Package 'onls'

September 9, 2015

Type Package

LazyLoad no

LazyData yes

Title Orthogonal Nonlinear Least-Squares Regression

Version 0.1-1

Date 2015-08-20

Author Andrej-Nikolai Spiess <`a.spiess@uke.uni-hamburg.de`>

Maintainer Andrej-Nikolai Spiess <`a.spiess@uke.uni-hamburg.de`>

Description
Orthogonal Nonlinear Least-Squares Regression using Levenberg-Marquardt Minimization.

License GPL (>= 2)

Depends R (>= 2.13.0), minpack.lm

NeedsCompilation no

Repository CRAN

Date/Publication 2015-09-09 21:19:30

R topics documented:

check_o *Check the fit for orthogonality of all points*

Description

Checks for orthogonality of all points by calculating the angle between the slope of the tangent at (x_{0i}, y_{0i}) and the slope of the Euclidean vector $\|\vec{D}_i\|$ to (x_i, y_i), which should be $90°$ if the Euclidean distance has been minimized. See 'Details'.

Usage

```
check_o(object, plot = TRUE)
```

Arguments

object	an object returned from onls.
plot	logical. If TRUE, the α-values are plotted for a quick overview of orthogonality for all points.

Details

This is a validation method for checking the orthogonality between all (x_{0i}, y_{0i}) and (x_i, y_i). The function calculates the angle between the slope m_i of the tangent obtained from the first derivative at (x_{0i}, y_{0i}) and the slope n_i of the onls-minimized Euclidean distance between (x_{0i}, y_{0i}) and (x_i, y_i):

$$\tan(\alpha_i) = \left|\frac{\mathrm{m}_i - \mathrm{n}_i}{1 + \mathrm{m}_i \cdot \mathrm{n}_i}\right|, \ \mathrm{m}_i = \frac{df(x, \beta)}{dx_{0i}}, \ \mathrm{n}_i = \frac{y_i - y_{0i}}{x_i - x_{0i}}$$

$$=> \alpha_i[°] = \tan^{-1}\left(\left|\frac{\mathrm{m}_i - \mathrm{n}_i}{1 + \mathrm{m}_i \cdot \mathrm{n}_i}\right|\right) \cdot \frac{360}{2\pi}$$

Value

A dataframe containing x_i, x_{0i}, y_i, y_{0i}, α_i, $\frac{df}{dx}$ and a logical for $89.95° < \alpha_i < 90.05°$. If plot = TRUE, a plot of the α-values in black if orthogonal, or red otherwise.

Author(s)

Andrej-Nikolai Spiess

Examples

```
## Compare 'data range extended' orthogonal model with a
## 'data range restricted' model by setting "extend = c(0, 0)"
## => some x may not be orthogonal!
x <- 1:20
y <- 10 + 3*x^2
y <- sapply(y, function(a) rnorm(1, a, 0.1 * a))
```

```
DAT <- data.frame(x, y)

mod1 <- onls(y ~ a + b * x^2, data = DAT, start = list(a = 1, b = 1))
check_o(mod1)

mod2 <- onls(y ~ a + b * x^2, data = DAT, start = list(a = 1, b = 1),
             extend = c(0, 0))
check_o(mod2)
```

`confint.onls` *Confidence intervals for 'onls' model parameters*

Description

Computes confidence intervals for one or more parameters of an `onls` model. As in `MASS:::confint.nls`, these are based on profile likelihoods, using `onls:::profile.onls` and `onls:::confint.profile.onls`.

Usage

```
## S3 method for class 'onls'
confint(object, parm, level = 0.95, ...)
```

Arguments

`object`	an object returned from `onls`.
`parm`	a specification of which parameters are to be given confidence intervals, either a vector of numbers or a vector of names. If missing, all parameters are considered.
`level`	the confidence level required.
`...`	additional argument(s) for methods.

Details

Profiling the likelihood uses the following strategy:
If θ is the parameter to be profiled and δ the vector of remaining parameters,
1) compute the log-likelihood of the model $\mathcal{L}(\theta^*, \delta^*)$ using the converged parameters,
2) compute a lower bound $\theta^* - 0.6 \cdot \sigma(\theta^*)$ for the lower confidence limit,
3) define a grid of values ranging from $\theta^{'}$ to θ^* (e.g., 100 equidistant points),
4) for each grid value θ_i, compute the profile log-likelihood value $\mathcal{L}_1(\theta_i)$ by maximizing $\mathcal{L}(\theta_i, \delta)$ over δ-values by fixing θ at θ_i,
5) find the confidence `level` by interpolation of the profile traces obtained from 4).

