# Package 'hoa'

August 12, 2015

**Version** 2.1.4

**Date** 2015-08-11

**Title** Higher Order Likelihood Inference

**Author** R port by Alessandra R. Brazzale <alessandra.brazzale@unipd.it>, following earlier work by Douglas Bates. The function tem is based on work by Anthony Davison <Anthony.Davison@epfl.ch>

**Maintainer** Alex-Antoine Fortin <alex@fortin.bio>

**Depends** R (>= 3.0.0)

**Imports** graphics, statmod, stats, survival, utils

**Suggests** boot, csampling

**Description** Performs likelihood-based inference for a wide range of regression models. Provides higher-order approximations for inference based on extensions of saddlepoint type arguments as discussed in the book Applied Asymptotics: Case Studies in Small-Sample Statistics by Brazzale, Davison, and Reid (2007).

**License** GPL (>= 2) | file LICENCE

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

**LazyLoad** yes

**LazyData** yes

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2015-08-12 00:35:40

## R topics documented:


aids . . . . . 3
airway . . . . . 4
babies . . . . . 5
C1 . . . . . 6
chlorsulfuron . . . . . 7
cond . . . . . 8

cond.glm . . . 9
cond.object . . . 13
cond.rsm . . . 14
contour.all.nlreg.profiles . . . 17
daphnia . . . 20
darwin . . . 21
Dmean . . . 22
dormicum . . . 24
Dvar . . . 25
expInfo . . . 27
expInfo.nlreg . . . 28
family.cond . . . 29
family.rsm . . . 30
family.rsm.object . . . 31
family.summary.cond . . . 32
fraudulent . . . 32
fungal . . . 33
helicopter . . . 34
hoa . . . 35
houses . . . 39
Huber . . . 40
logLik.nlreg . . . 41
logLik.rsm . . . 42
marg.object . . . 43
metsulfuron . . . 45
mpl . . . 46
mpl.nlreg . . . 47
mpl.object . . . 49
nlreg . . . 51
nlreg.diag . . . 54
nlreg.object . . . 57
nuclear . . . 59
obsInfo . . . 60
obsInfo.nlreg . . . 61
param . . . 62
plot.cond . . . 63
plot.fr . . . 65
plot.marg . . . 67
plot.nlreg.contours . . . 69
plot.nlreg.diag . . . 70
plot.nlreg.profiles . . . 74
print.summary.cond . . . 76
print.summary.marg . . . 77
profile.nlreg . . . 78
rabbits . . . 81
residuals.rsm . . . 82
ria . . . 83
rsm . . . 84

rsm.diag . . . 88
rsm.diag.plots . . . 90
rsm.families . . . 92
rsm.fit . . . 93
rsm.null . . . 95
rsm.object . . . 96
rsm.surv . . . 98
summary.all.nlreg.profiles . . . 99
summary.cond . . . 100
summary.fr . . . 102
summary.marg . . . 103
summary.mpl . . . 105
summary.nlreg . . . 107
summary.nlreg.profile . . . 108
summary.rsm . . . 110
tem . . . 112
update.rsm . . . 114
urine . . . 115
var2cor . . . 116
vcov.rsm . . . 117
venice . . . 118

**Index** **120**

---

`aids` *AIDS Symptoms and AZT Use Data*

---

## Description

The `aids` data frame has 4 rows and 4 columns.

On February 15, 1991, the *New York Times* published the results of a study on the presence of AIDS symptoms and AZT use. The data were cross-classified according to the race of the patients.

## Usage

```
data(aids)
```

## Format

This data frame contains the following columns:

`yes` the number of patients with AIDS symptoms;

`no` the number of patients without AIDS symptoms;

`azt` an indicator variable for AZT use;

`race` an indicator variable for the race (`w`=white, `b`=black).

## Source

The data were obtained from the *New York Times* (2/15/91).

## Examples

```
data(aids)
summary(aids)
```

---

`airway` *Airway Data*

---

## Description

The `airway` data frame has 35 rows and 6 columns.

Study to compare two devices (tracheal tube and laryngeal mask) used to secure airway in patients undergoing surgery. The response variable is the presence of a sore throat. Further information on age, sex, use of a lubricant, and duration of the surgery is available.

## Usage

```
data(airway)
```

## Format

This data frame contains the following columns:

`response` an indicator variable for sore throat (1=yes, 0=no);

`type` the type of airway used (1=tracheal tube, 0=laryngeal mask);

`age` the age of the patient (in years);

`sex` an indicator variable for sex (1=male, 0=female);

`lubricant` an indicator variable for lubricant use (1=yes, 0=no);

`duration` the duration of the surgery (in minutes).

## Source

The data were obtained from

"Binary Data" by D. Collet in *Encyclopedia of Biostatistics* (1998).

## Examples

```
data(airway)
summary(airway)
par(mfrow=c(1,2))
plot(age ~ response, data = airway)
plot(duration ~ response, data = airway)
```

`babies` *Crying Babies Data*

## Description

The `babies` data frame has 36 rows and 4 columns.

Matched pairs of binary observations concerning the crying of babies. The babies were observed on 18 days and on each day one child was lulled. Interest focuses on the treatment effect "lulling".

## Usage

```
data(babies)
```

## Format

This data frame contains the following columns:

`r1` number of children not crying on one day;

`r2` number of children crying on one day;

`lull` indicator variable for the treatment;

`day` factor variable for the days.

## Source

The data were obtained from

Cox, D. R. (1970) *Analysis of Binary Data* (page 61). London: Chapman \& Hall.

## References

Davison, A. C. (1988) Approximate conditional inference in generalized linear models. *J. R. Statist. Soc.* B, **50**, 445–461.

## Examples

```
data(babies)
coplot(r2/(r1+r2) ~ day | lull, data = babies)
##
babies.glm <- glm(formula = cbind(r1, r2) ~ day + lull - 1,
                  family = binomial, data = babies)
babies.cond <- cond(object = babies.glm, offset = lullyes)
babies.cond
```

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()
```

---

cond *Approximate Conditional Inference - Generic Function*

---

## Description

Performs approximate conditional inference.

## Usage

```
cond(object, offset, ...)
```

## Arguments

| | |
|---|---|
| `object` | a fitted model object. Families supported are binomial and Poisson with canonical link function (class `glm`), and regression-scale models (class `rsm`). |
| `offset` | the covariate occurring in the model formula whose coefficient represents the parameter of interest. May be numerical or a two-level factor. In case of a two-level factor, it must be coded by contrasts and not appear as two dummy variables in the model. Can also be a call to a mathematical function (such as `exp`, `sin`, `...`) or to a mathematical operator (`^`, `/`, `...`) applied to a numerical variable. The call must always agree with the label used to identify the corresponding parameter in the fitted model object passed through the `object` argument. Beware that the label includes the identity function `I()` if an arithmetic operator was used. Other function types (e.g. `factor`) and interactions are not admitted. |
| `...` | absorbs any additional arguments. See `cond.glm` and `cond.rsm` for details. |

## 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: `glm` and `rsm`.

## Value

The returned value is an *approximate conditional inference* object. Classes already supported are `cond` and `marg` depending on whether the fitted model object passed through the `object` argument has class `glm` or `rsm`. See `cond.object` or `marg.object` for more details.

## References

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

## See Also

cond.glm, cond.rsm, cond.object, marg.object

## Examples

```
## Urine Data
data(urine)
urine.glm <- glm(r ~ gravity + ph + osmo + conduct + urea + log(calc),
                 family = binomial, data = urine)
##
## function call as offset variable
labels(coef(urine.glm))
cond(urine.glm, log(calc))
##
## large estimate of regression coefficient
urine.glm <- glm(r ~ gravity + ph + osmo + conduct + urea + calc,
                 family = binomial, data = urine)
coef(urine.glm)
urine.glm <- glm(r ~ I(gravity * 100) + ph + osmo + conduct + urea + calc,
                 family = binomial, data = urine)
coef(urine.glm)
urine.cond <- cond(urine.glm, I(gravity * 100))
plot(urine.cond, which = 4)

## House Price Data
## Not run:
data(houses)
houses.rsm <- rsm(price ~ ., family = student(5), data = houses)
##
## parameter of interest: scale parameter
houses.marg <- cond(houses.rsm, scale)
plot(houses.marg, which = 2)

## End(Not run)
```

---

cond.glm *Approximate Conditional Inference for Logistic and Loglinear Models*

---

## Description

Performs approximate conditional inference on a scalar parameter of interest in logistic and loglinear models. The output is stored in an object of class cond.

## Usage

```
## S3 method for class 'glm'
cond(object, offset, formula = NULL, family = NULL,
         data = sys.frame(sys.parent()), pts = 20,
         n = max(100, 2*pts), tms = 0.6, from = NULL, to = NULL,
         control = glm.control(...), trace = FALSE, ...)
```

## Arguments

| | |
|---|---|
| object | a glm object. Families supported are binomial and Poisson with canonical link function. |
| offset | the covariate occurring in the model formula whose coefficient represents the parameter of interest. May be numerical or a two-level factor. In case of a two-level factor, it must be coded by contrasts and not appear as two dummy variables in the model. Can also be a call to a mathematical function (such as exp, sin, ...) or to a mathematical operator (^, /, ...) applied to a numerical variable. The call must always agree with the label used to identify the corresponding parameter in the glm object passed through the object argument or defined by formula and family. Beware that the label includes the identity function I() if an arithmetic operator was used. Other function types (e.g. factor) and interactions are not admitted. |
| formula | a formula expression (only if no glm object is defined). |
| family | a family object defining the variance function (only if no glm object is defined). Families supported are binomial and Poisson with canonical link function. |
| data | an optional data frame in which to interpret the variables occurring in the formula (only if no glm object is defined). |
| pts | number of output points (minimum 10) that are calculated exactly. The default is 20. |
| n | approximate number of output points (minimum 50) produced by the spline interpolation. The default is the maximum between 100 and twice pts. |
| tms | defines the range MLE +/- tms * S.E. where cubic spline interpolation is replaced by polynomial interpolation. The default is 0.6. |
| from | starting value of the sequence that contains the values of the parameter of interest for which output points are calculated exactly. The default is MLE - 3.5 * S.E. |
| to | ending value of the sequence that contains the values of the parameter of interest for which output points are calculated exactly. The default is MLE + 3.5 * S.E. |
| control | a list of iteration and algorithmic constants that controls the GLM fit. See \ glm.control for their names and default values. |
| trace | if TRUE, iteration numbers will be printed. |
| ... | additional arguments, such as subset etc., used by the glm fitting routine if the glm object is defined through formula and family. See glm for their definition and use. The arguments weights, offset and contrasts are not admitted. The returned value is an object of class cond; see cond.object for details. |

## Details

This function is a method for the generic function cond for class glm. It can be invoked by calling cond for an object of the appropriate class, or directly by calling cond.glm regardless of the class of the object. cond.glm has also to be used if the glm object is not provided throught the object argument but specified by formula and family.

The function cond.glm implements several small sample asymptotic methods for approximate conditional inference in logistic and loglinear models. Approximations for both the conditional log likelihood function and conditional tail area probabilities are available (see cond.object for details). Attention is restricted to a scalar parameter of interest. The associated covariate can be either numerical or a two-level factor.

Approximate conditional inference is performed by either updating a fitted generalized linear model or defining the model formula and family. All approximations are calculated exactly for pts equally spaced points ranging from from to to. A cubic spline interpolation is used to extend them over the whole interval of interest, except for the range of values defined by MLE +/- tms * S.E. where the spline interpolation is replaced by a higher order polynomial interpolation. This is done in order to avoid numerical instabilities which are likely to occur for values of the parameter of interest close to the MLE. Results are stored in an object of class cond. Method functions like print, summary and plot can be used to examine the output or represent it graphically. Components can be extracted using coef, formula and family.

Main references for the methods considered are the papers by *Pierce and Peters (1992)* and *Davison (1988)*. More details on the implementation are given in *Brazzale (1999, 2000)*.

## Value

The returned value is an object of class cond; see cond.object for details.

## Note

In rare occasions, cond.glm dumps because of non-convergence of the function glm which is used to refit the model for a fixed value of the parameter of interest. This happens for instance if this value is too extreme. The arguments from and to may then be used to limit the default range of MLE +/- 3.5 * S.E. A further possibility is to fine-tuning the constants (number of iterations, convergence threshold) that control the GLM fit through the control argument.

cond.glm may also dump if the estimate of the parameter of interest is large (tipically > 400) in absolute value. This may be avoided by reparametrizing the model.

The output of cond.glm may be unreliable if part of the data have a degenerate distribution. For example take the fungal infections treatment data contained in the fungal data frame. Of the five $2 \times 2$ contingency tables, two (the first and the third) are degenerate. As they make no contribution to the exact conditional likelihood, they should be omitted from the approximate conditional fit.

## References

Brazzale, A. R. (1999) Approximate conditional inference for logistic and loglinear models. *J. Comput. Graph. Statist.*, **8**, 1999, 653–661.

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

Davison, A. C. (1988) Approximate conditional inference in generalized linear models. *J. R. Statist. Soc.* B, **50**, 445–461.

Pierce, D. A. and Peters, D. (1992) Practical use of higher order asymptotics for multiparameter exponential families (with Discussion). *J. R. Statist. Soc.* B, **54**, 701–737.

## See Also

cond.object, summary.cond, plot.cond, glm

## Examples

```
## Crying Babies Data
data(babies)
babies.glm <- glm(formula = cbind(r1, r2) ~ day + lull - 1,
                  family = binomial, data = babies)
babies.cond <- cond(object = babies.glm, offset = lullyes)
babies.cond

## Urine Data
## (function call as offset variable)
data(urine)
urine.glm <- glm(r ~ gravity + ph + osmo + conduct + urea + log(calc),
                 family = binomial, data = urine)
labels(coef(urine.glm))
urine.cond <- cond(urine.glm, log(calc))
##
## (large estimate of regression coefficient)
urine.glm <- glm(r ~ gravity + ph + osmo + conduct + urea + calc,
                 family = binomial, data = urine)
coef(urine.glm)
urine.glm <- glm(r ~ I(gravity * 100) + ph + osmo + conduct + urea + calc,
                 family = binomial, data = urine)
coef(urine.glm)
urine.cond <- cond(urine.glm, I(gravity * 100))

