlreg.profile object. This matrix contains the upper and lower confidence bounds for the test statistics considered and for the confidence levels defined through alpha. The remaining components are the following:

mle	a $2 \times d$ matrix containing the MLEs and S.E.s of the d parameters.
offset	a vector of character strings returning the parameter names.
twoside	a logical value indicating whether two-sided or one-sided confidence intervals were calculated.
hoa	a logical value indicating whether higher order solutions were calculated.
digits	the number of significant digits to be printed.
call	an image of the call that produced the object, but with all arguments named.

See Also

nlreg.profile.objects, profile.nlreg, summary

Examples

```
## Not run:
data(metsulfuron)
metsulfuron.nl <-
    nlreg( formula = log(area) ~ log( b1+(b2-b1) / (1+(dose/b4)^b3) ),
           weights = ~ ( 1+dose^exp(g) )^2, data = metsulfuron,
           start = c(b1 = 138, b2 = 2470, b3 = 2, b4 = 0.07, g = log(0.3)),
           hoa = TRUE )
##
metsulfuron.prof <- profile( metsulfuron.nl, trace = TRUE )
summary( metsulfuron.prof, alpha = c(0.1, 0.05) )
## End(Not run)
```

---

summary.mpl *Summary Method for 'mpl' Objects*

Description

Returns a summary list for objects of class mpl.

Usage

```
## S3 method for class 'mpl'
summary(object, correlation = FALSE, digits = NULL, ...)
```

Arguments

object	a fitted mpl object, that is, the result of a call to mpl.nlreg.
correlation	logical argument. If TRUE, the (asymptotic) correlation matrix for the parameter estimates is computed; default is FALSE.
digits	the number of significant digits to be printed. Defaults to NULL.
...	absorbs any additional argument.

Details

This function is a method for the generic function summary for class mpl. It can be invoked by calling summary for an object of the appropriate class, or directly by calling summary.mpl regardless of the class of the object.

Value

A list is returned with the following components:

varPar	the maximum adjusted profile likelihood estimates of the variance parameters.
coefficients	the constrained MLEs of the regression coefficients given the maximum adjusted profile likelihood estimates of the variance parameters.
offset	the values passed through the offset argument in the call to mpl.nlreg that generated the mpl object and to which the variance parameters were fixed.
varParMLE	the MLEs of the variance parameters.
coefMLE	the MLEs of the regression coefficients.
varParCov	the (asymptotic) covariance matrix of the variance parameters, that is, the corresponding block in the inverse of the observed information matrix.
coefCov	the (asymptotic) covariance matrix of the regression coefficients, that is, the corresponding block in the inverse of the observed information matrix.
lmp	the adjusted profile log likelihood from the fit.
lp	the profile log likelihood from the fit.
stats	the indicator of which higher order solution was used.
formula	the model formula.
meanFun	the formula expression of the mean function.
varFun	the formula expression of the variance function.
data	a list representing a summary of the original data with the following components.
	'offset name' the predictor variable with no duplicated value.
	repl the number of replicates available for each value of the predictor.

dupl a vector of the same length than the predictor variable indicating the position of each data point in the *offset name* component.

t1 the sum of the reponses for each design point in the *offset name* component.

t2 the sum of the squared responses for each design point in the *offset name* component.

nobs the number of observations.

iter the number of interations needed for convergence; only if offset was not NULL in the call to mpl.nlreg which generated object.

call an image of the call to mpl.nlreg, but with all the arguments explicitly named.

ws the workspace component of the original nlreg object plus the following components:

corr a logical value indicating whether the correlation matrix should be printed.

digits the number of significant digits to be printed.

See Also

mpl.object, nlreg.object, summary

Examples

```
data(metsulfuron)
metsulfuron.nl <-
    nlreg( formula = log(area) ~ log( b1+(b2-b1) / (1+(dose/b4)^b3) ),
           weights = ~ ( 1+dose^exp(g) )^2, data = metsulfuron, hoa = TRUE,
           start = c(b1 = 138, b2 = 2470, b3 = 2, b4 = 0.07, g = log(0.3)) )
##
metsulfuron.mpl <- mpl( metsulfuron.nl, trace = TRUE )
summary( metsulfuron.mpl, corr = FALSE )
```

---

summary.nlreg *Summary Method for Nonlinear Heteroscedastic Models*

Description

Returns a summary list for a fitted nonlinear heteroscedastic model.

Usage

```
## S3 method for class 'nlreg'
summary(object, observed = TRUE, correlation = FALSE,
        digits = NULL, ...)
```

Arguments

object	a fitted nlreg object. This is assumed to be the result of some fit that produces an object inheriting from the class nlreg, in the sense that the components returned by the nlreg function will be available.
observed	logical argument. If TRUE, the observed information is used to calculate the covariance matrix, the expected information otherwise. The default is TRUE.
correlation	logical argument. If TRUE, the correlation matrix for the parameter estimates is computed; default is TRUE.
digits	the number of significant digits to be printed.
...	absorbs any additional argument.

Details

This function is a method for the generic function summary for class nlreg. It can be invoked by calling summary for an object of the appropriate class, or directly by calling summary.nlreg regardless of the class of the object.