Value

A matrix (or vector) with columns giving lower and upper confidence limits for each parameter. These will be labelled as (1 - `level`)/2 and 1 - (1 - `level`)/2 in % (by default 2.5% and 97.5%).

Author(s)

Andrej-Nikolai Spiess, taken and modified from the `nls` functions.

Examples

```
## Not run:
DNase1 <- subset(DNase, Run == 1)
DNase1$density <- sapply(DNase1$density, function(x) rnorm(1, x, 0.1 * x))
mod1 <- onls(density ~ Asym/(1 + exp((xmid - log(conc))/scal)),
             data = DNase1, start = list(Asym = 3, xmid = 0, scal = 1))
confint(mod1)

## End(Not run)
```

`deviance_o` *Deviance of the orthogonal residuals*

Description

Returns the deviance $\sum_{i=1}^{N} \|D_i\|^2$ of the orthogonal residuals from the fitted `onls` model.

Usage

```
deviance_o(object)
```

Arguments

`object`	an object returned from `onls`.

Value

The deviance of the orthogonal residuals.

Author(s)

Andrej-Nikolai Spiess

Examples

```
DNase1 <- subset(DNase, Run == 1)
DNase1$density <- sapply(DNase1$density, function(x) rnorm(1, x, 0.1 * x))
mod <- onls(density ~ Asym/(1 + exp((xmid - log(conc))/scal)),
             data = DNase1, start = list(Asym = 3, xmid = 0, scal = 1))
deviance_o(mod)
```

logLik_o — *Log-Likelihood for the orthogonal residuals*

Description

Returns the log-likelihood as calculated from the orthogonal residuals obtained by `residuals_o`.

Usage

```
logLik_o(object)
```

Arguments

`object`	an object returned from `onls`.

Value

The log-likelihood as calculated from the orthogonal residuals.

Note

`logLik_o` has no other generic functions on top, so for calculating `AIC`, one has to apply `AIC(logLik_o(model))`, see 'Examples'. The usual `logLik` applies to the vertical residuals of the orthogonal model.

Author(s)

Andrej-Nikolai Spiess

Examples

```
DNase1 <- subset(DNase, Run == 1)
DNase1$density <- sapply(DNase1$density, function(x) rnorm(1, x, 0.1 * x))
mod <- onls(density ~ Asym/(1 + exp((xmid - log(conc))/scal)),
             data = DNase1, start = list(Asym = 3, xmid = 0, scal = 1))
logLik_o(mod)

## compare AIC of vertical versus orthogonal residuals.
AIC(mod)
AIC(logLik_o(mod))
```

`onls` *Orthogonal Nonlinear Least-Squares Regression*

Description

Determines the orthogonal nonlinear (weighted) least-squares estimates of the parameters of a nonlinear model. This is accomplished by minimizing the residual sum-of-squares of the orthogonal distances using Levenberg-Marquardt minimization in an outer loop and one-dimensional optimization for each (x_i, y_i) in an inner loop. See 'Details' for a description of the algorithm.

Usage

```
onls(formula, data = parent.frame(), start, jac = NULL,
     control = nls.lm.control(), lower = NULL, upper = NULL,
     trace = FALSE, subset, weights, na.action, window = 12,
     extend = c(0.2, 0.2), fixed = NULL, verbose = TRUE, ...)
```

Arguments

`formula`	a nonlinear model `formula` including variables and parameters. Will be coerced to a formula if necessary.
`data`	an optional data frame in which to evaluate the variables in `formula` and `weights`. Can also be a list or an environment, but not a matrix.
`start`	a named list or named numeric vector of starting estimates.
`jac`	A function to return the Jacobian.
`control`	an optional list of control settings. See `nls.lm.control` for the names of the settable control values and their effect.
`lower`	A numeric vector of lower bounds on each parameter. If not given, the default lower bound for each parameter is set to `-Inf`.
`upper`	A numeric vector of upper bounds on each parameter. If not given, the default lower bound for each parameter is set to `Inf`.
`trace`	logical value indicating if a trace of the iteration progress should be printed. Default is `FALSE`. If `TRUE`, the orthogonal residual (weighted) sum-of-squares and the parameter values are printed at the conclusion of each iteration.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`weights`	an optional numeric vector of (fixed) weights. When present, the objective function is orthogonal weighted least squares.
`na.action`	a function which indicates what should happen when the data contain NAs. The default is set by the `na.action` setting of `options`, and is `na.fail` if that is unset. The 'factory-fresh' default is `na.omit`.
`window`	a numeric value for a window around the predictor values in which the optimization takes place. Used when $n > 25$.

