Package 'onls'

November 9, 2026

Type Package

Title Orthogonal Nonlinear Least-Squares Regression

Version 0.3

Date 2026-11-09

Maintainer Andrej-Nikolai Spiess <draspiess@gmail.com>

Description Fits n-dimensional data by means of orthogonal nonlinear least-squares using Levenberg-Marquardt minimization, with a solver that exploits the block structure of the problem as 'ODRPACK' does, so that its cost grows linearly with the number of observations, and provides functionality for fit diagnostics and plotting. Delivers the same results as the 'ODRPACK' Fortran implementation described in Boggs et al. (1989) <doi:10.1145/76909.76913>, but is implemented in pure R.

License GPL (>= 2)

Depends R (>= 3.5.0)

Imports minpack.lm

Suggests rgl, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Andrej-Nikolai Spiess [aut, cre]

Repository CRAN

Date/Publication 2026-11-09 17:00:11 UTC

Contents

anova.onls 2
check_o 3
confint.onls 5
deviance_o 7
doGauge 8
logLik_o 11
onls 13
plot.onls 22

predict.onls 25
print.onls 27
profile.onls 28
residuals_o 30
summary.onls 31
x0, y0 32

Index **34**

`anova.onls` *Comparison of nested orthogonal regression models*

Description

Compares nested models fitted by `onls` to the same data with the same measurement errors, by the change in the minimized orthogonal objective S: an F test when the error variances are estimated, a likelihood-ratio chi-square test when they are known.

Usage

```
## S3 method for class 'onls'
anova(object, ...)
```

Arguments

`object`	an object returned from `onls`.
`...`	further `onls` fits to be compared with `object`, in order of increasing (or decreasing) complexity.

Details

`onls` estimates the parameters by minimizing the weighted sum of squared orthogonal distances

$$S = \sum_{i=1}^{n} d_i^2 = \sum_{i=1}^{n} \left[Qyy_i \left(y_i - f(\hat{\xi}_i, \hat{\theta}) \right)^2 + (\hat{\xi}_i - x_i)^T Qx_i (\hat{\xi}_i - x_i) \right]$$

(`objective` of the fit, see `onls`). The comparison of nested models is based on this criterion. As in `anova.nls`, each model is compared with the preceding one in the list. For a model a with df_a residual degrees of freedom and a model b with more parameters ($df_b < df_a$):

Estimated variances (`known_sigma = FALSE`): the extra-sum-of-squares F test

$$F = \frac{(S_a - S_b)/(df_a - df_b)}{S_b/df_b} \sim F_{df_a - df_b,\ df_b}$$

Known variances (`known_sigma = TRUE`): with the errors on their absolute scale, S is -2 times the log-likelihood of the errors-in-variables model up to a constant (see `logLik_o`), so that $S_a - S_b$ is the likelihood-ratio statistic, referred to a χ^2 distribution with $df_a - df_b$ degrees of freedom. The models must have been fitted to the same observations, with the same `weights`, `sigma_y` and `sigma_x` for the predictors they share, and with the same `known_sigma`.

Value

An object of class `"anova"` (a data frame) with one row per model: the residual degrees of freedom (`Res.Df`), the orthogonal objective S (`Res.Obj`), the change in degrees of freedom (`Df`) and in the objective (`Obj`) from the preceding model, and the test statistic with its p-value (`F value`, `Pr(>F)` or `Chisq`, `Pr(>Chi)`).

Author(s)

Andrej-Nikolai Spiess

See Also

`onls`, `logLik_o` for AIC/BIC of non-nested models, `profile.onls` and `confint.onls`, which use the same criterion for single parameters.

Examples

```
## Enzyme kinetics with errors in the substrate concentration (pipetting, 8%)
## and in the rate: Michaelis-Menten (h = 1) or cooperative Hill kinetics?
set.seed(42)
S_true <- rep(c(0.25, 0.5, 1, 2, 4, 8, 16, 32), each = 3)
ENZ <- data.frame(S = S_true * exp(rnorm(24, 0, 0.08)),
                  v = 10 * S_true^1.5 / (3^1.5 + S_true^1.5) + rnorm(24, 0, 0.25))

mm    <- onls(v ~ Vm * S / (K + S), data = ENZ, start = list(Vm = 10, K = 3))
hill  <- onls(v ~ Vm * S^h / (K^h + S^h), data = ENZ,
              start = list(Vm = 10, K = 3, h = 1))
hill0 <- onls(v ~ v0 + Vm * S^h / (K^h + S^h), data = ENZ,
              start = list(v0 = 0, Vm = 10, K = 3, h = 1.5))

## Sequential F tests on the orthogonal objective (variances estimated):
## the Hill exponent is needed, a baseline rate is not
anova(mm, hill, hill0)
```

---

check_o *Check the fit for orthogonality of all points*

Description

Checks, for every observation, whether the foot point (x_{0i}, y_{0i}) returned by `onls` is a genuine stationary point of the orthogonal-distance objective that `onls` minimizes, *i.e.* whether the first-order condition $\partial S/\partial \delta_i = 0$ of the joint ODRPACK-type problem (see `onls`) is met. The check is the angle between the residual vector (from the foot point to the observation) and the model curve or surface at the foot point, which is 90° at a stationary foot point. For weighted fits the angle is measured after each axis has been rescaled by its precision, so the same criterion covers unweighted, weighted, multivariate and correlated-error fits. See 'Details'.

Usage

```
check_o(object, plot = TRUE, tol_deg = 0.05, tol_kkt = NULL)
```

Arguments

`object`	an object returned from `onls`.
`plot`	logical. If `TRUE`, the angles α_i are plotted against the observation index, in black for orthogonal points and dark red otherwise, with a reference line at $90°$.
`tol_deg`	tolerance in degrees: a point is orthogonal if $\lvert\alpha_i - 90°\rvert < $ `tol_deg`. Default `0.05`, i.e. $89.95° < \alpha_i < 90.05°$.
`tol_kkt`	no longer used (the relative KKT residual of version 0.2 has been replaced by the angle in whitened coordinates); supplying it gives a warning.

Details

Stationarity condition. `onls` minimizes $S = \sum_i \left[Qyy_i(y_i - f(x_i + \delta_i, \theta))^2 + \delta_i^T Qx_i \delta_i\right]$ jointly over the parameters and the foot-point corrections $\delta_i = x_{0i} - x_i$. Setting $\partial S/\partial \delta_i = 0$ gives

$$Qx_i\,(x_{0i} - x_i) = Qyy_i\,(y_i - y_{0i})\,g_i \quad \text{with} \quad g_i = \nabla_x f(x_{0i}, \hat{\theta})$$

Whitened coordinates. Let L_i be a factor with $L_i^T L_i = Qx_i$. In the coordinates $(L_i x,\ Qyy_i^{1/2} y)$ the weighted distance d_i of `onls` is an ordinary Euclidean distance, and the condition above states that the residual vector

$$v_i = \left(L_i(x_i - x_{0i}),\ Qyy_i^{1/2}(y_i - y_{0i})\right)$$

is orthogonal to the tangent space of the model surface at the foot point, spanned by the columns of $T_i = (L_i^T,\ Qyy_i^{1/2} g_i)^T$. `check_o` computes the angle between v_i and this tangent space,

$$\cos\alpha_i = \frac{\sqrt{k_i^T (T_i^T T_i)^{-1} k_i}}{\|v_i\|};\; k_i = T_i^T v_i = Qyy_i(y_i - y_{0i})g_i - Qx_i(x_{0i} - x_i);\; T_i^T T_i = Qx_i + Qyy_i g_i g_i^T,$$

with $\|v_i\|^2 = d_i^2$. Hence $\alpha_i = 90°$ exactly when the stationarity condition holds. For a single predictor with unit precisions this is the classical angle between the residual vector and the tangent $(1, m_i)$, $m_i = df/dx$, as in earlier versions. The criterion is scale-free and uses the full precision matrix, so it is valid for correlated predictor errors (a global $p \times p$ `sigma_x`) and for predictors with very small `sigma_x`, for which the per-axis criteria of version 0.2 reported false failures.

A point is flagged orthogonal when $\lvert\alpha_i - 90°\rvert <$ `tol_deg`. If the residual vector has (numerically) zero length, $d_i^2 \le \epsilon_{mach}(x_i^T Qx_i x_i + Qyy_i y_i^2)$, the observation lies on the fitted surface, the angle is undefined, and the point is reported with `alpha = NA` and `Ortho = TRUE`; this includes observations with zero weight.

Slopes. The derivatives g_i are computed by central differences of the model formula at the foot points, evaluated in the environment of the formula, so that user-defined functions and constants used in the model are found.

Interpretation of failures. For observations with very small residuals, the angle is sensitive to the exact position of the foot point, so that a loose convergence tolerance in `onls` can lead to a few flagged points although the fit is otherwise converged. Tightening `control = list(ftol = 1e-12, ptol = 1e-12)` in `onls` sharpens the angles. Foot points held at a bound set through `extend`/`window` in `onls` are, by construction, not stationary.

Value

A data frame with one row per observation, in the original observation order (after `subset` and `na.action`): for single-predictor models ($p = 1$) the observed predictor x_i, the foot point `x0`, the observed response y_i, the foot-point response `y0`, the angle `alpha` in degrees (`NA` for an observation on the fitted curve), the model slope `df/dx` at the foot point, and a logical `Ortho`; for multivariate models ($p > 1$) the observed predictors, the foot-point predictors `x0_<name>`, the observed response, `y0`, `alpha`, the partial derivatives `df/dx_<name>` at the foot point, and `Ortho`. If `plot = TRUE`, the angles are additionally plotted.