## Fungal Infections Treatment Data (numerical instabilities around the
##                                   MLE)
## (full data analysis)
data(fungal)
fungal.glm <- glm(cbind(success, failure) ~ center + group - 1,
                  family = binomial, data = fungal,
                  control = glm.control(maxit = 50, epsilon = 1e-005))
fungal.cond <- cond(fungal.glm, groupT)
plot(fungal.cond, which = 2)
## (partial data analysis)
fungal.glm <- glm(cbind(success, failure) ~ center + group - 1,
                  family = binomial, data = fungal, subset = -c(1,2,5,6),
                  control = glm.control(maxit = 50, epsilon = 1e-005))
fungal.cond <- cond(fungal.glm, groupT)
plot(fungal.cond, which = 2)
## (Tables 1 and 3 are omitted).
```

`cond.object` *Approximate Conditional Inference Object*

## Description

Class of objects returned when performing approximate conditional inference for logistic and log-linear models.

## Arguments

| | |
|---|---|
| | Objects of class `cond` are implemented as a list. The following components are included: |
| | a list whose elements are the spline interpolations of several first order and higher order statistics. They are used to implement the following likelihood quantities:<br>- the profile and modified profile log likelihoods;<br>- the Wald pivots from the unconditional and conditional MLEs;<br>- the profile and modified likelihood roots (the latter one with a suitable continuity correction);<br>- the Lugannani-Rice tail area approximation (with suitable continuity correction);<br>- the correction term used in the higher order statistics;<br>- the information and nuisance parameter aspects.<br>Method functions work mainly on this part of the object. In order to avoid errors in the calculation of confidence intervals and tail probabilities, this part of the object should not be modified. |
| `coefficients` | a $2 \times 2$ matrix containing the unconditional and approximate conditional MLEs and their standard errors. |
| `call` | function call that created the `cond` object. |
| `formula` | the model formula. |
| `family` | the variance function. |
| `offset` | the covariate occurring in the model formula whose coefficient represents the parameter of interest. |
| `diagnostics` | diagnostics related to the decomposition of the higher order adjustments into an information and a nuisance parameters term. A value larger than 0.2 in absolute value is an index that higher order methods are needed. See *Pierce and Peters (1992)* for details. |
| `n.approx` | number of output points that have been calculated exactly. |
| `omitted.val` | range of values omitted in the spline interpolation of some of the higher order statistics considered. The aim is to avoid numerical instabilities around the maximum likelihood estimate. |

`is.scalar` a logical value indicating whether there are any nuisance parameters. If `FALSE` there are none.

Main references for the methods considered are the papers by *Pierce and Peters (1992)* and *Davison (1988)*. More details on the implementation and the methods considered are given in *Brazzale (1999, 2000)*.

## Generation

This class of objects is returned from calls to the function `cond.glm`.

## Methods

The class cond has methods for `summary`, `plot`, `print`, `coef` and `family`, amongst others.

## References

Brazzale, A. R. (1999) Approximate conditional inference for logistic and loglinear models. *J. Comput. Graph. Statist.*, **8**, 653–661.

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

Davison, A. C. (1988) Approximate conditional inference in generalized linear models. *J. R. Statist. Soc.* B, **50,** 445–461.

Pierce, D. A. and Peters, D. (1992) Practical use of higher order asymptotics for multiparameter exponential families (with Discussion). *J. R. Statist. Soc.* B, **54**, 701–737.

## See Also

`cond.glm`, `summary.cond`, `plot.cond`

---

# cond.rsm *Approximate Conditional Inference in Regression-Scale Models*

---

## Description

Performs approximate conditional inference on a scalar parameter of interest in regression-scale models. The output is stored in an object of class `marg`.

## Usage

```
## S3 method for class 'rsm'
cond(object, offset, formula = NULL, family = NULL,
     dispersion = NULL, data = sys.frame(sys.parent()), pts = 20,
     n = max(100, 2*pts), tms = 0.6, from = NULL, to = NULL,
     control = glm.control(...), trace = FALSE, ...)
```

## Arguments

| | |
|---|---|
| object | a rsm object; for instance the result of a call to rsm. |
| offset | either the covariate occurring in the model formula whose coefficient represents the parameter of interest or scale if the parameter of interest is the scale parameter. In the first case, the variable may be numerical or a two-level factor. In case of a two-level factor, it must be coded by contrasts and not as two dummy variables. Can also be a call to a mathematical function (such as exp, sin, ...) or to a mathematical operator (^, /, ...) applied to a numerical variable. The call must always agree with the label used to identify the corresponding parameter in the rsm object passed through the object argument or defined by formula and family. Beware that the label includes the identity function I() if an arithmetic operator was used. Other function types (e.g. factor) and interactions are not admitted. If interest focuses on the scale parameter, it must not be fixed in object or when using the dispersion argument in case no rsm object is supplied. |
| formula | a formula expression (only if no rsm object is defined). |
| family | a family.rsm object defining the error distribution (only if no rsm object is defined). See rsm.families for the families supported. |
| dispersion | argument only to be used if no rsm object is defined. If NULL, the scale parameter is taken to be unknown. If known, the numerical value can be passed. The default is NULL. Huber's least favourable error distribution represents a special case. If dispersion is NULL, the maximum likelihood estimate is computed, while if TRUE the MAD estimate is calculated and the scale parameter fixed to this value in subsequent computations. |
| data | an optional data frame in which to interpret the variables occurring in the formula (only if no rsm object is defined). |
| pts | number of output points (minimum 10) that are calculated exactly; the default is 20. |
| n | approximate number of output points (minimum 50) produced by the spline interpolation. The default is the maximum between 100 and twice pts. |
| tms | defines the range MLE +/- tms * S.E. where cubic spline interpolation is replaced by polynomial interpolation. The default is 0.6. |
| from | starting value of the sequence that contains the values of the parameter of interest for which output points are calculated exactly. The default is MLE - 3.5 * S.E. |
| to | ending value of the sequence that contains the values of the parameter of interest for which output points are calculated exactly. The default is MLE + 3.5 * S.E. |
| control | a list of iteration and algorithmic constants that control the rsm fit. See glm.control for their names and default values. |
| trace | if TRUE, iteration numbers will be printed. |
| ... | additional arguments, such as weights, subset, control etc. used by the rsm fitting routine if the rsm object is defined through formula and family. See rsm for their definition and use. |

## Details

This function is a method for the generic function cond for class rsm. It can be invoked by calling cond for an object of the appropriate class, or directly by calling cond.rsm regardless of the class of the object. cond.rsm has also to be used if the rsm object is not provided throught the object argument but specified by formula and family.

The function cond.rsm implements several small sample asymptotic methods for approximate conditional inference in regression-scale models. Approximations for both the modified/marginal log likelihood function and approximate conditional/marginal tail probabilities are available (see marg.object for details). Attention is restricted to a scalar parameter of interest, either a regression coefficient or the scale parameter. In the first case, the associated covariate may be either numerical or a two-level factor.

Approximate conditional (or equivalently marginal) inference is performed by either updating a fitted regression-scale model or defining the model formula and family. All approximations are calculated exactly for pts equally spaced points ranging from from to to. A spline interpolation is used to extend them over the whole interval of interest, except for the range of values defined by MLE +/- tms * S.E. where the spline interpolation is replaced by a higher order polynomial interpolation. This is done in order to avoid numerical instabilities which are likely to occur for values of the parameter of interest close to the MLE. Results are stored in an object of class marg. Method functions like print, summary and plot can be used to examine the output or represent it graphically. Components can be extracted using coef, formula and family.

Main references for the methods considered are the papers by *Barndorff-Nielsen (1991)*, *DiCiccio, Field and Fraser (1990)* and *DiCiccio and Field (1991)*. 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 6.3.1)*.

## Value

The returned value is an object of class marg; see marg.object for details.

## Note

If the parameter of interest is the scale parameter, all calculations are performed on the logarithmic scale, though most results are reported on the original scale.

In rare occasions, cond.rsm dumps because of non-convergence of the function rsm which is used to refit the model for a fixed value of the parameter of interest. This happens for instance if this value is too extreme. The arguments from and to may then be used to limit the default range of MLE +/- 3.5 * S.E. A further possibility is to fine-tuning the constants (number of iterations, convergence threshold) that control the rsm fit through the control argument.

cond.rsm may also dump if the estimate of the parameter of interest is large (tipically > 400) in absolute value. This may be avoided by reparametrizing the model.

## References

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

Brazzale, A. R. (1999) Approximate conditional inference for logistic and loglinear models. *J. Comput. Graph. Statist.*, **8**, 653–661.

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

DiCiccio, T. J., Field, C. A. and Fraser, D. A. S. (1990) Approximations of marginal tail probabilities and inference for scalar parameters. *Biometrika*, **77**, 77–95.

DiCiccio, T. J. and Field, C. A. (1991) An accurate method for approximate conditional and Bayesian inference about linear regression models from censored data. *Biometrika*, **78**, 903–910.

## See Also

`marg.object`, `summary.marg`, `plot.marg`, `rsm`

## Examples

```
## Sea Level Data
data(venice)
attach(venice)
Year <- 1:51/51
c11 <- cos(2*pi*1:51/11) ; s11 <- sin(2*pi*1:51/11)
c19 <- cos(2*pi*1:51/18.62) ; s19 <- sin(2*pi*1:51/18.62)
##
## quadratic model fitted to the sea level, includes 18.62-year
## astronomical tidal cycle and 11-year sunspot cycle
venice.rsm <- rsm(sea ~ Year + I(Year^2) + c11 + s11 + c19 + s19,
                  family = extreme)
names(coef(venice.rsm))
## "(Intercept)"  "Year"  "I(Year^2)"  "c11"  "s11"  "c19"  "s19"
##
## variable of interest: quadratic term
venice.marg <- cond(venice.rsm, I(Year^2))
##
detach()

## House Price Data
data(houses)
houses.rsm <- rsm(price ~ ., family = student(5), data = houses)
##
## parameter of interest: scale parameter
houses.marg <- cond(houses.rsm, scale)
```

---

# 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()
```

---

# darwin *Darwin's Data on Growth Rates of Plants*

---

## Description

The `darwin` data frame has 15 rows and 3 columns.

Charles Darwin conducted an experiment to examine the superiority of cross-fertilized plants over self-fertilized plants. 15 pairs of plants were used. Each pair consisted of one cross-fertilized plant and one self-fertilized plant which germinated at the same time and grew in the same pot. The heights of the plants were measured at a fixed time after planting. Four different pots were used.

## Usage

```
data(darwin)
```

## Format

This data frame contains the following columns:

`cross` the heights of the cross-fertilized plants (in inches);

`self` the heights of the self-fertilized plants (in inches);

`pot` a factor variable for the pot number.

## Source

The data were obtained from

Andrews, D. F. and Herzberg, A. M. (1985) *Data: A Collection of Problems From Many Fields for the Student and Research Worker* (Chapter 2). New York: Springer-Verlag.

## References

Darwin, C. (1878) *The Effects of Cross and Self Fertilisation in the Vegetable Kingdom* (2nd ed.). London: John Murray.

## Examples

```
data(darwin)
plot(cross - self ~ pot, data = darwin)
```

---

# 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
## ...
detach()
```

---

`dormicum` *Dormicum Data*

---

## Description

The `dormicum` data frame has 37 rows and 3 columns.

37 children in a pediatric intensive care unit were treated with varying doses and for varying duration with the drug *Dormicum*. The response variable is 1 if withdrawal symptoms were exhibited and 0 otherwise.

## Usage

```
data(dormicum)
```

## Format

This data frame contains the following columns:

`symp` indicator of the presence of withdrawal symptoms;

`dose` the drug dose in mg/kg;

`days` the number of days treated.

## Source

The data were supplied by *Spadille Biostatistik*, Denmark.

## References

Mehta, C. R., Patel, N. T. and Senchaudhuri, P. (2000) Efficient Monte Carlo methods for conditional logistic regression. *J. Amer. Statist. Ass.*, **95**, 99–108.

## Examples

```
data(dormicum)
par(mfrow = c(1,2))
plot(dose ~ symp, data = dormicum, xlab = "presence of withdrawal symptoms",
     ylab = "treatment dose (mg/kg)")
plot(days ~ symp, data = dormicum, xlab = "presence of withdrawal symptoms",
     ylab = "treatment days")
```

---

# 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
## ...
##
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

---

## family.cond *Use family() on a "cond" object*

---

### Description

This is a method for the function family() for objects inheriting from class cond. See family for the general behaviour of this function.

### Usage

```
## S3 method for class 'cond'
family(object, ...)
```

### Arguments

| | |
|---|---|
| object | any object from which a family object can be extracted. |
| ... | absorbs any additional argument. |

### See Also

family

---

## family.rsm *Use family() on a "rsm" object*

---

### Description

This is a method for the function family() for objects from which a family.rsm object can be extracted. Typically a fitted rsm model object. See family for the general behaviour of this function.

### Usage

```
## S3 method for class 'rsm'
family(object, ...)
```

### Arguments

| | |
|---|---|
| object | any object from which a family.rsm object can be extracted. |
| ... | absorbs any additional argument. |

### See Also

family.rsm.object, family

### Examples

```
## Sea Level Data
data(venice)
attach(venice)
Year <- 1:51/51
c11 <- cos(2*pi*1:51/11) ; s11 <- sin(2*pi*1:51/11)
c19 <- cos(2*pi*1:51/18.62) ; s19 <- sin(2*pi*1:51/18.62)
venice.rsm <- rsm(sea ~ Year + I(Year^2) + c11 + s11 + c19 + s19,
                  family = extreme)
family(venice.rsm)
detach()
```

```
## House Price Data
data(houses)
houses.rsm <- rsm(price ~ ., family = student(5), data = houses)
family(houses.rsm)
```

---

`family.rsm.object` *Family Object for Regression-Scale Models*

---

## Description

Class of objects that characterize the error distribution of a regression-scale model.

## Generation

This class of objects is returned by a call to a `family.rsm` generator function. See `rsm.families` for the distributions which are supported by the `marg` package. The object includes a list of functions and expressions that characterize the error distribution of a regression-scale model. These are used by the IRLS algorithm implemented in the `rsm` fitting routine. New families can be added to the ones already supported. See the demonstration file 'margdemo.R' that ships with the package. There is a `print` method for `family.rsm` objects which produces a simple summary without any detail; use `unclass(family.rsm.object)` to see the contents.

## Structure

The following components, with the corresponding functionality, are required for a `family.rsm` object:

`family` a character vector giving the family name.

`g0` a function that yields minus the log density of the error distribution in the regression-scale model.

`g1` a function that yields the first derivative of minus the log density.

`g2` a function that yields the second derivative of minus the log density.

`df` argument with `NULL` value; must be included to guarantee compatibility with the existing code.

`k` argument with `NULL` value; must be included to guarantee compatibility with the existing code.

## Note

For the sake of compatibility, the `g0`, `g1` and `g2` functions of a user-defined family can only take two arguments: `y` representing an observation and the `...` argument which absorbes any additional arguments.

## See Also

`rsm.families`, `family.rsm`, `rsm`

## family.summary.cond *Use family() on a "summary.cond" object*

### Description

This is a method for the function `family()` for objects inheriting from class `summary.cond`. See `family` for the general behaviour of this function.

### Usage