`extend`	a two-element vector defining an extended range of predictor values, so that (x_{0i}, y_{0i}) corresponding to (x_i, y_i) values at the beginning and end of the data can reside outside the original predictor range. See 'Details'.
`fixed`	a logical vector defining the parameters to be fixed during optimization. See 'Examples'.
`verbose`	logical. If `TRUE`, steps of the algorithm are printed to the console.
`...`	Additional optional arguments. None are used at present.

Details

In a standard nonlinear regression setup, we estimate parameters $\boldsymbol{\beta}$ in a nonlinear model $y_i = f(x_i, \beta) + \varepsilon_i$, with $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$, by minimizing the residual sum-of-squares of the vertical distances

$$\min_{\beta} \sum_{i=1}^{n} (y_i - f(x_i, \beta))^2$$

In contrast, orthogonal nonlinear regression aims to estimate parameters $\boldsymbol{\beta}$ in a nonlinear model $y_i = f(x_i + \delta_i, \beta) + \varepsilon_i$, with $\varepsilon_i, \delta_i \sim \mathcal{N}(0, \sigma^2)$, by minimizing the sum-of-squares of the orthogonal distances

$$\min_{\beta, \delta} \sum_{i=1}^{n} ([y_i - f(x_i + \delta_i, \beta)]^2 + \delta_i^2)$$

We do this by using the orthogonal distance D_i from the point (x_i, y_i) to some point (x_{0i}, y_{0i}) on the curve $f(x_i, \hat{\beta})$ that minimizes the Euclidean distance

$$\min \|D_i\| \equiv \min \sqrt{(x_i - x_{0i})^2 + (y_i - y_{0i})^2}$$

The minimization of the Euclidean distance is conducted by using an inner loop on each (x_i, y_i) with the `optimize` function that finds the corresponding (x_{0i}, y_{0i}) in some window $[a, b]$:

$$\underset{x_{0i} \in [a,b]}{\operatorname{argmin}} \sqrt{(x_i - x_{0i})^2 + (y_i - f(x_{0i}, \hat{\beta}))^2}$$

In detail, `onls` conducts the following steps:
1) A normal (non-orthogonal) nonlinear model is fit by `nls.lm` to the data. This approach has been implemented because parameters of the orthogonal model are usually within a small window of the normal model. The obtained parameters are passed to 2).
2) Outer loop: Levenberg-Marquardt minimization of the orthogonal distance sum-of-squares $\sum_{i=1}^{N} \|D_i\|^2$ using `nls.lm`, optimization of $\boldsymbol{\beta}$.
3) Inner loop: For each (x_i, y_i), find $(x_{0i}, f(x_{0i}, \hat{\beta}))$, $x_{0i} \in [a, b]$, that minimizes $\|D_i\|$, using `optimize`. Return vector of orthogonal distances $\|\vec{D}\|$.

The outer loop (`nls.lm`) scales with the number of parameters p in the model, probably with $\mathcal{O}(p)$ for evaluating the 1-dimensional Jacobian and $\mathcal{O}(p^2)$ for the two-dimensional Hessian. The inner loop has $\mathcal{O}(n)$ for finding $\min \|D_i\|$. Simulations with different number of n showed that the processor times scale linearly.

Two arguments of this function are VERY important.
1) `extend`: By default, it is set to `c(0.2, 0.2)`, which means that (x_{0i}, y_{0i}) in the inner loop are

also optimized in an extended predictor value x range of $[\min(x) - 0.2 \cdot \text{range}(x), \max(x) + 0.2 \cdot \text{range}(x)]$. This is important for the values at the beginning and end of the data, because the resulting model can display significantly different curvature if (x_{0i}, y_{0i}) are forced to be within the predictor range, often resulting in a loss of orthogonality at the end points. See `check_o` for an example. If the model should be forced to be within the given predictor range, supply `extend = c(0, 0)`.
2) `window`: defines the window $[x_{i-w}, x_{i+w}]$ in which `optimize` searches for (x_{0i}, y_{0i}) to minimize $\|D_i\|$. The default of `window = 12` works quite well with sample sizes $n > 25$, but may be tweaked.