Author(s)

Andrej-Nikolai Spiess

Examples

```
## Univariate fit
set.seed(123)
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)

## Multivariate fit => one check per predictor axis
set.seed(123)
n  <- 30
x1 <- runif(n, 1, 5); x2 <- runif(n, 1, 5)
z  <- 5 + 2 * x1 + 1.5 * x2^2 + rnorm(n, 0, 2)
DAT2 <- data.frame(x1 = x1, x2 = x2, z = z)
mod2 <- onls(z ~ b1 + b2 * x1 + b3 * x2^2, data = DAT2,
             start = list(b1 = 1, b2 = 1, b3 = 1))
check_o(mod2)

## Foot point equal to the observation (model slope zero at x = 0, dx = 0):
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)
DAT3 <- data.frame(x, y)
mod3 <- onls(y ~ b1 + b2 * (exp(b3 * x) - 1)^2, data = DAT3,
             start = list(b1 = 1500, b2 = -50, b3 = -0.1))
check_o(mod3, plot = FALSE)
```

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

Description

Computes confidence intervals for the free parameters of an onls model, from the profile of the orthogonal-distance objective (default), from the standard errors (Wald), or by a nonparametric bootstrap.

Usage

```
## S3 method for class 'onls'
confint(object, parm, level = 0.95, method = c("profile", "wald", "boot"),
        k = 200, max_tries = 10 * k, verbose = FALSE, ...)
```

Arguments

object	an object returned from onls.
parm	parameters for which intervals are computed, as names or indices; default all free (non-fixed) parameters. Fixed parameters are dropped with a warning.
level	the confidence level required.
method	"profile" (default), "wald" or "boot"; see 'Details'.
k	for method = "boot": the number of successful bootstrap fits.
max_tries	for method = "boot": the maximum number of refits attempted; if fewer than k succeed, a warning gives the count.
verbose	for method = "boot": logical, print a progress line.
...	for method = "profile": further arguments passed to profile.onls, such as maxpts.

Details

The profile and Wald intervals use the quantile q of the normal distribution when the variances are known (known_sigma = TRUE in onls) and of the t distribution with df_resid degrees of freedom otherwise.

"profile". Each parameter is held fixed on a grid around its estimate while all other free parameters and all foot points are re-optimized (profile.onls). With S the minimized orthogonal objective, the profile statistic is $\tau = \mathrm{sign}(\theta_j - \hat\theta_j)\sqrt{(S(\theta_j) - \hat S)/s^2}$, with $s^2 = \hat S/\mathrm{df}$ (estimated variances) or $s^2 = 1$ (known variances). The limits are where $\tau = \pm q$, found by interpolating the profile. Profile intervals follow the shape of the objective and can be asymmetric. Only the part of the profile on which τ increases steadily away from the estimate is used. A limit that the profile does not reach is returned as NA, with a warning that says why: most often the profile flattens out because the data do not bound the parameter on that side, or a lower/upper bound lies inside the interval.

"wald". $\hat\theta_j \pm q \cdot SE_j$, with the standard errors of vcov. Instant and accurate for well-determined parameters, but symmetric by construction, and too narrow when the objective is strongly asymmetric in a parameter.

"boot". Nonparametric case bootstrap. In each replicate, the rows of the data used in the fit (after subset and na.action) are sampled with replacement, together with their row-aligned weights, sigma_x and sigma_y, and the model is refitted with all other settings of the original call, starting

from the original estimates. Fits that do not converge are discarded; if all points of the original fit were orthogonal, refits with non-orthogonal points are discarded as well. The limits are the empirical quantiles of the `k` successful fits. At most `max_tries` refits are attempted. Case resampling is unreliable for very small n.

Value

A matrix with one row per parameter and columns giving the lower and upper confidence limits. For `method = "boot"`, the attribute `"n_boot"` holds the number of successful bootstrap fits.

Author(s)

Andrej-Nikolai Spiess

See Also

`profile.onls`, `vcov`.

Examples

```
set.seed(123)
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)                       # profile
confint(mod1, method = "wald")      # symmetric, too narrow for Asym

confint(mod1, method = "boot", k = 50) # just time optimization, consider k >= 200.
```

---

deviance_o *Deviance of the orthogonal residuals*

Description

Returns the deviance (residual sum of squares) $\sum_{i=1}^{n} \hat{d}_i^2$ of the fitted, precision-weighted orthogonal distances from the fitted `onls` model – the same quantity that `onls` itself minimizes. See 'Details' for what $\hat{d}_i$ represents in general; a single-predictor, unweighted fit is a special case where this reduces to plain squared Euclidean distance.

Usage

```
deviance_o(object)
```

Arguments

`object`	an object returned from `onls`.

Details

$\hat{d}_i$ is the fitted, precision-weighted orthogonal distance of observation i (see 'Details' in onls for its full definition and the construction of Qyy_i, Qx_i from weights, sigma_x, and sigma_y); it reduces to plain Euclidean distance only for a single-predictor, unweighted fit. deviance_o returns $\sum_{i=1}^{n} \hat{d}_i^2$, numerically identical to sum(residuals_o(object)^2) and to the value onls itself minimized (object$objective).

Value

The deviance of the fitted orthogonal (precision-weighted) distances.

Author(s)

Andrej-Nikolai Spiess

Examples

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

---

doGauge *Gauge onls against published, closed-form, ODRPACK and NIST reference values*

Description

Runs the demo gauge of the package, which fits the reference problems of orthogonal distance regression and compares every result of onls (parameters, standard errors, objective) with a reference value, and shows or writes its complete output (code and results). The result of each comparison (PASS, NOTE or DIFF) is highlighted. A summary of all comparisons is printed and returned.

Usage

```
doGauge(output = c("console", "html", "pdf"), outdir = tempdir(), fontsize = 14,
        silent = TRUE, file = system.file("demo", "gauge.R", package = "onls"))
```

Arguments

output	"console" (default): the output is printed; "html" or "pdf": it is written to the file onls-gauge.html or onls-gauge.pdf in outdir.
outdir	the directory in which the HTML or PDF file is written. The default is a temporary directory; use, *e.g.*, getwd() to keep the file.
fontsize	the font size of the output in pixels (HTML); for PDF, the size is scaled accordingly and reduced if necessary, so that the longest line fits the page width.
silent	logical. If TRUE, messages and warnings issued while the demo runs are suppressed.
file	the demo file; only needed to run a modified copy of the demo.

Details

The demo contains five kinds of comparison (column 2 of the output):

- `published`: values printed in the cited publications: the ODRPACK User's Reference Guide (example 2.C.i), Algorithm 676 (ODRPACK), the bounded example of ODRPACK95 (TOMS 869), Pearson's data with York's weights (York 1966; values of the York et al. 2004 equations, as reported by the R package 'bfsl'), Daeron & Vermeesch (2024, Table 3) and the XLSTAT Deming regression example. The tolerance is half a unit of the last printed decimal; for results of iterative ODRPACK runs, the convergence of that run is added (10^{-3} of the standard error for parameters, a relative 10^{-4} for standard errors and 10^{-7} for the objective).
- `closed form`: exact solutions computed in the demo: total least squares and generalized (correlated-error) total least squares by singular value decomposition (Golub & Van Loan 1980), and Deming regression; relative tolerance 10^{-6}.
- `ODRPACK`: ODRPACK 2.01 (Boggs et al. 1989) through `scipy.odr` 1.18.1, with the same data, model, weights and starting values, run to convergence (`sstol = partol = 1e-14`), for problems without published orthogonal-regression results: the example with separate response and predictor weights (ODRPACK's `WE`, `WD`), and the NIST problems Chwirut2, Gauss1, ENSO and the Rat43 data (Gompertz model) as orthogonal regressions with unit precisions. Parameters are compared to 10^{-3} of their standard error, standard errors to a relative 10^{-4}, the objective to a relative 10^{-7}.
- `consistency`: properties that must hold (the functions around the fit, and the agreement of the two algorithms); `PASS` if the property holds.
- `NIST`: the certified values of the NIST StRD nonlinear regression data sets Chwirut2, Gauss1, ENSO and Rat43, which refer to ordinary least squares. With a negligible `sigma_x`, the orthogonal fit becomes ordinary least squares, so `onls` must reproduce the certified parameters (relative tolerance 10^{-6}) and residual sums of squares (10^{-9}).

Results:

- `PASS`: agreement within the tolerance.
- `NOTE`: a documented discrepancy of the reference, explained in a comment of the demo (the ODRPACK95 reference run reports a higher objective than the bounded optimum; a truncated instead of rounded published value).
- `DIFF`: a deviation that is neither within the tolerance nor documented. The expected result is `DIFF = 0`.

The demo can also be run directly with `demo("gauge", package = "onls")`. It replaces the reference examples of `onls` and the function `NIST` of earlier versions.

Value

Invisibly, a list with

`file`	the path of the file written (`NULL` for `output = "console"`).
`results`	a data frame with one row per comparison: `item`, `kind` (of reference), `reference`, `onls`, `diff`, `tol` and `result`.
`summary`	a table with the numbers of `PASS`, `NOTE` and `DIFF`.

Author(s)

Andrej-Nikolai Spiess

References

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.
doi:10.1145/76909.76913

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.

Algorithm 869: ODRPACK95: A weighted orthogonal distance regression code with bound constraints.
Zwolak JW, Boggs PT and Watson LT.
ACM Trans Math Soft (2007), **33**: 27.
doi:10.1145/1268776.1268782

Least squares fitting of a straight line.
York D.
Canadian Journal of Physics (1966), **44**: 1079-1086.
doi:10.1139/p66090

Unified equations for the slope, intercept, and standard errors of the best straight line.
York D, Evensen NM, Martinez ML and De Basabe Delgado J.
American Journal of Physics (2004), **72**: 367-375.