```
## S3 method for class 'summary.cond'
family(object, ...)
```

### Arguments

| | |
|---|---|
| `object` | any object from which a `family` object can be extracted. |
| `...` | absorbs any additional argument. |

### See Also

`family`

## fraudulent *Fraudulent Automobile Insurance Claims Data*

### Description

The `fraudulent` data frame has 42 rows and 12 columns.

127 claims arising from automobile accidents in 1989 in Massachusetts (USA). Each claim was classified as either fraudulent or legitimate by consensus among four independent claims adjusters who examined each case file thoroughly. An exploratory analysis by Derrig and Weisberg (1993) identified 10 binary indicators, each of which denotes the presence or absence of a potential fraud characteristic in the claim situation. They fall into three broad groups relating to "Accident" (AC1, AC9 and AC16), "Claimant" (CL7 and CL11), and "Injury" (IJ2, IJ3, IJ4, IJ6 and IJ12).

### Usage

```
data(fraudulent)
```

## Format

This data frame contains the following columns:

`r1` the number of frauds detected;

`r2` the total number of automobile insurance claims;

`AC1`,`AC9`,`AC16` potential fraud characteristics pertaining to "Accident". The presence of the fraud characteristic is indicated by a 1, the absence is indicated by a 0.

`CL7`,`CL11` potential fraud characteristics pertaining to "Claimer". The presence of the fraud characteristic is indicated by a 1, the absence is indicated by a 0.

`IJ2`,`IJ3`,`IJ4`,`IJ6`,`IJ12` potential fraud characteristics pertaining to "Injury". The presence of the fraud characteristic is indicated by a 1, the absence is indicated by a 0.

## Source

The data were supplied by Dr. Richard Derrig of the Automobile Insurers Bureau of Massachusetts.

## References

Mehta, C. R., Patel, N. T. and Senchaudhuri, P. (2000) Efficient Monte Carlo methods for conditional logistic regression. *J. Amer. Statist. Ass.*, **95**, 99–108.

Derrig, R. A. and Weisberg, H. I. (1993). Quantitative methods for detecting fraudulent automobile bodily injury claims. *Manuscript.*

## Examples

```
data(fraudulent)
summary(fraudulent)
```

---

# fungal *Fungal Infections Treatment Data*

---

## Description

The `fungal` data frame has 10 rows and 4 columns.

Clinical trial on the success of a particular treatment for fungal infections. The study was carried out in five different research units. Interest focuses on the treatment effect.

## Usage

```
data(fungal)
```

## Format

This data frame contains the following columns:

`success` the number of patients that benefited from the treatment;

`failure` the number of patients with no benefit from the treatment;

`group` an indicator variable for treatment (`T`=treatment, `P`=placebo);

`center` a factor variable indicating the research unit where the study was carried out.

## Source

The data were supplied by *Sandoz Pharmaceuticals.*

## Examples

```
## (full data analysis)
data(fungal)
fungal.glm <- glm(cbind(success, failure) ~ center + group - 1,
                  family = binomial, data = fungal,
                  control = glm.control(maxit = 50, epsilon = 1e-005))
fungal.cond <- cond(fungal.glm, groupT)
plot(fungal.cond, which = 2)
## (partial data analysis)
fungal.glm <- glm(cbind(success, failure) ~ center + group - 1,
                  family = binomial, data = fungal, subset = -c(1,2,5,6),
                  control = glm.control(maxit = 50, epsilon = 1e-005))
fungal.cond <- cond(fungal.glm, groupT)
plot(fungal.cond, which = 2)
## (Tables 1 and 3 are omitted).
```

---

# `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 ) )
```

---

# hoa *Higher Order Likelihood Inference*

---

## Description

Performs likelihood-based inference for a wide range of regression models. Provides higher-order approximations for inference based on extensions of saddlepoint type arguments as discussed in the book Applied Asymptotics: Case Studies in Small-Sample Statistics by Brazzale, Davison, and Reid (2007).

## Details

| | |
|---|---|
| Package: | hoa |
| Version: | 2.1.2 |
| Date: | 2015-07-13 |
| Depends: | R (>= 3.0.0) |
| Imports: | graphics, statmod, stats, survival, utils |
| Suggests: | boot, csampling |
| License: | GPL (>= 2) |
| URL: | http://www.r-project.org, http://statwww.epfl.ch/AA/ |
| LazyLoad: | yes |
| LazyData: | yes |

Index:

Functions:
=========


| | |
|---|---|
| cond | Approximate Conditional Inference - Generic Function |
| cond.glm | Approximate Conditional Inference for Logistic and Loglinear Models |
| cond.object | Approximate Conditional Inference Object |
| family.cond | Use family() on a "cond" object |
| family.summary.cond | Use family() on a "summary.cond" object |
| plot.cond | Generate Plots for an Approximate Conditional Inference Object |
| print.summary.cond | Use print() on a "summary.cond" object |
| summary.cond | Summary Method for Objects of Class "cond" Function |
| cond.rsm | Approximate Conditional Inference in Regression-Scale Models |
| dHuber | Huber's Least Favourable Distribution |
| family.rsm | Use family() on a "rsm" object |
| family.rsm.object | Family Object for Regression-Scale Models |
| logLik.rsm | Compute the Log Likelihood for Regression-Scale Models |
| marg.object | Approximate Marginal Inference Object |
| plot.fr | Plot a Fraser-Reid Object |
| plot.marg | Generate Plots for an Approximate Marginal Inference Object |
| print.summary.marg | Use print() on a "summary.marg" object |
| residuals.rsm | Compute Residuals for Regression-Scale Models |
| rsm | Fit a Regression-Scale Model |
| rsm.diag | Diagnostics for Regression-Scale Models |
| rsm.diag.plots | Diagnostic Plots for Regression-Scale Models |
| rsm.families | Generate a RSM Family Object |
| rsm.fit | Fit a Regression-Scale Model Without Computing the Model Matrix |
| rsm.null | Fit an Empty Regression-Scale Model |

```
rsm.object              Regression-Scale Model Object
rsm.surv                Fit a Regression-Scale Model Without Computing
                        the Model Matrix
summary.marg            Summary Method for Objects of Class "marg"
summary.rsm             Summary Method for Regression-Scale Models
tem                     Tangent exponential model Higher Order Likelihood
                        Approximation
update.rsm              Update and Re-fit a RSM Model Call
vcov.rsm                Calculate Variance-Covariance Matrix for a
                        Fitted RSM Model

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.fr              Likelihood-Based Confidence Intervals Based on
```

```
                        Fraser-Reid Object
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:
========
aids                    AIDS Symptoms and AZT Use Data
airway                  Airway Data
babies                  Crying Babies Data
darwin                  Darwin's Data on Growth Rates of Plants
dormicum                Dormicum Data
fraudulent              Fraudulent Automobile Insurance Claims Data
fungal                  Fungal Infections Treatment Data
houses                  House Price Data
nuclear                 Nuclear Power Station Data
rabbits                 Rabbits Data
urine                   Urine Data
venice                  Sea Level Data
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 for Higher Order Likelihood Inference (source, pdf)
```

## Author(s)

R port by Alessandra R. Brazzale <alessandra.brazzale@unimore.it>, following earlier work by Douglas Bates. The function tem is based on work by Anthony Davison <Anthony.Davison@epfl.ch>

Maintainer: Alex-Antoine Fortin <alex@fortin.bio>

houses *House Price Data*

## Description

The houses data frame has 26 rows and 5 columns.

Ms. Terry Tasch of Long-Kogan Realty, Chicago, provides data on the selling prices of houses and on different variables describing their status. This data frame contains the prices and a subset of the covariates.

## Usage

```
data(houses)
```

## Format

This data frame contains the following columns:

`price` selling price (in thousands of dollars);

`bdroom` number of bedrooms;

`floor` floor space (in square feet);

`rooms` total number of rooms;

`front` front footage of lot (in feet).

## Source

The data were obtained from

Sen, A. and Srivastava, M. (1990) *Regression Analysis: Theory, Methods and Applications* (Exhibit 2.2, page 32). New York: Springer-Verlag.

## Examples

```
data(houses)
summary(houses)
pairs(houses)
```

Huber *Huber's Least Favourable Distribution*

## Description

Density, cumulative distribution, quantiles and random number generator for Huber's least favourable distribution.

## Usage

```
dHuber(x, k = 1.345)
pHuber(q, k = 1.345)
qHuber(p, k = 1.345)
rHuber(n, k = 1.345)
```

## Arguments

| | |
|---|---|
| x | vector of quantiles. Missing values (NAs) are allowed. |
| q | vector of quantiles. Missing values (NAs) are allowed. |
| p | vector of probabilities. Missing values (NAs) are allowed. |
| n | sample size. If length(n) is larger than 1, then length(n) random values are returned. |
| k | tuning constant. Values should preferably lie between 1 and 1.5. The default is 1.345, which gives 95% efficiency at the Normal. |

## Details

Inversion of the cumulative distribution function is used to generate deviates from Huber's least favourable distribution.

## Value

Density (dHuber), probability (pHuber), quantile (qHuber), or random sample (rHuber) for Huber's least favourable distribution with tuning constant k. If values are missing, NAs will be returned.

## Side Effects

The function rHuber causes creation of the dataset .Random.seed if it does not already exist; otherwise its value is updated.

## Background

Huber's least favourable distribution is a compound distribution with gaussian behaviour in the interval (-k,k) and double exponential tails. It is strongly related to Huber's M-estimator, which represents the maximum likelihood estimator of the location parameter.

## References

Hampel, F. R., Ronchetti, E. M., Rousseeuw, P. J. and Stahel, W. A. (1986) *Robust Statistics: The Approach Based on Influence Functions*. New York: Wiley.

## Examples

```
pHuber(0.5)
## 0.680374
pHuber(0.5, k = 1.5)
## 0.6842623
```

---

`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 )
```

---

# logLik.rsm *Compute the Log Likelihood for Regression-Scale Models*

---

## Description

Computes the log likelihood for regression-scale models.

## Usage

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

## Arguments

| | |
|---|---|
| `object` | an object inheriting from class `rsm` representing a fitted regression-scale model. |
| `...` | absorbs any additional argument. |

## Details

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

## Value

Returns an object class `logLik` which is a number with attributes, `attr(r, "df")` (degrees of freedom) giving the number of parameters (regression coefficients plus scale parameter, if not fixed) in the model.

## Note

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

## See Also

`rsm.object`, `logLik`

## Examples

```
## Sea Level Data
data(venice)
attach(venice)
Year <- 1:51/51
c11 <- cos(2*pi*1:51/11) ; s11 <- sin(2*pi*1:51/11)
c19 <- cos(2*pi*1:51/18.62) ; s19 <- sin(2*pi*1:51/18.62)
venice.rsm <- rsm(sea ~ Year + I(Year^2) + c11 + s11 + c19 + s19,
                  family = extreme)
##
logLik(venice.rsm)
detach()
```

---

`marg.object` *Approximate Marginal Inference Object*

---

## Description

Class of objects returned when performing approximate conditional inference for regression-scale models.

## Arguments

| | |
|---|---|
| | Objects of class `marg` are implemented as a list. The following components are included: |
| | a list whose elements are the spline interpolations of several first order and higher order statistics. They are used to implement the following likelihood quantities:<br>- the profile and modified profile/approximate marginal log likelihoods;<br>- the Wald pivots from the unconditional and conditional/approximate marginal MLEs;<br>- the profile and modified/approximate marginal likelihood roots;<br>- the conditional/approximate marginal Lugannani-Rice tail area approximation;<br>- the correction term used in the higher order statistics;<br>- the conditional/marginal information and nuisance parameter aspects.<br>Method functions work mainly on this part of the object. In order to avoid errors in the calculation of confidence intervals and tail probabilities, this part of the object should not be modified. |
| `coefficients` | a $2\times2$ matrix containing the unconditional and approximate conditional/marginal MLEs and their standard errors. |
| `call` | the function call that created the `marg` object. |
| `formula` | the model formula. |
| `family` | the name of the error distribution. |
| `offset` | the covariate occurring in the model formula whose coefficient represents the parameter of interest or `scale` if the parameter of interest is the scale parameter. |

| | |
|---|---|
| `diagnostics` | diagnostics related to the decomposition of the higher order adjustments into an information and a nuisance parameters term. |
| `n.approx` | the number of output points for which the statistics have been calculated exactly. |
| `omitted.val` | the range of values omitted in the spline interpolation of some of the higher order statistics considered. The aim is to avoid numerical instabilities around the maximum likelihood estimate. |
| `is.scalar` | a logical value indicating whether there are any nuisance parameters. If `FALSE` there are none. |

Main references for the methods considered are the papers by *Barndorff-Nielsen (1991)*, *DiCiccio, Field and Fraser (1990)* and *DiCiccio and Field (1991)*. The theory and statistics used are summarized in *Brazzale (2000, Chapters 2 and 3)*. More details of the implementation and of the methods considered are given in *Brazzale (1999; 2000, Section 6.3.1)*.

## Generation

This class of objects is returned from calls to the function `cond.rsm`.

## Methods

The class `marg` has methods for `summary`, `plot`, `print`, `coef` and `family`, among others.

## Note

If the parameter of interest is the scale parameter, all calculations are performed on the logarithmic scale, though most results are reported on the original scale.

## References

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

Brazzale, A. R. (1999) Approximate conditional inference for logistic and loglinear models. *J. Comput. Graph. Statist.*, **8**, 653–661.

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

DiCiccio, T. J., Field, C. A. and Fraser, D. A. S. (1990) Approximations of marginal tail probabilities and inference for scalar parameters. *Biometrika*, **77**, 77–95.

DiCiccio, T. J. and Field, C. A. (1991) An accurate method for approximate conditional and Bayesian inference about linear regression models from censored data. *Biometrika*, **78**, 903–910.

## See Also

`cond.rsm`, `summary.marg`, `plot.marg`

`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: |
| | 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<br>Var(error) = s^2 V(predictor)<br>where the constant term $\sigma^2$ 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 $\sigma^2$ 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 $\sigma^2$ in

Var(error) = s^2 V(predictor) is included by default. In order to work with a real value, $\sigma^2$ 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 $\sigma^2$ 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 $i$th 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 $i$th 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 $i$th 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:

| | |
|---|---|
| | 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. |
| coefVar | 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

---

nuclear *Nuclear Power Station Data*

---

## Description

This data frame contains data on the construction of 32 light water reactors constructed in the USA.

## Usage