IMPORTANT: If not all points are orthogonal to the fitted curve, `print.onls` gives a "FAILED: Only X out of Y fitted points are orthogonal" message. In this case, it is suggested to conduct a more detailed analysis using `check_o`. If any points are non-orthogonal, tweaking with either `extend` or `window` should in many cases resolve the problem.

The resulting orthogonal model houses information in respect to the (classical) vertical residuals as well as the (minimized Euclidean) orthogonal residuals.
The following functions use the **vertical** residuals:
`deviance`
`fitted`
`residuals`
`logLik`

The following functions use the **orthogonal** residuals:
`deviance_o`
`residuals_o`
`logLik_o`

Value

An orthogonal fit of class `onls` with the following list items:

`data`	the name of the `data` argument.
`call`	the matched call.
`convInfo`	a list with convergence information.
`model`	if `model = TRUE`, the model frame.
`parNLS`	the converged parameters of the normal (not orthogonal!) nonlinear model, as passed to the ONLS routine.
`parONLS`	the converged parameters of the not orthogonal nonlinear model, as obtained from the ONLS routine.
`x0`	the x_{0i} values from (x_{0i}, y_{0i}) that minimize $\|D_i\|$.
`y0`	the y_{0i} values from (x_{0i}, y_{0i}) that minimize $\|D_i\|$.
`fittedONLS`	the fitted values $\hat{y}_i$ corresponding to x_i and the orthogonal nonlinear model.
`fittedNLS`	the fitted values $\hat{y}_i$ corresponding to x_i and the non-orthogonal nonlinear model, as passed to the ONLS routine.

`residONLS`	the (vertical) residual values $y_i - \hat{y}_i$ corresponding to x_i and the orthogonal nonlinear model.
`residNLS`	the (vertical) residual values $y_i - \hat{y}_i$ corresponding to x_i and the non-orthogonal nonlinear model, as passed to the ONLS routine.
`resid_o`	the orthogonal residual values $\|\|D_i\|\|$.
`pred`	the orginal predictor values of `data`.
`resp`	the orginal response values of `data`.
`grad`	the numeric gradient of the model.
`QR`	the QR decomposition of the model.
`weights`	the weights used for fitting.
`control`	the control list used, see the `control` argument.
`ortho`	a vector of logicals for orthogonality of all points, as obtained from `check_o`.

Author(s)

Andrej-Nikolai Spiess

References

Nonlinear Perpendicular Least-Squares Regression in Pharmacodynamics.
Ko HC, Jusko WJ and Ebling WF.
Biopharm Drug Disp (1997), **18**: 711-716.

Orthogonal Distance Regression.
Boggs PT and Rogers JE.
NISTIR (1990), **89-4197**: 1-15.
`http://docs.scipy.org/doc/external/odr_ams.pdf`.

User's Reference Guide for ODRPACK Version 2.01
Software for Weighted Orthogonal Distance Regression.
Boggs PT, Byrd RH, Rogers JE and Schnabel RB. NISTIR (1992), **4834**: 1-113.
`http://docs.scipy.org/doc/external/odrpack_guide.pdf`.

ALGORITHM 676 ODRPACK: Software for Weighted Orthogonal Distance Regression.
Boggs PT, Donaldson JR, Byrd RH and Schnabel RB.
ACM Trans Math Soft (1989), **15**, 348-364.
`http://dl.acm.org/citation.cfm?id=76913`.