Omnivariant generalized least squares regression.
Daeron M and Vermeesch P.
Chemical Geology (2024), **647**: 121881.
doi:10.1016/j.chemgeo.2023.121881

An analysis of the Total Least Squares problem.
Golub GH and Van Loan CF.
SIAM Journal on Numerical Analysis (1980), **17**: 883-893.
doi:10.1137/0717073

Statistical Reference Datasets (StRD), nonlinear regression.
National Institute of Standards and Technology.
https://www.itl.nist.gov/div898/strd/nls/nls_main.shtml

See Also

onls, check_o.

Examples

```
res <- doGauge()
res$summary
## HTML file in the working directory:
## doGauge(output = "html", outdir = getwd())
```

`logLik_o` *Log-likelihood of the errors-in-variables model*

Description

Returns the log-likelihood of the errors-in-variables (functional) model fitted by `onls`, in which both the response and the predictors are observed with Gaussian errors, with the precisions given by `weights`, `sigma_x` and `sigma_y`. It is computed from the weighted orthogonal distances (`residuals_o`) at the fitted parameters and foot points.

Usage

```
logLik_o(object)
```

Arguments

`object`	an object returned from `onls`.

Details

Each observation i contributes $1 + p$ Gaussian error components, the response error with precision $s^{-2} Qyy_i$ and the p predictor errors with precision matrix $s^{-2} Qx_i$, where Qyy_i and Qx_i are the precisions used by `onls` and s^2 is a common variance scale. At the fitted parameters and foot points, the exponents add up to $S = \sum_i d_i^2$, the minimized objective, and with N observations of non-zero weight and $k = N(1 + p)$ components the log-likelihood is

$$\ell = -\frac{S}{2s^2} - \frac{k}{2}\log(2\pi s^2) + \frac{1}{2}\sum_{i=1}^{N}\log Qyy_i + \frac{1}{2}\sum_{i=1}^{N}\log|Qx_i|$$

With known variances (`known_sigma = TRUE` in `onls`), $s^2 = 1$, otherwise, s^2 is replaced by its maximum-likelihood estimate S/k, giving $\ell = -\frac{k}{2}(\log(2\pi S/k)+1)+\frac{1}{2}\sum\log Qyy_i+\frac{1}{2}\sum\log|Qx_i|$, and the scale counts as one estimated parameter.

$|Qx_i|$ is the determinant of the predictor precision matrix (a scalar for one predictor; it accounts for predictor correlation in the multivariate case). Observations with zero weight are excluded, as in `logLik` for weighted fits. When `sigma_x` and `sigma_y` are scaled by a common factor and the scale is estimated, ℓ does not change.

The foot points $\hat{\xi}_i$ (the incidental parameters of the functional model) are maximized out but not counted in `"df"`: their number grows with N, so the usual asymptotics of the likelihood do not apply to them, and the maximum-likelihood estimate S/k of the scale is biased downwards (`reduced_chisq` = $S/\mathrm{df_resid}$ is the usual estimate). `logLik_o` and the `AIC`/`BIC` derived from it are therefore meant for comparing `onls` fits to the *same* data (same response, predictors and observations) with the same `weights`, `sigma_x` and `sigma_y` that differ in the model function or in the fixed or free parameters; they are not comparable with `logLik` of an `nls` fit, which assumes error-free predictors.

The ratio of the predictor to the response error variance cannot be chosen by maximizing `logLik_o`: in the functional model, the likelihood with an estimated scale is unbounded as this ratio tends to zero (or infinity), a known property of errors-in-variables models (Solari 1969). The ratio has to be known, as in Deming regression.

Version 0.3-0 changed the definition: earlier versions treated the d_i as N (instead of k) Gaussian residuals with an estimated variance, also for known variances.

Value

An object of class `c("logLik_o", "logLik")`: the log-likelihood, with attributes `"df"` (the number of free model parameters, plus one if the variance scale was estimated, see 'Details') and `"nobs"`/`"nall"` (the number of observations with non-zero weight). `AIC` and `BIC` can be applied to it.

Note

The usual `logLik` applies to the vertical residuals of the orthogonal model, i.e. to an ordinary least-squares likelihood at the *observed* predictors.

Author(s)

Andrej-Nikolai Spiess

References

The Maximum Likelihood Solution of the Problem of Estimating a Linear Functional Relationship.
Solari ME.
Journal of the Royal Statistical Society, Series B (1969), **31**: 372-375.

Examples

```
set.seed(123)
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))
logLik_o(mod1)

## AIC of the errors-in-variables model (not comparable with AIC(mod1),
## the least-squares likelihood of the vertical residuals; see 'Details').
AIC(logLik_o(mod1))

## Comparing two models fitted with the same (known) error variances:
## three- and four-parameter logistic.
m3 <- onls(density ~ Asym/(1 + exp((xmid - log(conc))/scal)), data = DNase1,
           start = list(Asym = 3, xmid = 0, scal = 1), sigma_x = 0.05, sigma_y = 0.1)
m4 <- onls(density ~ y0 + (Asym - y0)/(1 + exp((xmid - log(conc))/scal)), data = DNase1,
         start = list(y0 = 0, Asym = 3, xmid = 0, scal = 1), sigma_x = 0.05, sigma_y = 0.1)
c(m3 = AIC(logLik_o(m3)), m4 = AIC(logLik_o(m4)))
```

onls *Weighted Orthogonal Nonlinear Least-Squares Regression (Orthogonal Distance Regression)*

Description

Fits nonlinear orthogonal-distance regression (ODR, also called errors-in-variables) models in the manner of ODRPACK (Boggs, Byrd and Schnabel, 1987; Boggs et al., 1992): the model parameters and one foot-point correction per observation are estimated *simultaneously* by minimizing the same weighted sum of squared response and predictor residuals that ODRPACK minimizes, using a Levenberg-Marquardt algorithm on the joint problem. In contrast to ordinary nonlinear least squares, `onls()` allows measurement error in both predictors and response variables. Predictor uncertainty may be specified through predictor-specific standard deviations, observation-specific error structures, or full covariance matrices allowing correlated predictor errors. Response uncertainty may be supplied globally or on a per-observation basis and can be combined with observation weights. The algorithm supports single- and multi-predictor nonlinear models, parameter bounds, fixed parameters, and covariance estimation for the fitted parameters that is equivalent to the ODRPACK (Gauss-Newton) covariance.

Usage

```
onls(formula, data, start = NULL, weights = NULL, sigma_x = NULL, sigma_y = 1,
     known_sigma = NULL, extend = NULL, window = NULL, control = list(),
    lower = NULL, upper = NULL, fixed = NULL, subset = NULL, na.action = NULL,
     trace = FALSE, algorithm = c("auto", "structured", "dense"))
```

Arguments

`formula`	A two-sided nonlinear model `formula` describing the response as a nonlinear function of one or more predictors and unknown model parameters. The response must be a plain variable name (no transformation of the left-hand side).
`data`	A data frame containing the variables appearing in `formula`.
`start`	A named list (or named vector) of starting values for all model parameters.
`weights`	Optional non-negative observation weights. As in `nls`, `weights`, `subset`, `sigma_x` and `sigma_y` are evaluated in `data` first and then in the calling environment, so that columns of `data` can be named directly (*e.g.* `weights = w`, `subset = conc > 0.1`; `NA` in a logical `subset` drops the observation). The weights are incorporated into the orthogonal-distance criterion through the response precision matrix and therefore affect both the foot-point corrections and the parameter estimates. When supplied, `weights` must have length equal to the number of observations.
`sigma_x`	Specification of predictor measurement error, as a standard deviation (not a variance or a weight). Accepts `NULL`, a scalar, a length-p vector, a $p \times p$ covariance matrix, or an $n \times p$ matrix of observation-specific standard deviations (a length-n vector is accepted as shorthand for the latter when $p = 1$). See 'Details'.

`sigma_y` Response measurement error, as a standard deviation. May be either a single positive value applied to all observations or a length-n vector of observation-specific standard deviations.

`known_sigma` Logical indicating whether the supplied (or default) `sigma_x`/`sigma_y` should be treated as known measurement-error standard deviations rather than as estimated from the data. If `NULL` (default), it is set to `TRUE` if `sigma_x` is supplied or `sigma_y` is explicitly passed by the user (regardless of its value), and `FALSE` otherwise. Standard errors are scaled by the reduced chi-square statistic (see `reduced_chisq` in 'Value') only when `known_sigma = FALSE`; when `known_sigma = TRUE` the supplied variances are assumed to already be on the correct absolute scale and no such scaling is applied. If `known_sigma` is `TRUE` but `sigma_x` is not given, a warning points out that the predictor errors are then taken to have standard deviation 1 in the units of each predictor.

`extend` Optional numeric vector of length one or two. If `NULL` (default), the foot-point corrections are unbounded, as in ODRPACK. If supplied (single-predictor models only), the foot points are confined to a range extended beyond the observed predictor range by the given fractions (lower, upper; a single value is used for both ends). See 'Details'.

`window` Optional integer window width, used only if `extend` is supplied, for single-predictor models with $n > 25$: restricts the foot point of each observation to the neighborhood of the corresponding sorted observations. See 'Details'.

`control` Optional list of optimization settings. Recognized elements are `outer_max`, the *total* budget of Levenberg-Marquardt iterations (default `5000`; with `algorithm = "dense"` spent in restarts of at most 1024 iterations each, see 'Details'); `ftol` and `ptol`, the relative-reduction and relative-step convergence tolerances (as in `nls.lm.control`; default `1e-10` each); `diag`, an optional length-q vector of positive parameter scaling factors (foot-point corrections are given the scale 1; if omitted, MINPACK's adaptive scaling is used); and `lower`/`upper`, alternatives to the arguments of the same name.