```
data(nuclear)
```

## Format

A data frame with 32 observations on the following 11 variables.

`cost` cost of construction (in billions of dollars adjusted to a 1976 base)

`date` date at which the construction permit was issued

`T1` time between application for and issue of permit

`T2` time between issue of operating license and construction permit

`cap` power plant capacity (in MWe)

`PR` 1 if light water reactor already present on site

`NE` 1 if constructed in north-east region of USA

`CT` 1 if cooling tower used

`BW` 1 if nuclear steam supply system manufactured by Babcock-Wilcox

`N` cumulative number of power plants constructed by each architect-engineer

`PT` 1 if partial turnkey plant

## Source

The data were obtained from

Cox, D.R. and Snell, E.J. (1981). *Applied Statistics* (page 81, Example G). Chapman and Hall, London.

## Examples

```
data(nuclear)
```

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.cond *Generate Plots for an Approximate Conditional Inference Object*

## Description

Creates a set of plots for an object of class cond.

## Usage

```
## S3 method for class 'cond'
plot(x = stop("nothing to plot"), from = x.axis[1], to = x.axis[n],
     which = NULL, alpha = 0.05, add.leg = TRUE, loc.leg = FALSE,
     add.labs = TRUE, cex = 0.7, cex.lab = 1, cex.axis = 1,
     cex.main = 1, lwd1 = 1, lwd2 = 2, lty1 = "solid",
     lty2 = "dashed", col1 = "black", col2 = "blue", tck = 0.02,
     las = 1, adj = 0.5, lab = c(15, 15, 5), ...)
```

## Arguments

| | |
|---|---|
| x | a cond object. This is assumed to be the result returned by the cond.glm function. |
| from | starting value for the x-axis range. The default value has been set by cond.glm. |
| to | ending value for the x-axis range. The default value has been set by cond.glm. |
| which | which plot should be printed. Admissible values are 2 to 8 corresponding to the choices in the menu below. |
| alpha | the level used to read off confidence intervals; default is 5%. |
| add.leg | if TRUE, a legend is added to each plot; default is TRUE. |
| loc.leg | if TRUE, position of the legend can be located by hand; default is FALSE. |
| add.labs | if TRUE, labels are added; default is TRUE. |
| cex, cex.lab, cex.axis, cex.main | 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 | line width 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 type used to compare different curves in the same plot; default is lty2 = "dashed" for the Wald statistic and lty1 = "solid" for the remaining first- and higher order statistics. |
| col1, col2 | colors used to compare different curves in the same plot; default is col2 = "blue" for higher order solutions, and col1 = "black" for the remaining first order statistics. |
| tck, las, adj, lab | further graphical parameters. See par for details. |
| ... | optional graphical parameters; see par for details. |

## Details

Several plots are produced for an object of class cond. A menu lists all the plots that can be produced. They may be one or all of the following ones:

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

1:plot: All
2:plot: Profile and modified profile log likelihoods
3:plot: Profile and modified profile likelihood ratios
4:plot: Profile and modified likelihood roots
5:plot: Modified and continuity corrected likelihood roots
6:plot: Lugannani-Rice approximations
7:plot: Confidence intervals
8:plot: Diagnostics based on INF/NP decomposition

Selection:
```

If no nuisance parameters are presented, a subset of the above pictures is produced. More details on the implementation are given in *Brazzale (1999, 2000)*.

This function is a method for the generic function `plot()` for class cond. It can be invoked by calling `plot` or directly `plot.cond` for an object of the appropriate class.

## Value

A plot is created on the current graphics device.

## Side Effects

The current device is cleared. When `add.leg = TRUE`, a legend is added to each plot, and if `loc.leg = TRUE`, it can be set by the user. All screens are closed, but not cleared, on termination of the function.

## Note

The diagnostic plots only represent a preliminary version and need further development.

The two panels on the right trace the information and nuisance correction terms, INF and NP, against the likelihood root statistic. These are generally smooth functions and used to approximate the information and nuisance parameter aspects as a function of the parameter of interest, as shown in the two panels on the left. This procedure has the advantage of largely eliminating the numerical instabilities that affect the statistics around the MLE. The circles in the two leftmost panels represent the limit of INF and NP at the MLE calculated exactly using numerical derivatives. All four pictures are intended to give an idea of the order of magnitude of the two correction terms while trying to deal with the numerical problems that likely occur for these kinds of data.

More details can be found in *Brazzale (2000, Appendix B.2)*.

## References

Brazzale, A. R. (1999) Approximate conditional inference for logistic and loglinear models. *J. Comput. Graph. Statist.*, **8**, 1999, 653–661.

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

`cond.glm`, `cond.object`, `summary.cond`

## Examples

```
## Crying Babies Data
data(babies)
babies.glm <- glm(formula = cbind(r1, r2) ~ day + lull - 1,
                  family = binomial, data = babies)
babies.cond <- cond(object = babies.glm, offset = lullyes)
## Not run:
plot(babies.cond)

## End(Not run)

## Urine Data
data(urine)
urine.glm <- glm(r ~ I(gravity * 100) + ph + osmo + conduct + urea + calc,
                 family = binomial, data = urine)
urine.cond <- cond(urine.glm, I(gravity * 100))
plot(urine.cond, which=4)
```

---

`plot.fr` *Plot a fr Object*

---

## Description

Produces plots to show how various pivots depend on the parameter of interest `psi` and showing its profile log likelihood and modified profile log likelihood. Two further plots are produced if `all=TRUE`, and tests are performed if `psi` is given a value.

## Usage

```
## S3 method for class 'fr'
plot(x, psi = NULL, all = FALSE, xl = "Interest parameter", ...)
```

## Arguments

| | |
|---|---|
| x | A Fraser-Reid object, the output of tem. |
| psi | Value of interest parameter psi for which tests may be required. |
| all | Logical, by default two plots are produced, but if all=T two additional plots are added. |
| xl | Character string containing label for abscissa of graphs. |
| ... | Absorbs additional graphical parameters. |

## Details

If psi=NULL, draws graphs showing: (top left) the likelihood root r, the modified likelihood root rstar, the Wald pivot z, and the likelihood modification q as functions of the parameter psi; (top right) the profile log likelihood and modified profile log likelihood as functions of psi; (lower left) the significance functions corresponding to the upper left panel; and (lower right) how the modification log(q/r)/r depends on r. If all=FALSE (the default) only the upper two graphs are plotted. The lower left panel is a transformation of that above it, and the lower right panel is intended as a possible diagnostic of failure of the asymptotic approximations; it should be a smooth function of r.

If psi is given a (single, scalar) value, then the significance functions for the Wald statistic, the likelihood root, and the modified likelihood root, are evaluated at this value, providing p-values for testing against larger values of psi.

## Value

See the details above.

## Author(s)

Anthony Davison <Anthony.Davison@epfl.ch>

## References

Brazzale, A. R., Davison, A. C. and Reid, N. (2007). *Applied Asymptotics: Case Studies in Small-Sample Statistics*. Cambridge University Press, Cambridge.

See also http://statwww.epfl.ch/AA.

## See Also

tem, summary.fr, lik.ci

## Examples

```
## See the examples to \link{tem}.
```

plot.marg *Generate Plots for an Approximate Marginal Inference Object*

## Description

Creates a set of plots for an object of class marg.

## Usage

```
## S3 method for class 'marg'
plot(x = stop("nothing to plot"), from = x.axis[1], to = x.axis[n],
     which = NULL, alpha = 0.05, add.leg = TRUE, loc.leg = FALSE,
     add.labs = TRUE, cex = 0.7, cex.lab = 1, cex.axis = 1,
     cex.main = 1, lwd1 = 1, lwd2 = 2, lty1 = "solid",
     lty2 = "dashed", col1 = "black", col2 = "blue", tck = 0.02,
     las = 1, adj = 0.5, lab = c(15, 15, 5), ...)
```

## Arguments

| | |
|---|---|
| x | a marg object. This is assumed to be the result returned by the cond.rsm function. |
| from | the starting value for the x-axis range. The default value has been set by cond.rsm. |
| to | the ending value for the x-axis range. The default value has been set by cond.rsm. |
| which | which plot to print. Admissible values are 2 to 7 corresponding to the choices in the menu below. |
| alpha | the level used to read off confidence intervals; the default is 5%. |
| add.leg | if TRUE, a legend is added to each plot; default is TRUE. |
| loc.leg | if TRUE, the position of the legend can be located by hand; default is FALSE. |
| add.labs | if TRUE, labels are added; default is TRUE. |
| 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 type used to compare different curves in the same plot; default is lty2 = "dashed" for the Wald statistic and lty1 = "solid" for the remaining first and higher order statistics. |
| col1, col2 | colors used to compare different curves in the same plot; default is col2 = "blue" for higher order solutions, and col1 = "black" for the remaining first order statistics. |
| tck, las, adj, lab | further graphical parameters. See par for details. |
| ... | optional graphical parameters; see par for details. |

## Details

Several plots are produced for an object of class `marg`. A menu lists all the plots that can be produced. They may be one or all of the following ones:

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

1: All
2: Profile and modified profile log likelihoods
3: Profile and modified profile likelihood ratios
4: Profile and modified likelihood root
5: Lugannani-Rice approximation
6: Confidence intervals
7: Diagnostics based on INF/NP decomposition

Selection:
```

If no nuisance parameters are presented, a subset of the above pictures is produced. A message is printed if this is the case. More details and examples are given in *Brazzale (2000, Sections 6.5 and 5.3.2)*.

This function is a method for the generic function `plot()` for class `marg`. It can be invoked by calling `plot` or directly `plot.marg` for an object of the appropriate class.

## Value

A plot is created on the current graphics device.

## Side Effects

The current device is cleared. When `add.leg` is `TRUE`, a legend is added to each plot. Furthermore, if `loc.leg` is `TRUE`, the location of the legend can be set by the user. All screens are closed, but not cleared, on termination of the function.

## Note

If the parameter of interest is the scale parameter, all calculations are performed on the log scale, though most results are reported on the original scale.

Four diagnostic plots are provided. The two panels on the right trace the information and nuisance correction terms, INF and NP, against the likelihood root statistic. These are generally smooth functions and used to approximate the information and nuisance parameter aspects as a function of the parameter of interest, as shown in the two panels on the left. This procedure has the advantage of largely eliminating the numerical instabilities that affect the statistics around the MLE. All four pictures are intended to give an idea of the order of magnitude of the two correction terms while trying to deal with the numerical problems that likely occur for these kinds of data.

More details can be found in *Brazzale (2000, Appendix B.2)*.

## References

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

cond.rsm, marg.object, summary.marg

## Examples

```
# Sea Level Data
data(venice)
attach(venice)
Year <- 1:51/51
c11 <- cos(2*pi*1:51/11) ; s11 <- sin(2*pi*1:51/11)
c19 <- cos(2*pi*1:51/18.62) ; s19 <- sin(2*pi*1:51/18.62)
#
# quadratic model fitted to the sea level, includes 18.62-year
# astronomical tidal cycle and 11-year sunspot cycle
venice.rsm <- rsm(sea ~ Year + I(Year^2) + c11 + s11 + c19 + s19,
                  family = extreme)
venice.marg <- cond(venice.rsm, I(Year^2))
plot(venice.marg, which = 4)
##
detach()
```

---

# 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), ...)

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), ...)
```

## 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)
```

---

`print.summary.cond` *Use print() on a "summary.cond" object*

---

## Description

This is a method for the function `print()` for objects inheriting from class `summary.cond`. See `print` and `print.default` for the general behaviour of this function and for the interpretation of `digits`.

## Usage

```
## S3 method for class 'summary.cond'
print(x, all = x$all, Coef = x$cf, int = x$int, test = x$hyp,
     digits = if(!is.null(x$digits)) x$digits else max(3, getOption("digits")-3),
      ...)
## S3 method for class 'summary.cond'
print(x, all, Coef, int, test, digits, ...)
```

## Arguments

| | |
|---|---|
| `x` | a `summary.cond` object. This is assumed to be the result returned by the `summary.cond` function. |
| `all` | if `TRUE` all the information stored in the `summary.cond` object is printed, else only a subset of it. The default is `FALSE`. |
| `Coef` | if `TRUE`, the unconditional and conditional parameter estimates are printed. The default is `TRUE`. |
| `int` | if `TRUE`, confidence intervals are printed. The default is `TRUE`. |
| `test` | if `TRUE`, tests statistics and tail probabilities are printed. The default is `FALSE`. |
| `digits` | number of significant digits to be printed. The default depends on the value of `digits` set by `options`. |
| `...` | additional arguments. |

## Details

Changing the default values of `all`, `Coef`, `int` and `test` allows only a subset of the information in the `summary.cond` object to be printed. With `all = FALSE`, one-sided confidence intervals and the Lugannani-Rice tail approximations are omitted. See `summary.cond` for more details.

## Note

The amount of information printed may vary depending on whether there are any nuisance parameters.

## See Also

summary.cond, cond.object, print.default

## Examples

```
## Urine Data
data(urine)
urine.glm <- glm(r ~ gravity + ph + osmo + conduct + urea + calc,
                 family = binomial, data = urine)
urine.cond <- cond(urine.glm, urea)
print(summary(urine.cond, all = TRUE), digits = 4)
print(summary(urine.cond), Coef = FALSE)
```

---

# print.summary.marg *Use print() on a "summary.marg" object*

---

## Description

This is a method for the function print() for objects of class summary.marg. See print and print.default for the general behaviour of this function and for the interpretation of digits.

## Usage

```
## S3 method for class 'summary.marg'
print(x, all = x$all, Coef = x$cf, int = x$int, test = x$hyp,
     digits = if(!is.null(x$digits)) x$digits else max(3, getOption("digits")-3),
      ...)
## S3 method for class 'summary.marg'
print(x, all, Coef, int, test, digits, ...)
```

## Arguments

| | |
|---|---|
| x | a summary.marg object. This is assumed to be the result returned by the summary.marg function. |
| all | if TRUE all the information stored in the summary.marg object is printed, else only a subset of it. The default is FALSE. |
| Coef | if TRUE all parameter estimates are printed. The default is TRUE. |
| int | if TRUE confidence intervals are printed. The default is TRUE. |
| test | if TRUE test statistics and tail probabilities are printed. The default is FALSE. |
| digits | the number of significant digits to be printed. The default depends on the value of digits set by options. |
| ... | additional arguments. |

## Details

Changing the default values of `all`, `Coef`, `int` and `test` allows only a subset of the information in the `summary.marg` object to be printed. With `all = FALSE`, one-sided confidence intervals and the Lugannani-Rice tail area approximation are omitted. See `summary.marg` for more details.

## Note

If the parameter of interest is the scale parameter, all calculations are performed on the log scale, though most results are reported on the original scale.

The amount of information printed may vary depending on whether there are any nuisance parameters. A message is printed if there are none.

## See Also

`summary.marg`, `marg.object`

## Examples

```
## House Price Data
data(houses)
houses.rsm <- rsm(price ~ ., family = student(5), data = houses)
houses.cond <- cond(houses.rsm, front)
print(summary(houses.cond), digits = 4)
print(summary(houses.cond), Coef = FALSE)
```

---

# 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 all.profiles.nlreg. |
| vd | a function definition that returns the first two derivatives of the variance function; used by all.profiles.nlreg. |
| all | logical switch used by all.profiles.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 $\pm$ $\delta$, until the absolute value of all statistics exceeds the threshold 2.4. The step size $\delta$ 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)
```

---

`rabbits` *Rabbits Data*

---

## Description

The `rabbits` data frame has 10 rows and 4 columns.