Value

A list is returned with the following components:

coefficients	a matrix with four columns, containing the MLEs of the regression coefficients, their standard errors, the z-value (or Wald statistic) and the associated p-value based on the standard normal approximation to the distribution of the z statistic.
varPar	a matrix with two columns, containing the MLEs of the variance parameters and their standard errors.
offset	a numerical vector with a single named element indicating the parameter of interest and the value to which it was fixed while fitting the nonlinear model.
residuals	the response residuals from the fit.
covariance	the (asymptotic) covariance matrix for the parameter estimates.
correlation	the (asymptotic) correlation matrix for the parameter estimates.
logLik	the log likelihood from the fit.
call	an image of the call that produced the nlreg object, but with the arguments all named.
digits	then number of significant digits to be printed.
ws	the ws component of the nlreg object.

See Also

nlreg.object, summary

Examples

```
data(metsulfuron)
metsulfuron.nl <-
    nlreg( formula = log(area) ~ log( b1+(b2-b1) / (1+(dose/b4)^b3) ),
           weights = ~ ( 1+dose^exp(g) )^2, data = metsulfuron,
           start = c(b1 = 138, b2 = 2470, b3 = 2, b4 = 0.07, g = log(0.3)),
           hoa = TRUE )
##
summary( metsulfuron.nl, digits = 3 )
##
print( summary( metsulfuron.nl )$cov, digits = 3 )
print( summary( metsulfuron.nl, observed = FALSE )$cov, digits = 3 )
```

---

summary.nlreg.profile *Summary Method for Objects of Class 'nlreg.profile'*

Description

Returns a summary list for objects of class `nlreg.profile`.

Usage

```
## S3 method for class 'nlreg.profile'
summary(object, alpha = 0.05, twoside = TRUE, digits = NULL, ...)
```

Arguments

`object`	a `nlreg.profile` object, that is, the result of a call to `profile.nlreg`.
`alpha`	a vector of levels for confidence intervals; the default is $1 - \alpha = 0.95$.
`twoside`	a logical value. If `TRUE`, two-sided confidence intervals are returned. The default is `TRUE`.
`digits`	the number of significant digits to be printed.
`...`	absorbs any additional argument.

Details

This function is a method for the generic function `summary` for objects of class `nlreg.profile`. It can be invoked by calling `summary` or directly `summary.nlreg.profile` for an object of the appropriate class.

Value

A list is returned with the following components:

`CI`	a matrix with $k \dim($`alpha`$)$ rows and 2 columns, where k equals 2 or 4 depending on whether `hoa = TRUE` in the call that generated `object`. This matrix contains the upper and lower confidence bounds for the considered test statistics and for the confidence levels specified through `alpha`.

`inf.sk`, `np.sk`, `inf.fr`, `np.fr`
: the information and nuisance parameters aspects, that is, the two terms into which the higher order adjustment leading to the r^* statistic can be decomposed. The two versions refer to respectively *Skovgaard's (1996)* proposal and *Fraser, Reid and Wu's (1999)* solution. Only if `hoa = TRUE` in the function call that generated the `nlreg.profile` object argument `object`.

`mle`
: a numerical vector giving the MLE of the parameter of interest and its standard error.

`offset`
: character string giving the name of the interest parameter.

`twoside`
: a logical value indicating whether two-sided or one-sided confidence intervals were calculated.

`points`
: the number of output points at which the considered statistics were calculated exactly.

`n`
: the approximate number of points used in the spline interpolation of the considered statistics.

`hoa`
: a logical value indicating whether higher order solutions were calculated.

`digits`
: the number of significant digits to be printed.

`call`
: an image of the call that produced the object, but with all arguments named.

`...`
: absorbs additional arguments.

References

Fraser, D.A.S., Reid, N. and Wu, J. (1999). A simple general formula for tail probabilities for frequentist and Bayesian inference. *Biometrika*, **86**, 249–264.

Skovgaard, I. (1996) An explicit large-deviation approximation to one-parameter tests. *Bernoulli*, **2**, 145–165.

See Also

`nlreg.profile.object`, `profile.nlreg`, `summary`

Examples

```
data(metsulfuron)
metsulfuron.nl <-
    nlreg( formula = log(area) ~ log( b1+(b2-b1) / (1+(dose/b4)^b3) ),
           weights = ~ ( 1+dose^exp(g) )^2, data = metsulfuron,
           start = c(b1 = 138, b2 = 2470, b3 = 2, b4 = 0.07, g = log(0.3)),
           hoa = TRUE )
##
metsulfuron.prof <- profile( metsulfuron.nl, offset = g, trace = TRUE )
summary( metsulfuron.prof, alpha = c(0.9, 0.95) )
```

var2cor *Convert Covariance Matrix to Correlation Matrix — Generic Function*

Description

This function converts the covariance matrix from a fitted model into the correlation matrix.

Usage

```
var2cor(object, ...)
```

Arguments

object	any fitted model object from which a covariance matrix may be extracted.
...	absorbs any additional argument.

Details

This function is generic (see methods); method functions can be written to handle specific classes of data. Classes which already have methods for this function include: nlreg, summary.nlreg, mpl and summary.mpl.

Value

the correlation matrix of the estimates from a fitted model.

See Also

var2cor.nlreg, var2cor.mpl, methods

Examples

```
data(metsulfuron)
metsulfuron.nl <-
    nlreg( log(area) ~ log( b1+(b2-b1) / (1+(dose/b4)^b3) ),
           weights = ~( 1+dose^exp(g) )^2, data = metsulfuron,
           start = c(b1 = 138, b2 = 2470, b3 = 2, b4 = 0.07, g = log(0.3)),
           hoa = TRUE )
var2cor( metsulfuron.nl )
##
metsulfuron.sum <- summary( metsulfuron.nl, corr = FALSE )
var2cor( metsulfuron.sum )
```

Index