`lower` Optional vector of lower bounds for the model parameters, of length q (same ordering as `start`, fixed parameters included).

`upper` Optional vector of upper bounds for the model parameters, of length q (same ordering as `start`, fixed parameters included).

`fixed` Optional logical vector with the same length and ordering as `start`. Parameters marked `TRUE` are held fixed at their supplied starting values throughout optimization. If all parameters are fixed, only the foot points are optimized (as used by `profile.onls` for one-parameter models).

`subset` Optional logical or index expression, evaluated in `data`, selecting the observations to be used for fitting. Row-aligned `weights`, `sigma_x` and `sigma_y` are subset accordingly.

`na.action` Function indicating how missing values should be handled. Row-aligned `weights`, `sigma_x` and `sigma_y` follow the removal of observations.

`trace` Logical. If `TRUE`, progress messages from the NLS warm start and the orthogonal-distance optimization (iterations or restarts) are printed.

`algorithm` The solver for the joint problem; both minimize the same objective. `"auto"` (default) uses `"structured"` for more than 100 observations or when parameter

bounds (`lower`/`upper`) are given, and `"dense"` otherwise. `"structured"` uses the block structure of the problem, as ODRPACK does, and its cost per iteration grows linearly with n. `"dense"` passes the full joint problem to `nls.lm` (the solver of **onls** 0.2), with a cost per iteration growing with n^3. Fits with the foot-point bounds `extend`/`window` always use `"dense"`. See 'Details'.

Details

Model and objective. Assume a nonlinear model $y = f(x, \theta)$, where x is a vector of p predictors and θ is a vector of q unknown model parameters. Unlike ordinary nonlinear least squares, which assumes error only in the response, ODR assumes that both the response and the predictors are observed with error. For observation i, let x_i be the observed predictor vector, y_i the observed response, δ_i the unknown correction to the predictors (so that $\xi_i = x_i + \delta_i$ is the *foot point* on the model surface), Qyy_i the response precision and Qx_i the predictor precision matrix (both constructed below). The parameters and all n corrections are estimated jointly by minimizing

$$S(\theta, \delta_1, \ldots, \delta_n) = \sum_{i=1}^{n} \left[Qyy_i \left(y_i - f(x_i + \delta_i, \theta) \right)^2 + \delta_i^T Qx_i \, \delta_i \right]$$

This is the explicit ODR problem of ODRPACK, with Qyy_i playing the role of the response weights (`WE`) and Qx_i the role of the predictor weights (`WD`). It has $q_{free} + np$ unknowns, where q_{free} is the number of free (non-`fixed`) parameters. For fixed θ, minimizing S over δ_i alone gives the weighted squared orthogonal distance of observation i to the model surface,

$$d_i^2 = Qyy_i \left[y_i - f(\hat{\xi}_i, \theta) \right]^2 + (\hat{\xi}_i - x_i)^T Qx_i (\hat{\xi}_i - x_i)$$

so that the estimate $\hat{\theta}$ is the minimizer of $\sum_i d_i^2$; the d_i are returned as `orth_dist` (`residuals_o`), and $\sum_i d_i^2$ as `objective`.

Predictor precision (`sigma_x`):
The predictor precision Qx_i used above is constructed from `sigma_x`, which is always supplied on the standard-deviation scale:
`NULL`: unit variance for every predictor, $Qx_i = I_p$ for all i
scalar σ: isotropic, $Qx_i = I_p/\sigma^2$ for all i
length-p vector $(\sigma_{x1}, \ldots, \sigma_{xp})$: diagonal, predictor-specific, but identical across observations,

$$Qx_i = \mathrm{diag}\left(1/\sigma_{x1}^2, \ldots, 1/\sigma_{xp}^2\right) \text{ for all } i$$

$p \times p$ matrix Σ_x: global covariance allowing correlated predictor errors, identical across observations,

$$Qx_i = \Sigma_x^{-1} \text{ for all } i$$

$n \times p$ matrix (or, when $p = 1$, a length-n vector as shorthand): observation-specific standard deviations $\sigma_{x,i1}, \ldots, \sigma_{x,ip}$, giving a genuinely observation-varying, diagonal-only precision

$$Qx_i = \mathrm{diag}\left(1/\sigma_{x,i1}^2, \ldots, 1/\sigma_{x,ip}^2\right)$$

Cross-predictor correlation is not supported at the per-observation level (only globally, via the $p \times p$ form above), since that would require a separate $p \times p$ matrix per observation. In every case except the last, Qx_i is the same fixed matrix/vector for all observations and is simply written Qx. If σ_x is (nearly) zero for a predictor that is in fact measured without error (for example a time

index), supplying a small `sigma_x` makes the fit approach ordinary nonlinear least squares in that direction. With the default unit precisions, the relative scale of the predictors and the response directly determines how much of the misfit is attributed to x rather than y.

Response precision (`sigma_y`, `weights`): The baseline response precision is $1/\sigma_{y,i}^2$, using either the common `sigma_y` or, if a length-n vector is supplied, the observation-specific value. If observation weights w_i are supplied,

$$Qyy_i = w_i/\sigma_{y,i}^2$$

so that observation weights act multiplicatively on the response precision and thus enter the objective S directly (affecting both the foot-point corrections and the parameter estimates), rather than acting solely as conventional `nls`-style regression weights. Weights act on the response side only; the predictor precision Qx_i is not scaled by them.

Relation to ODRPACK weights (`WE`, `WD`). ODRPACK takes a response weight `WE` and a predictor weight `WD` for every observation, both as precisions (inverse variances). In `onls` they correspond to $Qyy_i = WE_i$ and $Qx_i = WD_i$ (a diagonal matrix if $p > 1$), and are specified as `weights = WE` (together with the default `sigma_y = 1`; alternatively `sigma_y = 1/sqrt(WE)` without `weights`) and `sigma_x = 1/sqrt(WD)` (for $p > 1$ an $n \times p$ matrix with entries $1/\sqrt{WD_{ij}}$). ODRPACK always scales the standard errors by the residual variance, which corresponds to `known_sigma = FALSE` (supplying `sigma_x` or `sigma_y` otherwise sets `known_sigma = TRUE`). A full weight matrix that differs between observations (a three-dimensional `WD` in ODRPACK) is not supported, only a global covariance matrix (see `sigma_x`). The demo gauge (`doGauge`) fits the ODRPACK example with separate `WE` and `WD` weights.

Algorithm.

1) *Warm start.* A conventional nonlinear least-squares model is fitted using `nlsLM` (optionally weighted by `weights`; `fixed` parameters are substituted into the model formula) to obtain starting values $\theta^{(0)}$. If this fit fails, a warning is issued and the raw `start` values are used instead.
2) *Joint Levenberg-Marquardt problem.* With L_i the factor satisfying $L_i^T L_i = Qx_i$ (elementwise square root for a diagonal Qx_i, Cholesky factor for a correlated global Σ_x), define the residual vector of length $n + np$

$$r(\theta, \delta) = \left(\left\{ Qyy_i^{1/2} \left(y_i - f(x_i + \delta_i, \theta) \right) \right\}_{i=1}^{n},\ \{L_i \delta_i\}_{i=1}^{n} \right)$$

so that $S = r^T r$. Starting at $(\theta^{(0)}, \delta = 0)$, the Levenberg-Marquardt iteration (Moré 1978; `nls.lm`) repeatedly solves

$$\left(J^T J + \mu D^2 \right) \Delta z = -J^T r \qquad z = (\theta_{free}, \delta_1, \ldots, \delta_n)$$

with a trust-region controlled damping μ and a diagonal scaling D (the running maximum of the Jacobian column norms, or `control$diag`). The Jacobian of r has the sparse 'arrow' structure of ODRPACK,

$$J = \begin{pmatrix} -W_y^{1/2} F_\theta & -W_y^{1/2} G \\ 0 & L_x \end{pmatrix}$$

where $W_y = \mathrm{diag}(Qyy_i)$, F_θ is the $n \times q_{free}$ matrix with rows $\partial f(\xi_i, \theta)/\partial \theta^T$, G is the $n \times np$ block-diagonal matrix whose i-th row holds $g_i^T = \partial f(\xi_i, \theta)/\partial x^T$ in the columns belonging to δ_i, and $L_x = \mathrm{blockdiag}(L_1, \ldots, L_n)$. Each correction δ_i thus affects only observation i. The two algorithms differ only in how the step is computed:
`algorithm = "structured"` (default for $n > 100$ or with parameter bounds). In the normal equations, the δ-part is block diagonal, with one $p \times p$ block $D_i = Qyy_i g_i g_i^T + Qx_i$ (+ damping) per

observation. These blocks are eliminated (Schur complement, as in ODRPACK), so that each step needs one $q_{free} \times q_{free}$ solve and n small block solves (Sherman-Morrison for diagonal Qx_i, a Cholesky factorization vectorized over the observations for a correlated Σ_x):

$$\Big(A + \mu D_\theta^2 - \sum_i B_i \tilde{D}_i^{-1} B_i^T\Big)\Delta\theta = -g_\theta + \sum_i B_i \tilde{D}_i^{-1} g_{\delta,i} \qquad \Delta\delta_i = -\tilde{D}_i^{-1}\big(g_{\delta,i} + B_i^T \Delta\theta\big)$$

with $A = F_\theta^T W_y F_\theta$, $B_i = Qyy_i\, \partial f_i/\partial\theta\, g_i^T$, $\tilde{D}_i = D_i + \mu D_{\delta,i}^2$, and g_θ, $g_{\delta,i}$ the parts of $J^T r$. The damping μ is chosen by MINPACK's trust-region rule (Moré 1978): $\mu = 0$ (a Gauss-Newton step) when that step lies inside the trust region, otherwise μ such that $\|D\Delta z\|$ matches its radius. Parameter bounds `lower`/`upper` are handled by holding a parameter whose step would leave its bounds at the bound and re-solving for the others. The cost per iteration is $O(np^3 + nq^2 + q^3)$, linear in n; a fit with $n = 100\,000$ takes seconds. For small n its per-iteration overhead in R is higher than that of the compiled **minpack.lm** code, so near-degenerate fits that run thousands of iterations (for example some bootstrap resamples in `confint.onls`) can take longer than with `"dense"`.