Five different doses of penicillin were administered to rabbits suffering from a streptococci infection and the number of recovering rabbits recorded. The rabbits are cross-classified according to whether the drug is administered immediately or delayed by an hour and a half. Interest focuses on whether the delay effects the treatment.

## Usage

```
data(rabbits)
```

## Format

This data frame contains the following columns:

`cured` the number of rabbits that recovered;

`died` the number of rabbits that died;

`delay` an indicator variable indicating whether the administration of penicillin was delayed by 1 1/2 hours;

`penicil` the penicillin dose.

## Source

Unknown.

## Examples

```
data(rabbits)
attach(rabbits)
fc <- cured/(cured + died)
coplot(fc ~ log(penicil) | delay, data = rabbits)
```

`residuals.rsm` *Compute Residuals for Regression-Scale Models*

## Description

Computes one of the six types of residuals available for regression-scale models.

## Usage

```
## S3 method for class 'rsm'
residuals(object, type = c("deviance", "pearson",
                           "response", "r.star", "prob", "deletion"),
          weighting = "observed", ...)
```

## Arguments

| | |
|---|---|
| `object` | an object inheriting from class `rsm` representing a fitted regression-scale model. |
| `type` | character string; defines the type of residuals, with choices `"deviance"`, `"pearson"`, `"response"`, `"r.star"`, `"prob"` or `"deletion"`; the first is the default. |
| `weighting` | character string; defines the weight matrix that should be used in the calculation of the residuals and diagnostics. Possible choices are `"observed"`, `"score"`, `"deviance"` and `"max"`; see *Jorgensen (1984)* for their definition. The default is `"observed"`. |
| `...` | absorbs any additional argument. |

## Details

This is a method for the function `residuals()` for objects inheriting from class `rsm`. As several types of residuals are available for `rsm` objects, there is an additional optional argument `type`. The `"deviance"`, `"pearson"`, `"r.star"`, `"prob"` and `"deletion"` residuals are derived from the final IRLS fit. The `"response"` residuals are standardized residuals on the scale of the response, the `"prob"` residuals are on the $U(0,1)$ scale, whereas the remaining ones follow approximately the standard normal distribution.

The default weighting scheme used is `"observed"`. The weights used are the values stored in the `q2` component of the `rsm` object. Some of the IRLS weights returned by `rsm` may be negative if the error distribution is Student's t or user-defined. In order to avoid missing values in the residuals, the default weighting scheme used is then `"score"` unless otherwise specified. The `"score"` weights are also used by default if Huber's least favourable error distribution is used.

More details, in particular of the use of these residuals, are given in *Brazzale (2000, Section 6.3.1).*

## Value

A numeric vector of residuals. See *Davison and Snell (1991)* for detailed definitions of each type of residual.

## Note

The `summary` method for `rsm` objects produces response residuals. The `residuals` component of a `rsm` object contains the response residuals.

## References

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

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.

Jorgensen, B. (1984). The delta algorithm and GLIM. *Int. Stat. Rev.*, **52**, 283–300.

## See Also

`rsm.object`, `residuals`

## Examples

```
## Sea Level Data
data(venice)
attach(venice)
Year <- 1:51/51
c11 <- cos(2*pi*1:51/11) ; s11 <- sin(2*pi*1:51/11)
c19 <- cos(2*pi*1:51/18.62) ; s19 <- sin(2*pi*1:51/18.62)
venice.rsm <- rsm(sea ~ Year + I(Year^2) + c11 + s11 + c19 + s19,
                  family = extreme)
##
residuals(venice.rsm)
## deviance residuals with observed weights
residuals(venice.rsm, type = "r.star", weighting = "score")
## r* residuals with score weights
detach()
```

---

# 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()
```

---

# rsm *Fit a Regression-Scale Model*

---

## Description

Produces an object of class `rsm` which is a regression-scale model fit of the data.

## Usage

```
rsm(formula = formula(data), family = gaussian,
    data = sys.frame(sys.parent()), dispersion = NULL,
    weights = NULL, subset = NULL, na.action = na.fail,
    offset = NULL, method = "rsm.surv",
    control = glm.control(maxit=100, trace=FALSE),
    model = FALSE, x = FALSE, y = TRUE, contrasts = NULL, ...)
```

## Arguments

| | |
|---|---|
| formula | a formula expression as for other linear regression models, of the form `response ~ predictors` where the predictors are separated by suitable operators. See the documentation of `lm` and `formula` for details. |
| family | a `family.rsm` object, i.e. a list of functions and expressions characterizing the error distribution. Families supported are `gaussian`, `student` (Student's t), `extreme` (Gumbel or extreme value), `logistic`, `logWeibull`, `logExponential`, `logRayleigh` and `Huber` (Huber's least favourable). These represent calls to the corresponding generator functions. The calls to `gaussian`, `extreme`, `logistic`, `logWeibull`, `logExponential` and `logRayleigh` can be given without parentheses. The functions `student` and `Huber` may take as argument respectively the degrees of freedom (`df`) and the tuning constant (`k`). Users can construct their own families, as long as they have components compatible with those given in `rsm.distributions`. The demonstration file 'margdemo.R' that ships with the package shows how to create a new generator function. The default is `gaussian`. |
| data | an optional data frame in which to interpret the variables occurring in the model formula, or in the `subset` and the `weights` arguments. If this is missing, then the variables in the formula should be on the search list. |
| dispersion | if `NULL`, the scale parameter is taken to be unknown. If known, the numerical value can be passed. The default is `NULL`. Huber's least favourable distribution represents a special case. If `dispersion` is `NULL`, the maximum likelihood estimate is computed, while if `TRUE` the MAD estimate is calculated and the scale parameter fixed to this value in subsequent computations. |
| weights | the optional weights for the fitting criterion. If supplied, the response variable and the covariates are multiplied by the weights in the IRLS algorithm. The length of the `weights` argument must be the same as the number of observations. The weights must be nonnegative and it is strongly recommended that they be strictly positive, since zero weights are ambiguous, compared to use of the `subset` argument. |
| subset | expression saying which subset of the rows of the data should be used in the fit. This can be a logical vector (which is replicated to have length equal to the number of observations), or a numeric vector indicating which observation numbers are to be included, or a character vector of the row names to be included. All observations are included by default. |
| na.action | a function to filter missing data. This is applied to the model frame after any `subset` argument has been used. The default (with `na.fail`) is to create an error if any missing value is found. A possible alternative is `na.omit`, which deletes observations that contain one or more missing values. |
| offset | this can be used to specify an *a priori* known component to be included in the linear predictor during fitting. An `offset` term can be included in the formula instead or as well, and if both are specified their sum is used. Defaults to `NULL` |
| method | the fitting method to be used; the default is `rsm.fit`. The method `model.frame` simply returns the model frame. |
| control | a list of iteration and algorithmic constants. See `glm.control` for their names and default values. |

| | |
|---|---|
| model | if TRUE, the model frame is returned; default is FALSE. |
| x | if TRUE, the model matrix is returned; default is FALSE. |
| y | if TRUE, the response variable is returned; default is TRUE. |
| contrasts | a list of contrasts to be used for some or all of the factors appearing as variables in the model formula. The names of the list should be the names of the corresponding variables, and the elements should either be contrast-type matrices (matrices with as many rows as levels of the factor and with columns linearly independent of each other and of a column of one's), or else they should be functions that compute such contrast matrices. |
| ... | absorbs any additional argument. |

## Details

The model is fitted using *Iteratively Reweighted Least Squares*, IRLS for short (*Green, 1984*, *Jorgensen, 1984*). The working response and iterative weights are computed using the functions contained in the family.rsm object.

The two workhorses of rsm are rsm.fit and rsm.surv, which expect an X and Y argument rather then a formula. The first function is used for the families student with df $< 3$ and Huber; the second one, based on the survreg.fit routine for fitting parametric survival models, is used in case of extreme, logistic, logWeibull, logExponential, logRayleigh and student (with df $> 2$) error distributions. In the presence of a user-defined error distribution the rsm.fit routine is used. The rsm.null function is invoked to fit an empty (null) model.

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

## Value

an object of class rsm is returned which inherits from glm and lm. See rsm.object for details.

The output can be examined by print, summary, rsm.diag.plots and anova. Components can be extracted using fitted, residuals, formula and family. It can be modified using update. It has most of the components of a glm object, with a few more. Use rsm.object for further details.

## Note

In case of extreme, logistic, logWeibull, logExponential, logRayleigh and student (with df $> 2$) error distributions, both methods, rsm.fit (default choice) and rsm.surv, can be used to fit the model. There are, however, examples where one of the two algorithms (most likely the one invoked by rsm.surv) breaks down. If this is the case, try and refit the model with the alternative choice.

The message "negative iterative weights returned!" is returned if some of the iterative weights (q2 component of the fitted rsm object) are negative. These would be used by default by the rsm.diag routine for the definition of residuals and regression diagnostics. In order to avoid missing values (NAs), the default weighting scheme "observed" automatically switches to "score" unless otherwise specified.

## References

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

Green, P. J. (1984) Iteratively reweighted least squares for maximum likelihood estimation, and some robust and resistant alternatives (with Discussion). *J. R. Statist. Soc.* B, **46**, 149–192.

Jorgensen, B. (1984) The delta algorithm and GLIM. *Int. Stat. Rev.*, **52**, 283–300.

## See Also

rsm.object, rsm.fit, rsm.surv, rsm.null, rsm.families

## Examples

```
## House Price Data
data(houses)
houses.rsm <- rsm(price ~ ., family = student(5), data = houses)
## model fit including all covariates
houses.rsm <- rsm(price ~ ., family = student(5), data = houses,
                  method = "rsm.fit", control = glm.control(trace = TRUE))
## prints information about the iterative procedure at each iteration
update(houses.rsm, ~ . - bdroom + offset(7 * bdroom))
## "bdroom" is included as offset variable with fixed (= 7) coefficient

## Sea Level Data
data(venice)
attach(venice)
Year <- 1:51/51
venice.2.rsm <- rsm(sea ~ Year + I(Year^2), family = extreme)
## quadratic model fitted to sea level data
venice.1.rsm <- update(venice.2.rsm, ~. - I(Year^2))
## linear model fit
##
c11 <- cos(2*pi*1:51/11) ; s11 <- sin(2*pi*1:51/11)
c19 <- cos(2*pi*1:51/18.62) ; s19 <- sin(2*pi*1:51/18.62)
venice.rsm <- rsm(sea ~ Year + I(Year^2) + c11 + s11 + c19 + s19,
                  family = extreme)
## includes 18.62-year astronomical tidal cycle and 11-year sunspot cycle
venice.11.rsm <- rsm(sea ~ Year + I(Year^2) + c11 + s11, family = extreme)
venice.19.rsm <- rsm(sea ~ Year + I(Year^2) + c19 + s19, family = extreme)
## includes either astronomical cycle
##
## comparison of linear, quadratic and periodic (11-year, 19-year) models
plot(year, sea, ylab = "sea level")
lines(year, fitted(venice.1.rsm))
lines(year, fitted(venice.2.rsm), col="red")
lines(year, fitted(venice.11.rsm), col="blue")
lines(year, fitted(venice.19.rsm), col="green")
##
detach()

## Darwin's Data on Growth Rates of Plants
```

```
data(darwin)
darwin.rsm <- rsm(cross - self ~ pot - 1, family  =  student(3),
                  data = darwin)
## Maximum likelihood estimates
darwin.rsm <- rsm(cross - self ~ pot - 1, family = Huber, data = darwin)
## M-estimates
```

---

`rsm.diag` *Diagnostics for Regression-Scale Models*

---

## Description

Calculates different types of residuals, Cook's distance and the leverages for a regression-scale model.

## Usage

```
rsm.diag(rsmfit, weighting = "observed")
```

## Arguments

| | |
|---|---|
| `rsmfit` | an `rsm` object, i.e. the result of a call to `rsm`. |
| `weighting` | character string; defines the weight matrix that should be used in the calculation of the residuals and diagnostics. Possible choices are `"observed"`, `"score"`, `"deviance"` and `"max"`; see *Jorgensen (1984)* for their definition. The default is `"observed"`. |

## Details

If the weighting scheme is `"observed"`, the weights used are the values stored in the `q2` component of the `rsm` object `rsmfit`. Otherwise, they are calculated by `rsm.diag`. Some of the IRLS weights returned by `rsm` may be negative if the error distribution is Student's t or user-defined. In order to avoid missing values in the residuals and regression diagnostics, the default weighting scheme used in `rsm.diag` switches automatically from `"observed"` to `"score"` unless otherwise specified. The `"score"` weights are also used by default if Huber's least favourable error distribution is used.

There are three types of residuals. The response residuals are taken on the response scale, whereas the probability transform residuals are on the $U(0,1)$ scale. The remaining ones follow approximately the standard normal distribution.

More details and in particular the definitions of the above residuals and diagnostics can be found in *Brazzale (2000, Section 6.3.1)*.

## Value

Returns a list with the following components:

| | |
|---|---|
| `resid` | the response residuals on the response scale. |
| `rd` | the standardized deviance residuals from the IRLS fit. |

| | |
|---|---|
| rp | the standardized Pearson residuals from the IRLS fit. |
| rg | the deletion residuals from the IRLS fit. |
| rs | the $r^*$ residuals from the IRLS fit. |
| rcs | the probability transform residuals from the IRLS fit. |
| cook | Cook's distance. |
| h | the leverages of the observations. |
| dispersion | the value of the scale parameter. |

## Acknowledgments

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

## Note

Huber's least favourable distribution represents a special case. The regression diagnostics are only meaningful if the errors *truly* follow a Huber-type distribution. This no longer holds if the option `family = Huber` in `rsm` is used to obtain the M-estimates of the parameters in place or the maximum likelihood estimates.

## References

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

Jorgensen, B. (1984) The delta algorithm and GLIM. *Int. Stat. Rev.*, **52**, 283–300.

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

`rsm.diag.plots`, `rsm.object`, `summary.rsm`

## Examples