Examples

```
## 1A. The DNase data from 'nls',
## use all generic functions.
DNase1 <- subset(DNase, Run == 1)
DNase1$density <- sapply(DNase1$density, function(x) rnorm(1, x, 0.1 * x))
mod1 <- onls(density ~ Asym/(1 + exp((xmid - log(conc))/scal)),
             data = DNase1, start = list(Asym = 3, xmid = 0, scal = 1))
print(mod1)
plot(mod1)
summary(mod1)
```

```
predict(mod1, newdata = data.frame(conc = 6))
logLik(mod1)
deviance(mod1)
formula(mod1)
weights(mod1)
df.residual(mod1)
fitted(mod1)
residuals(mod1)
vcov(mod1)
coef(mod1)

DNase2 <- DNase1
DNase2$conc <- DNase2$conc * 2
mod2 <- update(mod1, data = DNase2)
print(mod2)

## Not run:
## 1B. Same model as above, but using the restricted
## predictor range which results in non-orthogonality
## of some points.
onls(density ~ Asym/(1 + exp((xmid - log(conc))/scal)),
     data = DNase1, start = list(Asym = 3, xmid = 0, scal = 1),
     extend = c(0, 0))

## 2. Example from odrpack_guide.pdf, 2.C.i, pages 39ff.
x <- c(0, 0, 5, 7, 7.5, 10, 16, 26, 30, 34, 34.5, 100)
y <- c(1265, 1263.6, 1258, 1254, 1253, 1249.8, 1237, 1218, 1220.6,
1213.8, 1215.5, 1212)
DAT <- data.frame(x, y)
mod3 <- onls(y ~ b1 + b2 * (exp(b3 * x) -1)^2, data = DAT,
             start = list(b1 = 1500, b2 = -50, b3 = -0.1))
deviance_o(mod3) # 21.445 as in page 47
coef(mod3) # 1264.65481/-54.01838/-0.08785 as in page 48

## 3. Example from Algorithm 676: ODRPACK, page 355 + 356.
x <- c(0, 10, 20, 30, 40, 50, 60, 70, 80, 85, 90, 95, 100, 105)
y <- c(4.14, 8.52, 16.31, 32.18, 64.62, 98.76, 151.13, 224.74, 341.35,
       423.36, 522.78, 674.32, 782.04, 920.01)
DAT <- data.frame(x, y)
mod4 <- onls(y ~ b1 * 10^(b2 * x/(b3 + x)), data = DAT,
             start = list(b1 = 1, b2 = 5, b3 = 100))
coef(mod4) # 4.4879/7.1882/221.8383 as in page 363
deviance_o(mod4) # 15.263 as in page 363

## 4. Example with bounds from simple_example.f90
## in http://www.netlib.org/toms/869.zip.
x <- c(0.982, 1.998, 4.978, 6.01)
y <- c(2.7, 7.4, 148.0, 403.0)
DAT <- data.frame(x, y)
mod5 <- onls(y ~ b1 * exp(b2 * x), data = DAT,
            start = list(b1 = 2, b2 = 0.5),
            lower = c(0, 0), upper = c(10, 0.9))
coef(mod5) # 1.4397(1.6334)/0.9(0.9) ## Different to reference!
```

```
deviance_o(mod5) # 0.1919 (0.2674) => but lower RSS than original ODRPACK!

## 5. Example with a fixed parameter
## => Asym = 3.
DNase1 <- subset(DNase, Run == 1)
DNase1$density <- sapply(DNase1$density, function(x) rnorm(1, x, 0.1 * x))
mod6 <- onls(density ~ Asym/(1 + exp((xmid - log(conc))/scal)),
             data = DNase1, start = list(Asym = 3, xmid = 0, scal = 1),
             fixed = c(TRUE, FALSE, FALSE))
print(mod6)

## 6. Example to show that one can even conduct
## linear orthogonal regression (Deming regression).
## Comparison to XLstat
## http://www.xlstat.com/de/lern-center/tutorials/
## method-comparison-with-the-deming-regression-with-xlstat-life.html
x <- c(9.8, 9.7, 10.7, 10.9, 12.4, 12.5, 12.8, 12.8, 12.9, 13.3,
       13.4, 13.5, 13.7, 14.9, 15.2, 15.5)
y <- c(10.1, 11.4, 10.8, 11.3, 11.8, 12.1, 12.3, 13.6, 14.2, 14.4,
       14.6, 15.3, 15.5, 15.8, 16.2, 16.5)
DAT <- data.frame(x, y)
mod7 <- onls(y ~ a + b * x, data = DAT,
            start = list(a = 2, b = 3))
print(mod7) ## -1.909/1.208 as on webpage
plot(mod7)

## 7. Use the more complex nonlinear models from
## the NIST database
## => http://www.itl.nist.gov/div898/strd/nls/nls_main.shtml.
onls:::NIST()

## End(Not run)
```

`plot.onls` *Plotting function for 'onls' objects*

Description

Plots an orthogonal nonlinear model obtained from `onls`.

Usage

```
## S3 method for class 'onls'
plot(x, fitted.nls = TRUE, fitted.onls = TRUE,
                      segments = TRUE,...)
```

Arguments

`x`	an object returned from `onls`.

`fitted.nls`	logical. If `TRUE`, the fit from the normal (vertical) nonlinear model is plotted as a blue line for comparison purposes.
`fitted.onls`	logical. If `TRUE`, the fit from the orthogonal nonlinear model is plotted as a red line.
`segments`	logical. If `TRUE`, segments connecting (x_i, y_i) and (x_{0i}, y_{0i}) are displayed.
`...`	other parameters to `plot`.

Value

A plot of the `onls` model.