`algorithm = "dense"`. The joint problem is passed as one nonlinear least-squares problem with the full $(n + np) \times (q + np)$ Jacobian to `nls.lm`, whose dense QR factorization costs $O((n+q)^3)$ per iteration (about 15 s per fit at $n = 1000$). It is kept for comparison and is the only algorithm that supports the foot-point bounds `extend`/`window`.

Both algorithms give the same estimates up to the convergence tolerance on well-posed problems. For models with several local minima (for example periodic models with free periods) they can end in different minima, because their iterates differ slightly.

3) *Analytic derivatives.* The blocks F_θ and G are supplied to the solver exactly, by symbolic differentiation of the model formula (`deriv`) when possible, and by central finite differences otherwise (for example if the formula calls a user-defined function). This matters: MINPACK's own forward-difference Jacobian uses a step proportional to the size of each unknown, which becomes smaller than the floating-point resolution of $f(x_i + \delta_i)$ once a correction δ_i is small, and would then drive that correction to exactly zero, i.e. to a foot point that is not orthogonal.

4) *Iterations and convergence.* The structured algorithm iterates until convergence (codes 1 to 3, as in `nls.lm`, giving `convergence = 0`) or until `control$outer_max` iterations are used (`convergence = 1`, with a warning). **minpack.lm** caps a single `nls.lm` call at 1024 iterations, so the dense algorithm is restarted from its last iterate (which also resets the trust region) until it converges (`nls.lm` codes 1 to 4), a restart no longer reduces the objective, or the budget is exhausted. Convergence is declared with the tolerances `ftol` and `ptol` (default `1e-10`). If the joint optimization fails to run at all, the NLS estimates with zero corrections are returned, with a warning.

Stationarity (first-order) conditions. At a solution, $\partial S/\partial\delta_i = 0$ and $\partial S/\partial\theta = 0$ give

$$Qx_i(\hat{\xi}_i - x_i) = Qyy_i\left(y_i - f(\hat{\xi}_i, \hat{\theta})\right)\nabla_x f(\hat{\xi}_i, \hat{\theta}) \qquad \sum_{i=1}^{n} Qyy_i\left(y_i - f(\hat{\xi}_i, \hat{\theta})\right)\frac{\partial f(\hat{\xi}_i, \hat{\theta})}{\partial\theta} = 0$$

For unit precisions the first condition states that the vector from the foot point to the observation is orthogonal to the model surface at the foot point, which is what `check_o` verifies, as the angle between the residual vector and the model surface after each axis has been rescaled by its precision.

Optional foot-point bounds (`extend`, `window`). By default the corrections δ_i are unbounded, as in ODRPACK. For single-predictor models, supplying `extend` confines the foot points to $[\mathrm{XLOW}, \mathrm{XUPP}]$,

$$\mathrm{XLOW} = \min(x) - \mathrm{extend}[1] \times \mathrm{range}(x) \qquad \mathrm{XUPP} = \max(x) + \mathrm{extend}[2] \times \mathrm{range}(x)$$

by bounding δ_i. If, in addition, `window` is given and $n > 25$, the foot point of observation i (in sorted predictor order) is confined to $[x_{(i-\mathrm{window}+1)},\ x_{(i+\mathrm{window}-1)}]$, clipped at the two ends of the

data to XLOW and XUPP. With bounds, the bounded variant of the Levenberg-Marquardt algorithm is used, and a foot point sitting at a bound is generally not orthogonal. The bounded variant can also stop before the optimum, in particular with window; onls then warns when foot points are neither orthogonal nor at their bound. window has no effect unless extend is supplied; for multivariate models ($p > 1$) neither argument is used. These bounds are rarely needed and are kept for backward compatibility and for models whose foot points must be kept on one branch of the curve.

Fixed parameters: If fixed is supplied, parameters flagged TRUE are held, throughout both the NLS warm start and the joint optimization, at their user-supplied start value (not at the NLS warm-start estimate). Fixed parameters contribute neither to the residual degrees of freedom (df_resid $= n - q_{free}$, with q_{free} the number of free parameters) nor to the Jacobian columns used for vcov/std_errors; their standard error is reported as 0. Observations with zero weight do not count towards df_resid.

Covariance of the parameter estimates. The covariance is the ODRPACK covariance: the θ-block of the inverse Gauss-Newton matrix $(J^T J)^{-1}$ of the joint problem (second-derivative terms are neglected, as in ODRPACK). Eliminating the δ-blocks of $J^T J$ exactly (Schur complement, using the Woodbury identity) yields the compact form used by onls, valid for any number of predictors p:

$$\widehat{\mathrm{Var}}(\hat{\theta}_{free}) = \left(F_\theta^T W F_\theta\right)^{-1} \qquad \widehat{\mathrm{Var}}(\hat{\theta}_{free}) = \hat{\sigma}^2 \left(F_\theta^T W F_\theta\right)^{-1}$$

for known_sigma = TRUE and known_sigma = FALSE, respectively, and with F_θ evaluated at the final foot points $\hat{\xi}_i$ and estimates $\hat{\theta}$, $W = \mathrm{diag}(w_1, \ldots, w_n)$ and the effective ("effective-variance") observation weights

$$w_i = \left(Qyy_i^{-1} + g_i^T\, Qx_i^{-1}\, g_i\right)^{-1} \qquad g_i = \nabla_x f(\hat{\xi}_i, \hat{\theta})$$

which combine the response error and the predictor error propagated through the local model gradient (for $p = 1$, $w_i = (1/Qyy_i + s_i^2/Qx_i)^{-1}$ with slope s_i). Here $\hat{\sigma}^2 = \left(\sum_i d_i^2\right)/$df_resid is the reduced chi-square (reduced_chisq in 'Value'). Standard errors are the square roots of the diagonal.

Starting values, local minima and degenerate solutions. The objective can have several local minima (for example for periodic models with free periods, or for models that contain a limiting case such as the Richards curve, whose limit is the Gompertz curve), and the solution found depends on start. Starting values that make the model insensitive to some parameters (for example a sigmoid whose exponent underflows for all observations, so that the model is a constant) create a degenerate stationary point at which the solver may report convergence; onls then issues a warning naming the parameters that have no measurable influence on the fit, and the results should be discarded. A low orthogonal residual sum of squares is not by itself evidence of a good fit: with unit precisions and a steep model, an ODR fit can lower the objective by shifting observations horizontally. Compare the vertical residuals (residuals) with those of the NLS fit at parNLS, and choose sigma_x/sigma_y to reflect the actual measurement errors.

The returned object contains both classical vertical residual information and orthogonal-distance quantities such as foot points (xi, yi; the predictor corrections are xi - X_obs) and weighted orthogonal distances.

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. Because orthogonality of the foot points is the stationarity condition $\partial S/\partial\delta_i = 0$, the most common cause is a convergence tolerance that is too loose for observations with very small residuals, for which the orthogonality angle is very sensitive to

the foot-point position: tightening `control = list(ftol = 1e-12, ptol = 1e-12)` (or increasing `control$outer_max` if the iteration budget was exhausted) will normally resolve it. Foot points held at a bound set through `extend`/`window` are not expected to be orthogonal.

ALSO IMPORTANT: Regular R-like notation such as `lm(y ~ x1 + x2)` or `lm(y ~ x1 * x2)` needs to be changed to classical notation $y = b_0 + b_1x1 + b_2x2$ or $y = b_0 + b_1x1 + b_2x2 + b_3x1x2$, respectively.

The resulting orthogonal model houses information in respect to the (classical) vertical residuals as well as the (minimized, precision-weighted) 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` `anova.onls`

Value

An orthogonal fit of class `onls` (also inheriting from `nls`), a list with the following elements:

`call`, `data`, `formula`
: The matched call, the `data` argument as given (unevaluated) and the model formula.

`algorithm`
: The algorithm used, `"structured"` or `"dense"` (see `algorithm`).

`coefficients`
: Named vector of the final parameter estimates.

`std_errors`, `vcov`
: Standard errors (`0` for `fixed` parameters) and covariance matrix of the estimates; see 'Details'.

`parNLS`
: Parameter estimates of the initial nonlinear least-squares fit (the starting values, if the warm start failed; fixed parameters at their starting values).

`convergence`, `convInfo`
: Convergence code (`0` if the joint optimization converged, termination codes 1 to 4; `1` otherwise) and details: `isConv`, `finIter` (total number of Levenberg-Marquardt iterations), `finTol` (the `ftol` used), `stopCode` and `stopMessage`.

`objective`
: The minimized objective $S = \sum_i d_i^2$.

`reduced_chisq`
: Reduced chi-square $S/\mathrm{df_resid}$.

`known_sigma`
: Logical: were the variances treated as known?

`df_resid`
: Residual degrees of freedom: the number of observations with non-zero weight minus the number of free parameters.

`n`
: Number of observations used (after `subset` and `na.action`).

`X_obs`, `y_obs`
: Observed predictors ($n \times p$ matrix) and responses.