```
## Sea Level Data
data(venice)
attach(venice)
Year <- 1:51/51
c11 <- cos(2*pi*1:51/11) ; s11 <- sin(2*pi*1:51/11)
c19 <- cos(2*pi*1:51/18.62) ; s19 <- sin(2*pi*1:51/18.62)
venice.rsm <- rsm(sea ~ Year + I(Year^2) + c11 + s11 + c19 + s19,
                  family = extreme)
venice.diag <- rsm.diag(venice.rsm)
## observed weights
detach()

## Darwin's Data on Growth Rates of Plants
```

```
data(darwin)
darwin.rsm <- rsm(cross-self ~ pot - 1, family = Huber, data = darwin)
darwin.diag <- rsm.diag(darwin.rsm)
## score weights
```

---

`rsm.diag.plots` *Diagnostic Plots for Regression-Scale Models*

---

## Description

Generates diagnostic plots for a regression-scale model using different types of residuals, Cook's distance and the leverages.

## Usage

```
rsm.diag.plots(rsmfit, rsmdiag = NULL, weighting = NULL,
               which = NULL, subset = NULL, iden = FALSE,
               labels = NULL, ret = FALSE, ...)
## S3 method for class 'rsm'
plot(x, ...)
```

## Arguments

| | |
|---|---|
| `rsmfit, x` | a `rsm` object, i.e. the result of a call to `rsm`. |
| `rsmdiag` | the object returned by a call to `rsm.diag` containing the regression diagnostics for the regression-scale model defined by `rsmfit`. If not supplied, this object is created by `rsm.diag.plots` and returned upon request (if `ret = TRUE`). |
| `weighting` | character string; defines the weight matrix that should be used in the calculation of the residuals and diagnostics. Possible choices are `"observed"`, `"score"`, `"deviance"` and `"max"`; see *Jorgensen (1984)* for their definition. Will only be used if the `rsmdiag` argument is missing. |
| `which` | which plot to print. Admissible values are `2` to `7` corresponding to the choices in the menu below. |
| `subset` | subset of data used in the original `rsm` fit: should be the same than the `subset` option used in the call to `rsm` which generated `rsmfit`. Needed only if the `subset` option was used in the call to `rsm`. |
| `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, then the labels are derived from `rsmfit`. |
| `ret` | logical argument indicating if `rsmdiag` should be returned; the default is `FALSE`. |
| `...` | additional arguments such as graphical parameters. |

## Details

The diagnostics required for the plots are calculated by `rsm.diag`. These are then used to produce the plots on the current graphics device.

A menu lists all the plots that can be produced. They may be one or all of the following:

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

1: All
2: Response residuals against fitted values
3: Deviance residuals against fitted values
4: QQ-plot of deviance residuals
5: Normal QQ-plot of r* residuals
6: Cook statistic against h/(1-h)
7: Cook statistic against observation number

Selection:
```

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

In general, when plotting Cook's distance 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 estimated parameters. 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` is `TRUE` then the value of `rsmdiag` 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 the package **boot**.

## References

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, London.

Davison, A. C. and Tsai, C.-L. (1992) Regression model diagnostics. *Int. Stat. Rev.*, **60**, 337–353.
Jorgensen, B. (1984) The Delta Algorithm and GLIM. *Int. Stat. Rev.*, **52**, 283–300.

## See Also

rsm.diag, rsm.object, identify

## Examples

```
## Sea Level Data
data(venice)
attach(venice)
Year <- 1:51/51
c11 <- cos(2*pi*1:51/11) ; s11 <- sin(2*pi*1:51/11)
c19 <- cos(2*pi*1:51/18.62) ; s19 <- sin(2*pi*1:51/18.62)
venice.rsm <- rsm(sea ~ Year + I(Year^2) + c11 + s11 + c19 + s19,
                  family = extreme)
## Not run:
rsm.diag.plots(venice.rsm, which = 3)

## End(Not run)
## or
## Not run:
plot(venice.rsm)

## End(Not run)
## menu-driven
##
rsm.diag.plots(venice.rsm, which = 5, las = 1)
## normal QQ-plot of r* residuals
## Not run:
rsm.diag.plots(venice.rsm, which = 7, iden = T, labels = paste(1931:1981))

## End(Not run)
## year 1932 highly influential
detach()
```

---

# rsm.families *Generate a RSM Family Object*

---

## Description

Generates a family.rsm object containing a list of functions and expressions used by rsm.

## Usage

```
extreme()
Huber(k = 1.345)
logistic()
logWeibull()
student(df = stop("Argument \"df\" is missing, with no default"))
```

### Arguments

| | |
|---|---|
| k | the tuning constant in Huber's least favourable distribution. |
| df | the degrees of freedom in Student's t distribution. |

### Details

Each of the names are associated with a member of the class of error distributions for regression-scale models. Users can construct their own families, as long as they have components compatible with those given in `rsm.distributions`. The demonstration file 'margdemo.R' that accompanies the package shows how to create a new generator function. When passed as an argument to `rsm` with the default setting, the empty parentheses `()` can be omitted. There is a `print` method for the class `family.rsm`.

### Value

A `family.rsm` object, which is a list of functions and expressions used by `rsm` in the iteratively reweighted least-squares algorithm. See `family.rsm.object` for details.

### See Also

`family.rsm.object`, `family.rsm`, `rsm`, `Huber`

### Examples

```
student(df = 3) ## generates Student's t error distribution with 3 d.f.
## Not run:
rsm(formula = value, data = value, family = extreme)

## End(Not run)
```

---

## rsm.fit *Fit a Regression-Scale Model Without Computing the Model Matrix*

---

### Description

Fits a `rsm` model without computing the model matrix of the response vector.

### Usage

```
rsm.fit(X, Y, offset, family, dispersion, score.dispersion, maxit, epsilon,
        trace, ...)
```

## Arguments

| | |
|---|---|
| X | the model matrix (design matrix). |
| Y | the response vector. |
| dispersion | if NULL, the MLE of the scale parameter is returned, otherwise the scale parameter is fixed to the numerical value passed through the argument. If Huber's least favourable distribution is used and dispersion is TRUE, the MAD is computed and the scale parameter fixed to this value in subsequent calculations. |
| score.dispersion | must default to NULL. |
| offset | optional offset added to the linear predictor. |
| family | a family.rsm object, i.e. a list of functions and expressions characterizing the error distribution. Families supported are gaussian, student (Student's t), extreme (Gumbel or extreme value), logistic, logWeibull, logExponential, logRayleigh and Huber (Huber's least favourable). Users can construct their own families, as long as they have components compatible with those given in rsm.distributions. The demonstration file 'margdemo.R' that ships with the package shows how to create a new generator function. |
| maxit | maximum number of iterations allowed. |
| epsilon | convergence threshold. |
| trace | if TRUE, iterations details are printed during execution. |
| ... | not used, but do absorb any redundant argument. |

## Details

The rsm.fit function is called internally by the rsm routine to do the actual model fitting. Although it is not intended to be used directly by the user, it may be useful when the same data frame is used over and over again. It might save computational time, since the model matrix is not created. No formula needs to be specified as an argument. As no weights argument is available, the response Y and the model matrix X must already include the weights if weighting is desired.

## Value

an object which is a subset of a rsm object.

## Note

The rsm.fit function is the workhorse of the rsm fitting routine for the student (with df $\leq 2$), Huber and user-defined error distributions. It receives X and Y data rather than a formula, but still uses the family.rsm object to define the IRLS steps. Users can write their own versions of rsm.fit, and pass the name of their function via the method argument to rsm. Care should be taken to include as many of the arguments as feasible, but definitely the ... argument, which will absorb any additional argument given in the call from rsm.

## See Also

rsm, rsm.surv, rsm.null, rsm.object, rsm.families

rsm.null *Fit an Empty Regression-Scale Model*

## Description

Fits a rsm model with empty model matrix.

## Usage

```
rsm.null(X = NULL, Y, offset, family, dispersion, score.dispersion, maxit,
         epsilon, trace, ...)
```

## Arguments

| | |
|---|---|
| X | defaults to NULL. |
| Y | the response vector. |
| dispersion | either NULL or TRUE. If NULL, the MLE of the scale parameter is returned. If Huber's least favourable distribution is used and dispersion is TRUE, the MAD is computed and the scale parameter fixed to this value in subsequent calculations. |
| score.dispersion | must default to NULL. |
| offset | optional offset added to the linear predictor. |
| family | a family.rsm object, i.e. a list of functions and expressions characterizing the error distribution. Families supported are gaussian, student (Student's t), extreme (Gumbel or extreme value), logistic, logWeibull, logExponential, logRayleigh and Huber (Huber's least favourable). Users can construct their own families, as long as they have components compatible with those given in rsm.distributions. The demonstration file 'margdemo.R' that ships with the package shows how to create a new generator function. |
| maxit | maximum number of iterations allowed. |
| epsilon | convergence threshold. |
| trace | if TRUE, iterations details are printed during execution. |
| ... | not used, but do absorb any redundant argument. |

## Details

The rsm.null function is called internally by the rsm routine to do the actual model fitting in case of an empty model. It is not intended to be used directly by the user. As no weights argument is available, the response Y and the model matrix X must already include the weights if weighting is desired.

## Value

an object which is a subset of a rsm object.

## See Also

rsm, rsm.surv, rsm.fit, rsm.object, rsm.families

---

rsm.object *Regression-Scale Model Object*

---

## Description

Class of objects returned when fitting a regression-scale model.

## Arguments

| | |
|---|---|
| | The following components must be included in a rsm object: |
| | the coefficients of the linear predictor, which multiply the columns of the model matrix. The names of the coefficients are the names of the single-degree-of-freedom effects (the columns of the model matrix). If the model is over-determined there will be missing values in the coefficients corresponding to inestimable coefficients. |
| dispersion | the (estimated or known) value of the scale parameter. |
| fixed | a logical value. If TRUE, the scale parameter is fixed. |
| residuals | the response residuals from the fit. If weights were used, they are not taken into account. If you need other kinds of residuals, use the residuals.rsm function. |
| fitted.values | the fitted values from the fit. If weights were used, the fitted values are not adjusted for the weights. |
| loglik | the log likelihood from the fit. |
| q1 | the value of the first derivative of minus the log density for each observation. |
| q2 | the value of the second derivative of minus the log density for each observation. |
| rank | the computed rank (number of linearly independent columns in the model matrix). |
| R | the unscaled observed information matrix. |
| score.dispersion | a list containing the value of the objective function, its gradient and the convergence diagnostic, that result from estimating the scale parameter. |
| iter | the number of IRLS iterations used to compute the estimates. |
| weights | the (optional) weights used for the fit. |
| assign | the list of assignments of coefficients (and effects) to the terms in the model. The names of this list are the names of the terms. The *i*th element of the list is the vector saying which coefficients correspond to the *i*th term. It may be of length 0 if there were no estimable effects for the term. |
| df.residuals | the number of degrees of freedom for residuals. |
| family | the entire family.rsm object used. |

| | |
|---|---|
| user.def | a logical value. If TRUE, the error distribution is user-defined. |
| dist | a character string representing the name of the error distribution. |
| formula | the model formula. |
| data | the data frame in which to interpret the variables occurring in the model formula, or in the subset and the weights arguments to rsm. |
| terms | an object of mode expression and class term summarizing the formula. |
| contrasts | a list containing sufficient information to construct the contrasts used to fit any factors occurring in the model. The list contains entries that are either matrices or character vectors. When a factor is coded by contrasts, the corresponding contrast matrix is stored in this list. Factors that appear only as dummy variables and variables in the model that are matrices correspond to character vectors in the list. The character vector has the level names for a factor or the column labels for a matrix. |
| control | a list of iteration and algorithmic constants used in rsm to fit the model. |
| call | an image of the call that produced the object, but with the arguments all named and with the actual formula included as the formula argument. |
| y | optionally the response, if y = TRUE in the original rsm call. |
| x | optionally the model matrix, if x = TRUE in the original rsm call. |
| model | optionally the model frame, if model = TRUE in the original rsm call. |

## Generation

This class of objects is returned by the rsm function to represent a fitted regression-scale model. Class rsm inherits from classes glm and lm, since it is fitted by iteratively reweighted least squares. The object returned has all the components of a weighted least squares object.

## Methods

Objects of this class have methods for the functions print, summary, anova and 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.

## See Also

rsm, glm, lm.

rsm.surv *Fit a Regression-Scale Model Without Computing the Model Matrix*

## Description

Fits a rsm model without computing the model matrix of the response vector.

## Usage

```
rsm.surv(X, Y, offset, family, dispersion, score.dispersion, maxit, epsilon,
         trace, ...)
```

## Arguments

| | |
|---|---|
| X | the model matrix (design matrix). |
| Y | the response vector. |
| offset | optional offset added to the linear predictor. |
| family | a family.rsm object, i.e. a list of functions and expressions characterizing the error distribution. Families supported are extreme (Gumbel or extreme value), logWeibull, logExponential, logRayleigh, logistic and student (Student's t) with df $> 2$. |
| dispersion | if NULL, the MLE of the scale parameter is returned, otherwise the scale parameter is fixed to the numerical value passed through the argument. |
| score.dispersion | must default to NULL. |
| maxit | maximum number of iterations. |
| epsilon | convergence threshold. |
| trace | if TRUE, iterations details are printed during execution. |
| ... | not used, but do absorb any redundant argument. |

## Details

The rsm.surv function is called internally by the rsm routine to do the actual model fitting. Although it is not intended to be used directly by the user, it may be useful when the same data frame is used over and over again. It might save computational time, since the model matrix is not created. No formula needs to be specified as an argument. As no weights argument is available, the response Y and the model matrix X must already include the weights if weighting is desired.

## Value

an object, which is a subset of a rsm object.

## Note

The rsm.surv function is the default option for rsm for the extreme, logistic, logWeibull, logExponential, logRayleigh and student (with df larger than 2) error distributions. It makes use of the survreg.fit routine to estimate parametric survival models. It receives X and Y data rather than a formula, but still uses the family.rsm object to define the IRLS steps. The rsm.surv routine cannot be used for Huber-type and user-defined error distributions.

## See Also

rsm, rsm.fit, rsm.null, rsm.object, rsm.families

---

# 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.cond` *Summary Method for Objects of Class "cond"*

---

## Description

Returns a summary list for objects of class cond.