Author(s)

Andrej-Nikolai Spiess

Examples

```
## Quadratic model with 10% added noise.
## Omitting the "nls" curve,
## display orthogonal segments.
x <- 1:20
y <- 10 + 3*x^2
y <- sapply(y, function(a) rnorm(1, a, 0.1 * a))
DAT <- data.frame(x, y)
mod <- onls(y ~ a + b * x^2, data = DAT, start = list(a = 1, b = 1))
plot(mod, fitted.nls = FALSE)

## Due to different scaling,
## orthogonality of fit is not evident.
## We need to have equal x/y-scaling for that:
plot(mod, fitted.nls = FALSE, xlim = c(0, 50),
     ylim = c(0, 50))
```

`print.onls` *Printing function for 'onls' objects*

Description

Provides a printed summary of the converged fit obtained from `onls`.

Usage

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

Arguments

`x`	an object returned from `onls`.
`...`	other parameters for future methods.

Value

A printed summary containing the formula, data name, converged parameters, vertical residual sum-of-squares

$$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

, orthogonal residual sum-of-squares

$$\sum_{i=1}^{n}\|D_i\|^2$$

, number of points with orthogonality as obtained from `check_o` and number of iterations to convergence.

Author(s)

Andrej-Nikolai Spiess

Examples

```
## See 'onls'.
```

`residuals_o` *The orthogonal residuals*

Description

Returns a vector with the orthogonal residuals $\|D_i\|$ from the fitted `onls` model.

Usage

```
residuals_o(object)
```

Arguments

`object`	an object returned from `onls`.

Value

The orthogonal residuals.

Author(s)

Andrej-Nikolai Spiess

Examples

```
DNase1 <- subset(DNase, Run == 1)
DNase1$density <- sapply(DNase1$density, function(x) rnorm(1, x, 0.1 * x))
mod <- onls(density ~ Asym/(1 + exp((xmid - log(conc))/scal)),
            data = DNase1, start = list(Asym = 3, xmid = 0, scal = 1))
residuals_o(mod)
```

`summary.onls` *Summary function for 'onls' objects*

Description

Provides a summary for the parameters of the converged fit including their error estimates, and the standard errors in respect to vertical and orthogonal residuals.

Usage

```
## S3 method for class 'onls'
summary(object, correlation = FALSE, symbolic.cor = FALSE, ...)
```

Arguments

`object`	an object returned from `onls`.
`correlation`	logical. If `TRUE`, the correlation matrix of the estimated parameters is returned and printed.
`symbolic.cor`	logical. If `TRUE`, print the correlations in a symbolic form.
`...`	further arguments passed to or from other methods.

Value

A summary similar to `summary.nls` containing the formula, parameter estimates with their errors/significance, and standard errors in respect to vertical and orthogonal residuals.

Author(s)

Andrej-Nikolai Spiess

Examples

```
## See 'onls'.
```

x0 *x0-values from orthogonal nonlinear least squares regression*

Description

Returns the x_{0i} values as obtained from minimizing the Euclidean distance

$$\min \|D_i\| \equiv \min \sqrt{(x_i - x_{0i})^2 + (y_i - y_{0i})^2}$$

Usage

```
x0(object)
```

Arguments

`object`	an object returned from `onls`.

Value

A vector of x_{0i} values.

Author(s)

Andrej-Nikolai Spiess

Examples

```
DNase1 <- subset(DNase, Run == 1)
DNase1$density <- sapply(DNase1$density, function(x) rnorm(1, x, 0.1 * x))
mod <- onls(density ~ Asym/(1 + exp((xmid - log(conc))/scal)),
            data = DNase1, start = list(Asym = 3, xmid = 0, scal = 1))
x0(mod)
```

y0 *y0-values from orthogonal nonlinear least squares regression*

Description

Returns the y_{0i} values as obtained from minimizing the Euclidean distance

$$\min \|D_i\| \equiv \min \sqrt{(x_i - x_{0i})^2 + (y_i - y_{0i})^2}$$

Usage

```
y0(object)
```

Arguments

`object`	an object returned from `onls`.

Value

A vector of y_{0i} values.

Author(s)

Andrej-Nikolai Spiess

Examples

```
DNase1 <- subset(DNase, Run == 1)
DNase1$density <- sapply(DNase1$density, function(x) rnorm(1, x, 0.1 * x))
mod <- onls(density ~ Asym/(1 + exp((xmid - log(conc))/scal)),
             data = DNase1, start = list(Asym = 3, xmid = 0, scal = 1))
y0(mod)
```

Index