`xi`, `yi`
: Foot points $\hat{\xi}_i$ ($n \times p$ matrix) and foot-point responses $f(\hat{\xi}_i, \hat{\theta})$; see `x0`.

`fittedONLS` Model evaluated at the *observed* predictors, $f(x_i, \hat{\theta})$; the vertical residuals are `y_obs - fittedONLS` (`residuals`).

`orth_dist` Weighted orthogonal distances d_i.

`weights` Observation weights.

`Q_yy`, `Lx` Response precisions Qyy_i (length n) and predictor precisions as an $n \times p \times p$ array of upper Cholesky factors, $L_i^T L_i = Qx_i$: one representation for every form of `sigma_x`.

`sigma_x`, `sigma_y`, `per_obs_x`
`sigma_x` and `sigma_y` as used (after `subset`/`na.action`), and whether `sigma_x` was given per observation.

`fixed`, `lower`, `upper`, `control`, `extend`, `window`
The arguments of the same names, used when the model is refitted by `profile.onls` and `confint.onls`.

`na.action`, `dataClasses`
Information on omitted observations, and the classes of the model variables (used by `predict`).

`ortho` Logical vector, one value per observation: the `Ortho` column of `check_o` (`NA` if it could not be computed).

All per-observation elements are in the original observation order. Version 0.3-0 removed the duplicated elements of earlier versions (`parONLS`, `x0`/`y0`, `resid_o`, `pred`/`resp`, `residONLS`, `residNLS`, `fittedNLS`, `grad`, `QR`, `Q_x`, `Sigma_x`, `model`, ...); use `coef`, `x0`/`y0`, `residuals_o`, `residuals` and `fitted` instead.

Author(s)

Andrej-Nikolai Spiess

References

A stable and efficient algorithm for nonlinear orthogonal distance regression.
Boggs PT, Byrd RH and Schnabel RB.
SIAM J Sci Stat Comput (1987), **8**: 1052-1078.
doi:10.1137/0908085.

The Levenberg-Marquardt algorithm: implementation and theory.
More JJ.
In: Watson GA (ed.), Numerical Analysis, Lecture Notes in Mathematics **630**, Springer (1978): 105-116.
doi:10.1007/BFb0067700.

Orthogonal Distance Regression.
Boggs PT and Rogers JE.
NISTIR (1990), **89-4197**: 1-15.
https://static.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.
https://static.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.
doi:10.1145/76909.76913.

Omnivariant generalized least squares regression.
Daeron M and Vermeesch P, Chemical Geology (2024), **647**: 121881.
doi:10.1016/j.chemgeo.2023.121881.

An analysis of the Total Least Squares problem.
Golub GH and Van Loan CF.
SIAM Journal on Numerical Analysis (1980), **17**, 883-893.
doi:10.1137/0717073.

Least squares fitting of a straight line with correlated errors.
York D.
Earth and Planetary Science Letters (1968), **5**, 320-324.
doi:10.1016/S0012821X(68)800597.

Examples

```
## 1. The DNase data from 'nls', use all generic functions.
DNase1 <- subset(DNase, Run == 1)
set.seed(123)
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)

## 2a. Update model
DNase2 <- DNase1
DNase2$conc <- DNase2$conc * 2
mod2a <- update(mod1, data = DNase2)
print(mod2a)

## 2b. Example with a fixed parameter
## => Asym = 3.
mod2b <- 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(mod2b)

## 3. Multivariate example: matched curvature,
## low noise, decorrelated predictors
set.seed(123)
n  <- 25
x1 <- runif(n, 1, 5)
x2 <- runif(n, 1, 5)
b1_true <- 5
b2_true <- 2
b3_true <- 1.5
z_true <- b1_true + b2_true * x1 + b3_true * x2^2
z  <- z_true + rnorm(n, 0, 2)
x1 <- x1 + rnorm(n, 0, 0.5)
x2 <- x2 + rnorm(n, 0, 0.5)
DAT <- data.frame(x1 = x1, x2 = x2, z = z)
mod3 <- onls(z ~ b1 + b2 * x1 + b3 * x2^2, data = DAT,
             start = list(b1 = 1, b2 = 1, b3 = 1), trace = TRUE)
print(mod3)

## The reference tests against published ODR results, closed-form solutions, ODRPACK and
## the NIST certified values are collected in one demo: doGauge(), or
## demo("gauge", package = "onls").


## 4. The two algorithms: same fit, very different cost for large n.
set.seed(1)
n <- 400
xt <- seq(0, 10, length.out = n)
DAT <- data.frame(x = xt + rnorm(n, 0, 0.2), y = 5/(1 + exp(-(xt - 5))) + rnorm(n, 0, 0.2))
system.time(m_s <- onls(y ~ b1/(1 + exp(-(x - b2)/b3)), data = DAT,
                   start = list(b1 = 4, b2 = 4, b3 = 1.5)))   # auto: structured for n > 100
system.time(m_d <- onls(y ~ b1/(1 + exp(-(x - b2)/b3)), data = DAT,
                        start = list(b1 = 4, b2 = 4, b3 = 1.5), algorithm = "dense"))
rbind(structured = coef(m_s), dense = coef(m_d))
```

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

Description

Plots orthogonal nonlinear models obtained from `onls`: the observed data, the fitted `onls` curve (and, optionally, the ordinary `nls` warm-start curve), and segments connecting each observation (x_i, y_i) to its foot point (x_{0i}, y_{0i}).

Usage

```
## S3 method for class 'onls'
```

```
plot(x, panel = NULL, fitted.nls = TRUE, fitted.onls = TRUE, asp = TRUE,
                    segments = TRUE, npoints = 200, nmesh = 25, ...)
```

Arguments

x	an object returned from onls.
panel	for multivariate models only (more than one predictor): an integer index or predictor name selecting a single predictor to plot as one full-size plot, with the same treatment (including asp = 1 by default) as a single-predictor model. If NULL (default), all predictors are plotted together as a grid of smaller panels instead (see 'Details'). Ignored, with a warning, for single-predictor models.
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.
asp	logical. If TRUE, plots are generated in a 1:1 axis ratio for exact display of the orthogonality.
segments	logical. If TRUE, segments connecting (x_i, y_i) and (x_{0i}, y_{0i}) are displayed.
npoints	number of points used to draw the smooth fitted.onls/fitted.nls curve(s).
nmesh	number of nmesh x nmesh divisions used to draw the 3D mesh when two predictors are used.
...	other parameters to plot, such as xlim, ylim or main. A numeric aspect ratio cannot be passed here, because asp is matched to the logical argument above.

Details

There are three plot types, depending on the model and on panel:

Single-predictor plot (p = 1; single-predictor models, or a multivariate model with panel set to one predictor):
a single, full-size plot of y against that predictor, with asp = 1 used by default (unless overridden via ...). Under asp = 1, one data unit in x is rendered the same physical length as one data unit in y, so for an *unweighted* onls fit the segments connecting (x_i, y_i) to (x_{0i}, y_{0i}) genuinely appear at right angles to the fitted curve – a visual check of what check_o verifies numerically. If only one of xlim/ylim is supplied via ..., the other is automatically chosen (via extendrange on the data that falls inside the supplied range) so that both remain sensibly matched to the data before asp = 1 does its own adjustment; supplying only one of the two is therefore usually enough to zoom in on a region of interest.

3D plot (p = 2; bivariate models):
An rgl-based 3D plot (package 'rgl' must be installed) with x1/x2/z setup, and where the segments connect the estimated values to the foot points on the 3D surface. The advantage of this plot is the rotation- and zoom-ability to inspect orthogonality of the points.

Multivariate grid ($p > 2$; multivariate models):
One partial-dependence panel per predictor, each showing y against one predictor with the *other* predictors held fixed at their mean foot-point value. asp is *not* forced to 1 here (unless explicitly supplied via ...): panels are shrunk by the mfrow layout, and forcing asp = 1 in a small panel can require aggressively re-expanding the axes to preserve a strict 1:1 unit scale, which would silently

override any xlim/ylim requested for that panel. These grid panels are also not a literally faithful orthogonality check to begin with: since the curve fixes the *other* predictors at a shared mean value while a given point's own segment endpoint uses its own foot point in every dimension, the two generally do not coincide exactly on the drawn curve, so segments should not be expected to look exactly perpendicular even for an unweighted fit. Use check_o for a reliable per-observation orthogonality check rather than eyeballing these panels; use panel to get a single full-size, asp = 1 plot for one predictor at a time instead.

Value

A plot of the onls model (single-predictor plot, single chosen panel, multivariate grid or 3D).