## Usage

```
## S3 method for class 'cond'
summary(object, alpha = 0.05, test = NULL, all = FALSE, coef = TRUE,
        int = ifelse( (is.null(test) || all), TRUE, FALSE),
        digits = NULL, ...)
```

## Arguments

| | |
|---|---|
| object | a cond object. This is assumed to be the result returned by the cond.glm function. |
| alpha | vector of levels for confidence intervals. The default is 5%. |
| test | vector of values of the parameter of interest one wants to test for. If NULL, no test is performed. The default is NULL. |
| all | logical value; if TRUE, all the information stored in the summary.cond object is printed, else only a subset of it. The default is FALSE. |
| coef | logical value; if TRUE, the unconditional and conditional parameter estimates are printed. The default is TRUE. |
| int | logical value; if TRUE confidence intervals are printed. The default is TRUE. |
| digits | number of significant digits to be printed. The default depends on the value of digits set by options. |
| ... | absorbs any additional argument. |

## Details

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

## Value

A list is returned with the following components.

| | |
|---|---|
| coefficients | a $2 \times 2$ matrix containing the unconditional and approximate conditional MLEs and their standard errors. |
| conf.int | a matrix containing, for each level given in alpha, the upper and lower confidence bounds derived from several first- and higher order test statistics. One-sided and two-sided confidence intervals are considered. See cond.object for details on the test statistics. |
| signif.tests | a list with two elements. The first (stats) contains, for each value given in test, the values and tail probabilities of several first- and higher order test statistics. See cond.object for details on the test statistics.The second element of the list (qTerm) contains for each tested hypothesis the correction term used in the higher order solutions. |
| call | the function call that created the cond object. |
| formula | the model formula. |
| family | the variance function. |
| offset | the covariate occurring in the model formula whose coefficient represents the parameter of interest. |
| alpha | vector of levels used to compute the confidence intervals. |
| hypotheses | values for the parameter of interest that have been tested for. |
| diagnostics | information and nuisance parameters aspects; see cond.object for details. |

| | |
|---|---|
| n.approx | number of output points that have been calculated exactly. |
| all | logical value; if TRUE, all the information stored in the summary.cond object is printed. |
| cf | logical value; if TRUE, the unconditional and conditional parameter estimates are printed. |
| int | logical value; if TRUE, confidence intervals are printed. |
| is.scalar | a logical value indicating whether there are any nuisance parameters. If FALSE there are none. |
| digits | number of significant digits to be printed. |

## Note

The amount of information calculated may vary depending on whether there are any nuisance parameters.

## See Also

summary, cond.object

## Examples

```
## Crying Babies Data
data(babies)
babies.glm <- glm(formula = cbind(r1, r2) ~ day + lull - 1,
                  family = binomial, data = babies)
babies.cond <- cond(object = babies.glm, offset = lullyes)
summary(babies.cond, test = 0, coef = FALSE)
```

---

# summary.fr *Likelihood-Based Confidence Intervals Based on fr Object*

---

## Description

Prints confidence limits based on the Wald statistic, the likelihood root, and the modified likelihood root, for a default two-sided 0.95 confidence interval. It also prints the point estimate based on these 3 statitics. Summary.fr is a compatibility wrapper for lik.ci, to which it is identical.

## Usage

```
## S3 method for class 'fr'
summary(object, conf = c(0.975, 0.025), ...)
lik.ci(object, conf = c(0.975, 0.025), ...)
```

## Arguments

| | |
|---|---|
| object | A fr object |
| conf | Confidence levels for which limits are required |
| ... | Absorbs any additional parameter. |

## Value

List containing the following quantities:

| | |
|---|---|
| `mle` | MLE and its asymptotic standard error |
| `mle.hoa` | Modified MLE and its asymptotic standard error |
| `pointEst.z` | Point estimate for psi based on Wald pivot `z` |
| `pointEst.r` | Point estimate for psi based on likelihood root `r` |
| `pointEst.rstar` | Point estimate for psi based on modified likelihood root `rstar` |
| `z.lims` | Confidence limits based on Wald pivot `z` |
| `r.lims` | Confidence limits based on likelihood root `r` |
| `rstar.lims` | Confidence limits based on modified likelihood root `rstar` |

## Author(s)

Anthony Davison <Anthony.Davison@epfl.ch>

## References

Brazzale, A. R., Davison, A. C. and Reid, N. (2007). *Applied Asymptotics: Case Studies in Small-Sample Statistics*. Cambridge University Press, Cambridge.

See also http://statwww.epfl.ch/AA.

## See Also

`tem`

## Examples

```
## See the examples to \link{tem}.
```

---

`summary.marg` *Summary Method for Objects of Class "marg"*

---

## Description

Returns a summary list for objects of class `marg`.

## Usage

```
## S3 method for class 'marg'
summary(object, alpha = 0.05, test = NULL, all = FALSE,
        coef = TRUE, int = ifelse((is.null(test) || all), TRUE, FALSE),
        digits = NULL, ...)
```

## Arguments

| | |
|---|---|
| object | a marg object. This is assumed to be the result returned by the cond.rsm function. |
| alpha | a vector of levels for confidence intervals; the default is 5%. |
| test | a vector of values of the parameter of interest one wants to test for. If NULL no test is performed. The default is NULL. |
| all | logical value; if TRUE all the information stored in the summary.marg object is printed, else only a subset of it. The default is FALSE. |
| coef | logical value; if TRUE the unconditional and approximate conditional/marginal MLEs are printed. The default is TRUE. |
| int | logical value; if TRUE confidence intervals are printed. The default is TRUE. |
| digits | the number of significant digits to be printed. The default depends on the value of digits set by options |
| ... | absorbs any additional argument. |

## Details

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

## Value

A list is returned with the following components:

| | |
|---|---|
| coefficients | a $2\times2$ matrix containing the unconditional and approximate conditional/marginal MLEs and their standard errors. |
| conf.int | a matrix containing, for each level given in alpha, the upper and lower confidence bounds derived from several first and higher order test statistics. One-sided and two-sided confidence intervals are considered. See marg.object for details on the test statistics used. |
| signif.tests | a list with two elements. The first (stats) contains, for each value given in test, the values and tail probabilities of several first and higher order test statistics. See marg.object for details on the test statistics. The second element of the list (qTerm) contains for each tested hypothesis the correction term used in the higher order solutions. |
| call | the function call that created the marg object. |
| formula | the model formula. |
| family | the name of the error distribution. |
| offset | the covariate occurring in the model formula whose coefficient represents the parameter of interest or scale if the parameter of interest is the scale parameter. |
| alpha | the vector of levels used to compute the confidence intervals. |
| hypotheses | the values for the parameter of interest that have been tested for. |
| diagnostics | the information and nuisance parameters aspects; see marg.object for details. |

| | |
|---|---|
| n.approx | the number of output points that have been calculated exactly. |
| all | logical value; if TRUE, all the information stored in the summary.marg object is printed. |
| cf | logical value; if TRUE, the parameter estimates are printed. |
| int | logical value; if TRUE, confidence intervals are printed. |
| is.scalar | a logical value indicating whether there are any nuisance parameters. If FALSE there are none. |
| digits | the number of significant digits to be printed. |

## Note

If the parameter of interest is the scale parameter, all calculations are performed on the log scale, though most results are reported on the original scale.

The amount of information calculated may vary depending on whether there are any nuisance parameters. A message is printed if there are none.

## See Also

summary, marg.object

## Examples

```
## House Price Data
data(houses)
houses.rsm <- rsm(price ~ ., family = student(5), data = houses)
houses.marg <- cond(houses.rsm, floor)
summary(houses.marg, test = 0, coef = FALSE)
```

---

# 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 $k\,\dim(\texttt{alpha})$ 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) )
```

---

# summary.rsm *Summary Method for Regression-Scale Models*

---

## Description

Returns a summary list for a fitted regression-scale model.

## Usage

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

## Arguments

| | |
|---|---|
| object | a fitted rsm object. This is assumed to be the result of some fit that produces an object inheriting from the class rsm, in the sense that the components returned by the rsm function will be available. |
| correlation | logical argument. If TRUE, the correlation matrix for the coefficients is computed; default is TRUE. |
| digits | a non-null value specifies the minimum number of significant digits to be printed in values. If NULL, the value of digits set by options is used. |
| ... | absorbs any additional argument. |

## Details

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

## Value

A list is returned with the following components:

| | |
|---|---|
| `coefficients` | a matrix with four columns, containing the coefficients, their standard errors, the $z$-values (or Wald statistics) and the associated $p$-values based on the standard normal approximation to the distribution of the $z$ statistic. |
| `dispersion` | the value of the scale parameter used in the computations. |
| `fixed` | a logical value. If `TRUE`, the scale parameter is known. |
| `residuals` | the response residuals. |
| `cov.unscaled` | the unscaled covariance matrix, i.e, a matrix such that multiplying it by the squared scale parameter, or an estimate thereof, produces an estimated (asymptotic) covariance matrix for the coefficients. |
| `correlation` | the computed correlation matrix for the coefficients in the model. |
| `family` | the entire `family.rsm` object used. |
| `loglik` | the computed log likelihood. |
| `terms` | an object of mode `expression` and class `term` summarizing the formula. |
| `df` | the number of degrees of freedom for the model and for the residuals. |
| `iter` | the number of IRLS iterations used to compute the estimates. |
| `nas` | a logical vector indicating which, if any, coefficients are missing. |
| `call` | an image of the call that produced the `rsm` object, but with the arguments all named and with the actual formula. |
| `digits` | the value of the `digits` argument. |

## See Also

`rsm.object`, `summary`, `rsm`

## Examples

```
## House Price Data
data(houses)
houses.rsm <- rsm(price ~ ., family = student(5), data = houses)
summary(houses.rsm)
```

tem *Tangent exponential model: Higher Order Likelihood Approximation*

## Description

This function computes the tangent exponential model approximation for higher order likelihood inference about a scalar interest parameter of a parametric model. The function creates an object of class `fr`.

## Usage

```
tem(psi = NULL, nlogL, phi, make.V, th.init, data,
            tol = 10^(-5), n.psi = 50)
```

## Arguments

| | |
|---|---|
| `psi` | A scalar value for the interest parameter. If `NULL` (the default) the vector is taken as a grid of values centred on the maximum likelihood estimate (MLE). |
| `nlogL` | A function to compute the negative log likelihood for the model of interest. It is a function of three quantities: `psi` is the scalar interest parameter, `lam` is the scalar or vector nuisance parameter, and `data` is the data. See below for an example. |
| `phi` | A function to compute the canonical parameter of a local exponential family approximation to the density of interest. It requires values of the parameter `theta` = (`psi`,`lam`), the quantities `V`, and the data `data`. The output is a vector of the same length as `theta`. |
| `make.V` | A function to compute `V`, using as input the parameter `theta` = (`psi`,`lam`) and the data `data`. The output is a matrix whose rows correspond to individual observations, and whose columns correspond to the elements of `theta`. |
| `th.init` | Initial value(s) of the parameter `theta`. |
| `data` | Data frame or other object containing the data. |
| `tol` | Tolerance used for numerical differentiation of `phi`. |
| `n.psi` | Number of values of `psi` at which the likelihood is computed, if `psi` is `NULL`. Avoid odd values, which may give numerical instabilities near the MLE. |

## Details

The function computes quantities used for higher order likelihood approximations, which are intended to provide highly accurate inferences on scalar parameters in parametric statistical models. The key aspect is maximisation of the likelihood over a grid of values of the interest parameter `psi`, and computation of likelihood modifications based on local exponential family approximation to the density. If $n$ is the sample size, then the resulting inferences should be accurate to order $n^{-3/2}$ in continuous models and to order $n^{-1}$ in discrete models, and in many cases they are very close to exact results. The approximations rely on numerical computation of observed information matrices and of derivatives, and may fail in certain cases. The confidence intervals themselves and useful

plots are produced using the functions summary and plot. For technical background and further details, see Sections 2.4 and 8.4.2 of the book cited below, which has many further references.

## Value

| | |
|---|---|
| normal | The MLE of the interest parameter, and its standard error |
| th.hat | MLEs of parameters (psi,lam) |
| th.hat.se | Standard errors of MLEs, based on observed information |
| th.rest | Restricted MLEs (psi,lam) on grid of values of psi |
| r | Values of likelihood root corresponding to psi |
| psi | Values of interest parameter psi |
| q | Values of likelihood modification |
| rstar | Values of modified likelihood root |

## Author(s)

Anthony Davison <Anthony.Davison@epfl.ch> Alex-Antoine Fortin <alex@fortin.bio>

## References

Brazzale, A. R., Davison, A. C. and Reid, N. (2007). *Applied Asymptotics: Case Studies in Small-Sample Statistics*. Cambridge University Press, Cambridge.

See also http://statwww.epfl.ch/AA.

## See Also

plot.fr, summary.fr, lik.ci

## Examples

```
# Cost data example from Section 3.5 of "Applied Asymptotics"
cost <- data.frame(
          f = factor( c(rep(1,13), rep(2,18)) ),
          y = c( 30,172,210,212,335,489,651,1263,1294,1875,2213,2998,
                 4935,121,172,201,214,228,261,278,279,351,561,622,
                 694,848,853,1086,1110,1243,2543 ) )
nlogL <- function(psi, lam, data)  {
  s1 <- exp(lam[2])
  m2 <- lam[1]
  s2 <- exp(lam[3])
  m1 <- psi + m2 + s2^2/2 - s1^2/2
  -sum( dnorm(log(data$y), mean=ifelse(data$f==1, m1, m2),
                  sd=ifelse(data$f==1, s1, s2), log=TRUE) )
}
phi <- function(th, V, data)  {
  psi <- th[1]
  lam <- th[-1]
  s1 <- exp(lam[2])
  m2 <- lam[1]
```

```
  s2 <- exp(lam[3])
  m1 <- psi + m2 + s2^2/2 - s1^2/2
  c( m1/s1^2, 1/s1^2, m2/s2^2, 1/s2^2 )
}
make.V <- function(th, data) NULL
cost.lnorm.rat <- tem(psi = NULL, nlogL = nlogL, phi = phi,
                      make.V = make.V, th.init = c(0, 5, 2, 5), data = cost)
plot(cost.lnorm.rat, psi = 0, all = TRUE)
summary(cost.lnorm.rat)
```

---

`update.rsm` *Update and Re-fit a RSM Model Call*

---

## Description

`update.rsm` is used to update a `rsm` model formulae. This typically involves adding or dropping terms, but updates can be more general.

## Usage

```
## S3 method for class 'rsm'
update(object, formula., ..., evaluate = TRUE)
```

## Arguments

| | |
|---|---|
| `object` | a model of class `rsm` to be updated. |
| `formula.` | changes to the formula – see `update.formula` for details. |
| `...` | additional arguments to the call, or arguments with changed values. Use `name = NULL` to remove the argument `name`. |
| `evaluate` | if `TRUE` evaluate the new call else return the call. |

## Value

If `evaluate = TRUE` the fitted object, otherwise the updated call.

## Note

Based upon `update.default`.

## See Also

`update`, `update.default`, `update.formula`

## Examples

```
data(houses)
houses.rsm <- rsm(price ~ ., family = student(5), data = houses)
## model fit including all covariates
##
houses.rsm <- update(houses.rsm, method = "rsm.fit",
                     control = glm.control(trace = TRUE))
## prints information about the iterative procedure at each iteration
##
update(houses.rsm, ~ . - bdroom + offset(7 * bdroom))
## "bdroom" is included as offset variable with fixed (= 7) coefficient
```

---

# urine *Urine Data*

---

## Description

The `urine` data frame has 77 rows and 7 columns.

79 urine specimens were analyzed in an effort to determine if certain physical characteristics of the urine might be related to the formation of calcium oxalate crystals.

## Usage

```
data(urine)
```

## Format

This data frame contains the following columns:

`r` indicator of the presence of calcium oxalate crystals;

`gravity` the specific gravity of the urine, i.e. the density of urine relative to water;

`ph` the pH reading of the urine;

`osmo` the osmolarity of the urine. Osmolarity is proportional to the concentration of molecules in solution (mOsm).

`conduct` The conductivity of the urine. Conductivity is proportional to the concentration of charged ions in solution (mMho milliMho).

`urea` the urea concentration in millimoles per litre;

`calc` the calcium concentration in millimoles per litre.

## Source

The data were obtained from

Andrews, D. F. and Herzberg, A. M. (1985) *Data: A Collection of Problems from Many Fields for the Student and Research Worker*, Cambridge: Cambridge University Press.

## References

Davison, A. C. and Hinkley, D. V. (1997). *Bootstrap Methods and Their Application* (Example 7.8). Cambridge: Cambridge University Press.

## Examples

```
data(urine)
summary(urine)
pairs(urine)
##
data(urine)
urine.glm <- glm(r ~ gravity + ph + osmo + conduct + urea + log(calc),
                 family = binomial, data = urine)
labels(coef(urine.glm))
urine.cond <- cond(urine.glm, log(calc))
##
## (large estimate of regression coefficient)
urine.glm <- glm(r ~ gravity + ph + osmo + conduct + urea + calc,
                 family = binomial, data = urine)
coef(urine.glm)
urine.glm <- glm(r ~ I(gravity * 100) + ph + osmo + conduct + urea + calc,
                 family = binomial, data = urine)
coef(urine.glm)
urine.cond <- cond(urine.glm, I(gravity * 100))
```

---

# 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 )
```

---

# vcov.rsm *Calculate Variance-Covariance Matrix for a Fitted RSM Model*

---

## Description

Returns the variance-covariance matrix of the parameters of a fitted rsm model object.

## Usage

```
## S3 method for class 'rsm'
vcov(object, correlation = FALSE, ...)
```

## Arguments

| | |
|---|---|
| object | a fitted model object of class rsm. |
| correlation | if TRUE the correlation matrix is returned instead of the variance-covariance matrix. |
| ... | absobs any additional argument. |

## Details

This is a method for function vcov for objects of class rsm.

## Value

A matrix of the estimated covariances between the parameter estimates of a fitted regression-scale model, or, if dispersion = TRUE the correlation matrix.

## See Also

vcov, rsm.object, rsm, summary.rsm

## Examples

```
## Sea Level Data
data(venice)
attach(venice)
Year <- 1:51/51
c11 <- cos(2*pi*1:51/11) ; s11 <- sin(2*pi*1:51/11)
c19 <- cos(2*pi*1:51/18.62) ; s19 <- sin(2*pi*1:51/18.62)
venice.rsm <- rsm(sea ~ Year + I(Year^2) + c11 + s11 + c19 + s19,
                  family = extreme)
##
vcov(venice.rsm)
vcov(venice.rsm, corr = TRUE)
##
detach()
```

---

# venice *Sea Level Data*

---

## Description

The venice data frame has 51 rows and 2 columns.

*Pirazzoli (1982)* collected the ten largest values of sea levels in Venice (with a few exceptions) for the years 1887–1981. The venice data frame contains the maxima for the years 1931–1981.

## Usage

```
data(venice)
```

## Format

This data frame contains the following columns:

year the years;

sea the sea levels (in cm).

## Source

The data were obtained from

Smith, R. L. (1986) Extreme value theory based on the $r$-largest annual events. *Journal of Hydrology* , **86**, 27–43.

## References

Pirazzoli, P. (1982) Maree estreme a Venezia (periodo 1872–1981). *Acqua Aria*, **10**, 1023–1039.

## Examples

```
data(venice)
attach(venice)
#
plot(sea ~ year, ylab = "sea level")
##
Year <- 1:51/51
venice.l <- rsm(sea ~ Year + I(Year^2), family = extreme)
lines(year, fitted(venice.l))
##
c11 <- cos(2*pi*1:51/11) ; s11 <- sin(2*pi*1:51/11)
c19 <- cos(2*pi*1:51/18.62) ; s19 <- sin(2*pi*1:51/18.62)
venice.p <- rsm(sea ~ Year + I(Year^2) + c11 + s11 + c19 + s19,
                family = extreme)
lines(year, fitted(venice.p), col = "red")
##
detach()
```

# Index


∗Topic **classes**
cond.object, 13
family.cond, 29
family.rsm, 30
family.rsm.object, 31
family.summary.cond, 32
marg.object, 43
mpl.object, 49
nlreg.object, 57
rsm.families, 92
rsm.object, 96
∗Topic **datasets**
aids, 3
airway, 4
babies, 5
C1, 6
chlorsulfuron, 7
daphnia, 20
darwin, 21
dormicum, 24
fraudulent, 32
fungal, 33
helicopter, 34
houses, 39
metsulfuron, 45
nuclear, 59
rabbits, 81
ria, 83
urine, 115
venice, 118
∗Topic **distribution**
Huber, 40
∗Topic **hplot**
plot.nlreg.contours, 69
plot.nlreg.diag, 70
plot.nlreg.profiles, 74
rsm.diag.plots, 90
∗Topic **htest**
plot.fr, 65
summary.fr, 102
tem, 112
∗Topic **methods**
cond, 8
cond.glm, 9
cond.rsm, 14
contour.all.nlreg.profiles, 17
expInfo.nlreg, 28
family.cond, 29
family.rsm, 30
family.rsm.object, 31
family.summary.cond, 32
logLik.nlreg, 41
logLik.rsm, 42
mpl.nlreg, 47
obsInfo.nlreg, 61
param, 62
plot.cond, 63
plot.marg, 67
plot.nlreg.contours, 69
plot.nlreg.diag, 70
plot.nlreg.profiles, 74
print.summary.cond, 76
print.summary.marg, 77
profile.nlreg, 78
residuals.rsm, 82
rsm.families, 92
summary.all.nlreg.profiles, 99
summary.cond, 100
summary.marg, 103
summary.mpl, 105
summary.nlreg, 107
summary.nlreg.profile, 108
summary.rsm, 110
var2cor, 116
vcov.rsm, 117
∗Topic **models**
cond, 8
cond.glm, 9

cond.object, 13
cond.rsm, 14
expInfo, 27
logLik.nlreg, 41
logLik.rsm, 42
marg.object, 43
mpl, 46
obsInfo, 60
param, 62
plot.fr, 65
residuals.rsm, 82
rsm, 84
rsm.diag, 88
rsm.diag.plots, 90
rsm.fit, 93
rsm.null, 95
rsm.object, 96
rsm.surv, 98
summary.fr, 102
summary.rsm, 110
tem, 112
update.rsm, 114
var2cor, 116
vcov.rsm, 117

∗Topic **nonlinear**
contour.all.nlreg.profiles, 17
Dmean, 22
Dvar, 25
expInfo.nlreg, 28
mpl.nlreg, 47
mpl.object, 49
nlreg, 51
nlreg.diag, 54
nlreg.object, 57
obsInfo.nlreg, 61
plot.nlreg.contours, 69
plot.nlreg.diag, 70
profile.nlreg, 78
summary.all.nlreg.profiles, 99
summary.mpl, 105
summary.nlreg, 107
summary.nlreg.profile, 108

∗Topic **package**
hoa, 35

∗Topic **print**
print.summary.cond, 76
print.summary.marg, 77

∗Topic **regression**
cond, 8
cond.glm, 9
cond.object, 13
cond.rsm, 14
expInfo.nlreg, 28
logLik.nlreg, 41
logLik.rsm, 42
marg.object, 43
mpl.nlreg, 47
mpl.object, 49
nlreg, 51
nlreg.diag, 54
nlreg.object, 57
obsInfo.nlreg, 61
plot.cond, 63
plot.marg, 67
plot.nlreg.contours, 69
plot.nlreg.diag, 70
plot.nlreg.profiles, 74
profile.nlreg, 78
residuals.rsm, 82
rsm, 84
rsm.diag, 88
rsm.diag.plots, 90
rsm.fit, 93
rsm.null, 95
rsm.object, 96
rsm.surv, 98
summary.all.nlreg.profiles, 99
summary.cond, 100
summary.marg, 103
summary.mpl, 105
summary.nlreg, 107
summary.nlreg.profile, 108
summary.rsm, 110
update.rsm, 114
vcov.rsm, 117

aids, 3
airway, 4
all.nlreg.profiles.object, *80*
all.profiles.nlreg, *79*
anova, *86*, *97*

babies, 5

C1, 6
C2 *(*C1*)*, 6
C3 *(*C1*)*, 6

C4 (C1), 6
chlorsulfuron, *6, 7*, 7
coef, *14*, *16*, *44*, *48*, *50*, *52*
cond, 8, *11*, *16*
cond.glm, *8, 9*, 9, *14*, *63*, *65*, *101*
cond.object, *8–12*, 13, *65*, *77*, *101, 102*
cond.rsm, *8, 9*, 14, *44*, *67*, *69*, *104*
contour, *19, 20*, *52*
contour.all.nlreg.profiles, 17, *69, 70*

D, *23*, *26*
daphnia, 20
darwin, 21
deriv3, *23*, *26*
dHuber (Huber), 40
Dmean, 22, *26*, *29*, *61*
dormicum, 24
Dvar, *23*, 25, *29*, *61*

expInfo, 27, *29*, *60*, *62*
expInfo.nlreg, *28*, 28
extreme (rsm.families), 92

family, *14*, *16*, *29, 30*, *32*, *44*, *86*
family.cond, 29
family.rsm, 30, *31*, *93*
family.rsm.object, *30*, 31, *93*
family.summary.cond, 32
fitted, *52*, *58*, *86*, *97*
formula, *16*, *85, 86*
fraser.reid (tem), 112
fraudulent, 32
fungal, *11*, 33

glm, *10*, *12*, *53*, *97*
glm.control, *10*, *15*, *85*
glm.diag, *56*
glm.diag.plots, *72*

helicopter, 34
hoa, 35
houses, 39
Huber, 40, *93*
Huber (rsm.families), 92

identify, *71*, *73*, *92*

lik.ci, *66*, *113*
lik.ci (summary.fr), 102
lm, *53*, *85*, *97*

logistic (rsm.families), 92
logLik, *41, 42*
logLik.nlreg, 41
logLik.rsm, 42
logWeibull (rsm.families), 92

M2 (C1), 6
M4 (C1), 6
marg.object, *8, 9*, *16, 17*, 43, *69*, *78*, *104, 105*
methods, *8*, *28*, *46*, *60*, *62*, *116, 117*
metsulfuron, *6, 7*, 45
mpl, 46, *48*, *50*
mpl.nlreg, *46*, 47, *49, 50*, *105, 106*
mpl.object, *48*, 49, *107*

nlreg, *22, 23*, *25, 26*, *29*, *47*, 51, *54*, *58, 59*, *61*, *71*, *78*, *107*
nlreg.contours.object, *70*
nlreg.diag, *29*, 54, *58*, *71–73*
nlreg.diag.plots, *52*, *57*
nlreg.diag.plots (plot.nlreg.diag), 70
nlreg.object, *23*, *26*, *28, 29*, *46*, *48*, *50*, *52*, *53*, *57*, 57, *60*, *62*, *73*, *107, 108*
nlreg.profile.object, *80*, *110*
nlreg.profile.objects, *20*, *75*, *100*
nls, *52, 53*, *59*
nuclear, 59

obsInfo, *28, 29*, 60, *61, 62*
obsInfo.nlreg, *60*, 61
optim, *47, 48*, *52*

par, *63*, *67*, *74*
param, *48*, *50*, *52*, 62
param.nlreg, *62*
pHuber (Huber), 40
plot, *14*, *16*, *44*, *52*, *69, 70*, *75*, *79*
plot.all.nlreg.profiles (plot.nlreg.profiles), 74
plot.cond, *12*, *14*, 63
plot.fr, 65, *113*
plot.marg, *17*, *44*, 67
plot.nlreg.contours, *20*, 69
plot.nlreg.diag, 70
plot.nlreg.profile (plot.nlreg.profiles), 74
plot.nlreg.profiles, 74
plot.rsm (rsm.diag.plots), 90
print, *14*, *16*, *44*, *48*, *50*, *52*, *58*, *76, 77*, *86*, *97*

print.default, *76, 77*
print.summary.cond, 76
print.summary.marg, 77
profile, *52*, *58*, *80*
profile.nlreg, *18*, *29*, *62*, *74, 75*, 78, *99, 100*, *109, 110*

qHuber *(*Huber*)*, 40

rabbits, 81
residuals, *52*, *83*, *86*
residuals.rsm, 82, *96*
rHuber *(*Huber*)*, 40
ria, 83
rsm, *15–17*, *31*, 84, *93–99*, *111*, *114*, *117, 118*
rsm.diag, *86*, 88, *91, 92*
rsm.diag.plots, *86*, *89*, 90
rsm.distributions, *85*, *94, 95*
rsm.families, *15*, *31*, *87*, 92, *94*, *96*, *99*
rsm.fit, *86, 87*, 93, *96*, *99*
rsm.null, *86, 87*, *94*, 95, *99*
rsm.object, *41, 42*, *83*, *86, 87*, *89*, *92*, *94*, *96*, 96, *99*, *111*, *118*
rsm.surv, *86, 87*, *94*, *96*, 98

student *(*rsm.families*)*, 92
summary, *14*, *16*, *44*, *48*, *50*, *52*, *58*, *79*, *86*, *97, 99, 100*, *102*, *105–111*
summary.all.nlreg.profiles, 99
summary.cond, *12*, *14*, *65*, *76, 77*, 100
summary.fr, *66*, 102, *113*
summary.marg, *17*, *44*, *69*, *78*, 103
summary.mpl, 105
summary.nlreg, *29*, *62*, 107
summary.nlreg.profile, 108
summary.rsm, *89*, 110, *118*
survreg.fit, *86*, *99*

tem, *66*, *103*, 112
try, *56*, *80*

update, *52*, *86*, *114*
update.default, *114*
update.formula, *114*
update.rsm, 114
urine, 115

var2cor, 116
var2cor.mpl, *117*
var2cor.nlreg, *117*
vcov, *117, 118*
vcov.rsm, 117
venice, 118