Author(s)

Andrej-Nikolai Spiess

Examples

```
## 1a. Quadratic model with added noise (SD 50).
set.seed(123)
x <- 1:20
y <- 10 + 3*x^2
y <- y + rnorm(20, 0, 50)
DAT <- data.frame(x, y)
mod1 <- onls(y ~ a + b * x^2, data = DAT, start = list(a = 10, b = 3), extend = c(0.2, 0))
plot(mod1)

## 1b. Zooming in on a region: supplying xlim alone is enough --
## a matching ylim is chosen automatically before asp = 1 is applied.
## Need to set asp = FALSE, but a 1-to-1 aspect is not exactly possible.
plot(mod1, fitted.nls = FALSE, xlim = c(0, 10), asp = FALSE)

## 2. Half-dome 3D example with rgl plot
set.seed(123)
n <- 60
r_true <- 6
ang <- runif(n, 0, 2 * pi)
rad <- sqrt(runif(n, 0, 0.55)) * r_true
x1  <- rad * cos(ang)
x2  <- rad * sin(ang)
z <- sqrt(r_true^2 - x1^2 - x2^2) + rnorm(n, 0, 0.15)
x1 <- x1 + rnorm(n, 0, 0.1)
x2 <- x2 + rnorm(n, 0, 0.1)
DAT <- data.frame(x1 = x1, x2 = x2, z = z)
maxrad <- max(sqrt(x1^2 + x2^2))
mod2 <- onls(z ~ sqrt(r^2 - x1^2 - x2^2), data = DAT,
                 start = list(r = r_true),
                 sigma_x = c(0.1, 0.1), sigma_y = 0.15,
                 lower = maxrad * 1.05, upper = 100)
check_o(mod2) # all orthogonal to dome surface
if (requireNamespace("rgl", quietly = TRUE)) plot(mod2)    # a visibly ROUND dome
```

```
## 3. Multivariate setup with 4 predictors
set.seed(123)
n  <- 60
x1 <- runif(n, 0, 10)
x2 <- runif(n, 0, 5)
x3 <- runif(n, 2, 10)
x4 <- runif(n, 2, 10)

b0 <- 2; b1 <- 0.8; b2 <- 0.5; b3 <- 2; b4 <- 3
z_true <- b0 + b1 * x1 + b2 * x2^2 + b3 * sqrt(x3) + b4 * log(x4 + 1)
z <- z_true + rnorm(n, 0, 0.5)

sd_x <- c(0.3, 0.2, 0.3, 0.3)
x1 <- x1 + rnorm(n, 0, sd_x[1])
x2 <- x2 + rnorm(n, 0, sd_x[2])
x3 <- x3 + rnorm(n, 0, sd_x[3])
x4 <- x4 + rnorm(n, 0, sd_x[4])

DAT <- data.frame(x1 = x1, x2 = x2, x3 = x3, x4 = x4, z = z)
mod3 <- onls(z ~ b0 + b1 * x1 + b2 * x2^2 + b3 * sqrt(x3) + b4 * log(x4 + 1),
            data = DAT, start = list(b0 = 1, b1 = 1, b2 = 1, b3 = 1, b4 = 1),
            sigma_x = sd_x, sigma_y = 0.5)

summary(mod3)
check_o(mod3)
plot(mod3)
```

`predict.onls` *Predictions, confidence and prediction intervals from an onls fit*

Description

Evaluates the fitted `onls` model at new predictor values, optionally with standard errors and with confidence intervals for the model function or prediction intervals for a new observation in the errors-in-variables model.

Usage

```
## S3 method for class 'onls'
predict(object, newdata, se.fit = FALSE, scale = NULL, df = NULL,
        interval = c("none", "confidence", "prediction"), level = 0.95,
        sigma_x = NULL, sigma_y = NULL, ...)
```

Arguments

`object`	an object returned from `onls`.
`newdata`	a data frame with columns for all predictors. If missing, the observed predictors of the fit are used.

`se.fit` logical: return the standard errors of the fitted values?

`scale` the residual standard deviation s. Default: 1 with known variances, otherwise $\sqrt{\text{reduced_chisq}}$. If given, the standard errors of the fitted values are rescaled to it too, as in `predict.lm`.

`df` the degrees of freedom of the t quantiles. Default: `Inf` (normal quantiles) with known variances, otherwise `df_resid`.

`interval` type of interval: none, a confidence interval for the model function $f(x, \theta)$, or a prediction interval for a new observation y at the observed (error-prone) predictor x.

`level` the confidence level.

`sigma_x, sigma_y` the standard deviations of the predictor and response errors of the new observations, in any form accepted by `onls`: a scalar, a length-p vector, a $p \times p$ covariance matrix (a square $p \times p$ matrix is always taken as a covariance, also if `newdata` has p rows; give per-row standard deviations for p rows as a data frame), or one row of standard deviations per row of `newdata`; `sigma_y` of length 1 or `nrow(newdata)`. Needed for `interval = "prediction"` only if the fit used observation-specific values; otherwise those of the fit are used.

`...` further arguments, ignored.

Details

With J the gradient of the model function with respect to the free parameters at `newdata` and V their covariance (`vcov`), the standard error of the fitted value is $se_{\text{fit}} = \sqrt{JVJ^T}$.

A prediction interval refers to a new observation whose *response and predictors* are both measured with error. Its variance is the effective variance of ODR (as in the parameter covariance, see `onls`) plus the variance of the fitted value:

$$se^2 = se_{\text{fit}}^2 + s^2 \left(\sigma_{y,\text{new}}^2 + g^T \Sigma_{x,\text{new}}\, g\right),$$

where g is the gradient of the model function with respect to the predictors at `newdata`, and $\Sigma_{x,\text{new}}$ the predictor covariance of the new observation. The quantiles are taken from the normal distribution for known variances (`known_sigma = TRUE`) and from the t distribution with `df_resid` degrees of freedom otherwise, as in `summary.onls` and `confint.onls`.

A prediction interval at the *observed* predictor is the relevant one for a future measurement pair (x, y). If the predictor of the new observation is known exactly, use `sigma_x = 0` (*i.e.* the response error only). A new observation has weight 1: if the fit used non-constant `weights`, a warning suggests to supply its `sigma_y`.

Value

If `se.fit = FALSE` and `interval = "none"`, a vector of fitted values. With an interval, a matrix with columns `fit`, `lwr` and `upr`. If `se.fit = TRUE`, a list with the components `fit` (vector or matrix as before), `se.fit` (the standard errors of the fitted values), `df` and `residual.scale` (s).

Author(s)

Andrej-Nikolai Spiess

See Also

onls, confint.onls, summary.onls.

Examples

```
## straight line, known error variances in x and y
set.seed(1)
xt <- 1:20; x <- xt + rnorm(20, 0, 0.5); y <- 2 + 0.5 * xt + rnorm(20, 0, 0.3)
mod <- onls(y ~ a + b * x, data.frame(x, y), start = list(a = 1, b = 1),
            sigma_x = 0.5, sigma_y = 0.3)
nd <- data.frame(x = c(5, 15))
predict(mod, nd, interval = "confidence")
predict(mod, nd, interval = "prediction")
## new observations with an error-free predictor
predict(mod, nd, interval = "prediction", sigma_x = 0)
```

---

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

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

(prefixed "weighted" when weights are non-constant), orthogonal residual sum-of-squares

$$\sum_{i=1}^{n} \hat{d}_i^2$$

(see onls for the general, precision-weighted definition of $\hat{d}_i$; plain Euclidean distance is a special case), a "PASSED"/"FAILED" message reporting how many fitted points are orthogonal as obtained from check_o, (or a note that it could not be checked), the number of iterations to convergence and the convergence tolerance ftol that was used.

Author(s)

Andrej-Nikolai Spiess

Examples

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

`profile.onls` *Profiles of the orthogonal-distance objective*

Description

Profiles the minimized orthogonal-distance objective of an `onls` fit for each free parameter, and plots the profiles or turns them into confidence intervals.

Usage

```
## S3 method for class 'onls'
profile(fitted, which = NULL, maxpts = 100, alphamax = 0.01,
        delta.t = cutoff/5, ...)
## S3 method for class 'profile.onls'
plot(x, absVal = TRUE, level = 0.95, ...)
## S3 method for class 'profile.onls'
confint(object, parm, level = 0.95, ...)
```

Arguments

`fitted`	an object returned from `onls`.
`which`	parameters to profile, as names or indices; default all free parameters. Fixed parameters are not profiled.
`maxpts`	maximum number of points in each direction of a profile.
`alphamax`	the profile is extended in each direction until $\|\tau\|$ exceeds the $1-$ `alphamax` quantile (see 'Details').
`delta.t`	step size of the profile on the τ scale; `cutoff` is that quantile.
`x`, `object`	an object returned from `profile.onls`.
`absVal`	logical. If `TRUE`, $\|\tau\|$ is plotted, otherwise τ.
`level`	confidence level for the reference lines of the plot and for the intervals.
`parm`	parameters for which intervals are computed, as names or indices.
`...`	further arguments, passed to `plot` by the plot method and ignored otherwise.

Details

For parameter θ_j, the parameter is held fixed at values on both sides of its estimate, and all other free parameters and all foot points are re-optimized by `onls` (using the data, weights and settings stored in `fitted`, so that the profile does not depend on objects in the calling environment). With $\hat{S}$ the minimized objective and $S(\theta_j)$ the constrained minimum, the profile statistic is

$$\tau(\theta_j) = \operatorname{sign}(\theta_j - \hat{\theta}_j)\sqrt{\left(S(\theta_j) - \hat{S}\right)/s^2}$$

with $s^2 = \hat{S}/\mathrm{df}$ when the variances are estimated and $s^2 = 1$ when they are known (`known_sigma = TRUE`). The grid is extended until $|\tau|$ exceeds $\sqrt{F_{1-\alpha}(1, \mathrm{df})}$ (estimated variances) or $\sqrt{\chi^2_{1-\alpha}(1)}$ (known variances), with α = `alphamax`, or a failed refit, a flat profile, a `lower`/`upper` bound (at which the profile is evaluated last) or `maxpts` stops it; the reason is stored in the `stop` element of the `"parameters"` attribute of each profile. The stepping follows `profile.nls`. Models with one free parameter are profiled by refits in which only the foot points are optimized. With estimated variances, an exact fit (no residual degrees of freedom) cannot be profiled and gives an error.

`confint` uses the part of the profile on which τ increases steadily away from the estimate, interpolates it (a monotone spline of τ over the parameter, inverted) at $\tau = \pm q$, with q the normal (known variances) or t quantile; `confint.onls` with `method = "profile"` calls it.

Value

`profile.onls`: a list with one data frame per profiled parameter, with components `tau` and `par.vals` (the parameter values along the profile), of class `c("profile.onls", "profile")`. The attributes `"original.fit"` and `"summary"` hold the fit and its summary.

`confint.profile.onls`: a matrix of lower and upper limits. `plot.profile.onls`: `NULL`, invisibly.

Author(s)

Andrej-Nikolai Spiess

See Also

`confint.onls`.

Examples

```
set.seed(123)
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))
prof <- profile(mod1)
plot(prof)
confint(prof)
```

`residuals_o` *The orthogonal residuals*

Description

Returns a vector of the fitted, precision-weighted orthogonal distances $\hat{d}_i$ from the fitted `onls` model, in the original observation order – the same per-observation quantities that are squared and summed by `deviance_o`. See 'Details' for what $\hat{d}_i$ represents in general; a single-predictor, unweighted fit is a special case where this reduces to plain Euclidean distance.

Usage

```
residuals_o(object)
```

Arguments

`object`	an object returned from `onls`.

Details

$\hat{d}_i$ is the fitted, precision-weighted orthogonal distance of observation i (see 'Details' in `onls` for its full definition and the construction of Qyy_i, Qx_i from `weights`, `sigma_x`, and `sigma_y`); it reduces to plain Euclidean distance only for a single-predictor, unweighted fit. Values are returned in the original observation order regardless of the number of predictors, and any rows omitted via `na.action` are correctly reinserted as `NA` (via `napredict`) at their original positions.

Value

A vector of the fitted orthogonal (precision-weighted) distances.

Author(s)

Andrej-Nikolai Spiess

Examples

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

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

Description

Provides a summary for the parameters of the converged fit, including their standard errors, t values (z values for known variances) and p values, together with residual standard errors with respect to both the vertical and the orthogonal residuals. See 'Details' for how parameters held `fixed` in the original `onls` call, and `known_sigma`-based scaling, are handled.

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 (non-`fixed`) 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.

Details

Fixed parameters. If `object` was fitted with some parameters held `fixed` (see `onls`), those parameters were never estimated: their `Estimate` is still shown (the fixed value used), but `Std. Error`, `t value` and `Pr(>|t|)` are reported as `NA` rather than `0`/`Inf`/a spuriously small p-value, matching R's own convention for non-estimated coefficients (e.g. aliased terms in `summary.lm`). The residual degrees of freedom in `df` count only the *free* (non-fixed) parameters, so that `df[1] + df[2]` equals the number of observations with non-zero weight, as in `summary.lm`/`summary.nls`. If `correlation = TRUE`, fixed parameters are dropped from the returned correlation matrix entirely (rather than padded with `NA` rows/columns), since correlation with a held-fixed constant is not a meaningful quantity.

Known variances. If the variances were treated as known (`known_sigma = TRUE` in `onls`), the standard errors are not estimated from the residuals, and the test statistics are compared with the normal distribution (`z value`, `Pr(>|z|)`); otherwise with the t distribution with `df_resid` degrees of freedom. `confint.onls` and `predict.onls` use the same rule.

cov.unscaled. `object$vcov` was already scaled by `onls` itself: not at all if `known_sigma = TRUE`, or by the *orthogonal*-distance reduced chi-square (`object$reduced_chisq`) if `known_sigma = FALSE`. `cov.unscaled` undoes exactly that scaling (dividing by `object$reduced_chisq` only in the latter case), so that `vcov(object) == cov.unscaled * sigma^2` holds with the appropriate `sigma`, matching the `summary.lm`/`summary.nls` convention. This is *not* the same quantity as the *vertical*-residual variance (`sigmaONLS^2`): the vertical and orthogonal residual sums of squares generally differ, even for an unweighted fit, so `cov.unscaled` must not be confused with, or derived from, `sigmaONLS`.

Value

A list of class "summary.onls", with components:

formula	the model formula.
residONLS	the vertical residuals (observed minus fitted response at the observed predictors), as residuals.
sigmaONLS	residual standard error of the *vertical* distances.
sigma_o	residual standard error of the *orthogonal* (precision-weighted) distances.
df	a length-2 vector, the number of free (non-fixed) parameters and the residual degrees of freedom; sum(df) equals the number of observations with non-zero weight.
cov.unscaled	the unscaled covariance matrix of the free parameters; see 'Details'.
call	the matched call.
convInfo	convergence information from onls.
control	the control settings used, see onls.
na.action	information on the handling of NAs.
coefficients, parameters	(identical) matrices of Estimate, Std. Error, t value and Pr(>|t|) (z value and Pr(>|z|) if the variances are known), one row per parameter (including fixed ones, with NA in the last three columns for those); see 'Details'.
known_sigma	as in object.
reduced_chisq	as in object.
correlation	(only if correlation = TRUE) the correlation matrix of the *free* parameters only; see 'Details'.
symbolic.cor	(only if correlation = TRUE) as supplied.

Author(s)

Andrej-Nikolai Spiess

Examples

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

---

x0, y0 — *x0/y0-values from orthogonal nonlinear least squares regression*

Description

Returns the fitted foot points $\hat{\xi}_i$ (x0) and the corresponding fitted response $f(\hat{\xi}_i, \hat{\theta})$ (y0) obtained by onls – the points on the fitted curve/surface that the orthogonal-distance criterion projects each observation onto. See 'Details' for the criterion actually minimized, which reduces to plain Euclidean distance only in the simplest (single-predictor, unweighted) case; for multivariate and/or weighted fits, a precision-weighted quadratic form is minimized instead.

Usage

```
x0(object)
y0(object)
```

Arguments

object	an object returned from onls.

Details

For a single-predictor ($p = 1$), unweighted onls fit, the foot point $\hat{\xi}_i$ (returned by x0) is obtained by minimizing the plain Euclidean distance between the observation (x_i, y_i) and a point $(\xi_i, f(\xi_i, \hat{\theta}))$ on the fitted curve,

$$\min_{\xi_i} \sqrt{(x_i - \xi_i)^2 + \left[y_i - f(\xi_i, \hat{\theta})\right]^2}$$

More generally – for multivariate models ($p > 1$) and/or fits using weights, sigma_x, or sigma_y – onls instead minimizes a precision-weighted quadratic form (see 'Details' in onls for the full construction),

$$\min_{\xi_i} \; Qyy_i \left[y_i - f(\xi_i, \hat{\theta})\right]^2 + (\xi_i - x_i)^T Qx_i (\xi_i - x_i)$$

which reduces to the plain Euclidean distance above only when $p = 1$ and $Qyy_i = Qx_i = 1$ (the default, unweighted case) – Euclidean distance is therefore a special case of the criterion minimized, not the general rule.

x0 returns $\hat{\xi}_i$: a length-n vector for single-predictor models, or an $n \times p$ matrix (one column per predictor, in the order of the predictors in the formula, as in object$X_obs) for multivariate models. y0 returns $f(\hat{\xi}_i, \hat{\theta})$, always a length-$n$ vector, since the response is univariate regardless of p.

Values are returned in the original observation order (after subset), with NA re-inserted at the positions of observations removed by na.action = na.exclude, so that they line up with the rows of the data (they are taken from the xi and yi components of the fit). Before version 0.2-1, single-predictor models returned them sorted by the predictor.

Value

For single-predictor models, a length-n vector of $\hat{\xi}_i$ (x0) or $f(\hat{\xi}_i, \hat{\theta})$ (y0) values. For multivariate models, x0 returns an $n \times p$ matrix (one column per predictor); y0 still returns a length-n vector, since the response remains univariate.

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(mod)
```

Index

∗ **htest**
anova.onls, 2
∗ **models**
anova.onls, 2
check_o, 3
confint.onls, 5
deviance_o, 7
doGauge, 8
logLik_o, 11
onls, 13
plot.onls, 22
predict.onls, 25
print.onls, 27
profile.onls, 28
residuals_o, 30
summary.onls, 31
x0, y0, 32
∗ **nonlinear**
anova.onls, 2
check_o, 3
confint.onls, 5
deviance_o, 7
doGauge, 8
logLik_o, 11
onls, 13
plot.onls, 22
predict.onls, 25
print.onls, 27
profile.onls, 28
residuals_o, 30
summary.onls, 31
x0, y0, 32
∗ **optimize**
check_o, 3
deviance_o, 7
logLik_o, 11
onls, 13
residuals_o, 30
x0, y0, 32

AIC, *12*
anova.onls, 2, *19*

BIC, *12*

check_o, 3, *10*, *17*, *18*, *20*, *23*, *24*, *27*
confint.onls, *3*, 5, *17*, *20*, *26*, *27*, *29*, *31*
confint.profile.onls *(*profile.onls*)*, 28

deriv, *17*
deviance, *19*
deviance_o, 7, *19*, *30*
doGauge, 8, *16*

extendrange, *23*

fitted, *19*, *20*
formula, *13*

logLik, *11*, *12*, *19*
logLik_o, *2*, *3*, 11, *19*

napredict, *30*
nls, *11*, *13*, *16*, *22*
nls.lm, *15–17*
nls.lm.control, *14*
nlsLM, *16*

onls, *2–4*, *6–11*, 13, *22–33*

plot, *23*, *28*
plot.onls, 22
plot.profile.onls *(*profile.onls*)*, 28
predict, *20*
predict.lm, *26*
predict.onls, 25, *31*
print.onls, *18*, 27
profile.nls, *29*
profile.onls, *3*, *6*, *7*, *14*, *20*, 28

residuals, *18–20*, *32*

residuals_o, *11*, *15*, *19, 20*, 30

summary.lm, *31*
summary.nls, *31*
summary.onls, *26, 27*, 31

vcov, *6, 7*, *26*

x0, *19, 20*
x0 *(*x0, y0*)*, 32
x0, y0, 32

y0, *20*
y0 *(*x0, y0*)*, 32