Package: onls (via r-universe)

October 21, 2026

Type Package

LazyLoad no

Title Orthogonal Nonlinear Least-Squares Regression

Version 0.2

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

Description Fits n-dimensional data by means of orthogonal nonlinear least-squares using Levenberg-Marquardt minimization 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 (>= 2.13.0), minpack.lm, rgl

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Andrej-Nikolai Spiess [aut, cre]

Config/pak/sysreqs cmake libfreetype6-dev libglu1-mesa-dev make texlive libpng-dev libuv1-dev libgl1-mesa-dev zlib1g-dev

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

Date/Publication 2026-09-21 05:10:02 UTC

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

RemoteRef HEAD

RemoteSha 8a789cee953de515405723547b13eae62a119fb0

Contents

check_o 2
confint.onls 5
deviance_o 6

logLik_o 7
NIST 9
onls 10
plot.onls 21
print.onls 24
residuals_o 25
summary.onls 26
x0, y0 28

Index **30**

`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. For unweighted (isotropic) fits, orthogonality is checked via the angle between the tangent to the model at (x_{0i}, y_{0i}) and the Euclidean vector from the foot point to (x_i, y_i), which should be $90°$. For weighted/heteroscedastic fits (see `sigma_x`, `sigma_y`, `weights` in `onls`), a plain right angle is no longer the correct criterion, so `check_o` instead checks the underlying first-order KKT stationarity condition directly. Multivariate models (more than one predictor) are checked one predictor axis at a time. See 'Details'.

Usage

```
check_o(object, plot = TRUE, tol_deg = 0.05, tol_kkt = 0.001)
```

Arguments

`object`	an object returned from `onls`.
`plot`	logical. If `TRUE`, the orthogonality diagnostic (α-values for unweighted fits, or relative KKT residuals for weighted fits) is plotted for a quick overview of all points. For multivariate models, one panel per predictor is drawn.
`tol_deg`	tolerance in degrees for the unweighted criterion: 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`	tolerance for the weighted criterion: a point is orthogonal if its relative KKT residual is smaller than `tol_kkt`. Default `0.001`.

Details

Which criterion is used. `check_o` automatically chooses between two criteria, based on whether the fitted `onls` model is effectively weighted, i.e. if *any* of the following holds:

- `object$known_sigma` is `TRUE`,

- `object$sigma_x` was supplied to `onls`,
- `object$weights` has more than one distinct value.

Otherwise (default `onls` call, unit `sigma_y`, no `sigma_x`, constant or absent `weights`) the fit is treated as unweighted/isotropic. Note that explicitly passing `sigma_y` to `onls` (even `sigma_y = 1`) sets `known_sigma = TRUE` and therefore selects the weighted (KKT) criterion.

Background. `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 the stationarity condition

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

which both criteria below test. For unit precisions it is exactly the statement that the vector from the foot point to the observation is orthogonal to the model surface.

Unweighted case: tangent-angle criterion.
Let $dx_i = x_i - x_{0i}$, $dy_i = y_i - y_{0i}$ be the residual vector from the foot point to the observation, and $m_i = df(x, \hat{\theta})/dx$ evaluated at $x = x_{0i}$ the slope of the tangent, whose direction vector is $(1, m_i)$. The function calculates the angle α_i between the residual vector and the tangent,

$$\alpha_i[^\circ] = \operatorname{atan2}\left(|m_i\,dx_i - dy_i|\,,\ |dx_i + m_i\,dy_i|\right) \cdot \frac{180}{\pi},$$

which lies in $[0^\circ, 90^\circ]$ and equals 90° exactly when $dx_i + m_i\,dy_i = 0$, the stationarity condition above with unit precisions. This is algebraically identical to the classical expression $\tan(\alpha_i) = |(m_i - n_i)/(1 + m_i n_i)|$ with the slope $n_i = dy_i/dx_i$ of the residual vector, but avoids the division by dx_i, so that points whose foot point coincides with the observation ($dx_i = 0$, for example where the model slope is zero, or observations at the boundary of a flat model) are evaluated correctly instead of returning `NaN`. A point is flagged orthogonal when $|\alpha_i - 90^\circ| <$ `tol_deg`. If the residual vector has (numerically) zero length, $\sqrt{dx_i^2 + dy_i^2} \le \sqrt{\epsilon_{mach}}\,(|x_i| + |y_i|)$, the observation lies on the fitted curve, the angle is undefined, and the point is reported with `alpha = NA` and `Ortho = TRUE`. This criterion is the appropriate one only when the underlying objective is the plain (unweighted) Euclidean distance, i.e. $Qyy_i = Qx_i = 1$ in the notation of `onls`.

Weighted case: KKT-residual criterion.
Once Qyy_i and/or Qx_i rescale the response/predictor axes anisotropically (see `onls`), a plain right angle is no longer the correct orthogonality condition. Instead `check_o` checks directly the first-order stationarity condition that the foot point x_{0i} satisfies at the `onls` optimum,

$$Qyy_i\,(y_i - y_{0i})\,m_i = Qx_i\,(x_{0i} - x_i),$$

by computing the relative residual of this equality,

$$\text{rel_resid}_i = \frac{|\,Qyy_i(y_i - y_{0i})m_i\ -\ Qx_i(x_{0i} - x_i)\,|}{\max\left(|Qyy_i(y_i - y_{0i})m_i|\,,\ |Qx_i(x_{0i} - x_i)|\,,\ \epsilon_{mach}\right)},$$

where ϵ_{mach} is `.Machine$double.eps` (used only to avoid division by zero when both terms vanish). A point is flagged orthogonal when rel_resid$_i <$ `tol_kkt`. Qyy_i is taken from `object$Q_yy`; Qx_i is taken from `object$Q_x` (its diagonal, or observation-specific row, as applicable, see `onls`). For single-predictor models, Qyy_i/Qx_i (stored in the original observation order) are internally re-aligned to match the sorted-predictor order used elsewhere in the returned data frame (see 'Value'), so they are correctly paired with the corresponding observation.

Multivariate models ($p > 1$).
One check (angle-based or KKT-residual-based, per the rule above) is performed *per predictor axis* k, using the partial derivative $\partial f/\partial x_k$ at the foot point in place of m_i, and (for the weighted case) the k-th diagonal precision element in place of Qx_i. For a global, correlated (non-diagonal) `sigma_x` covariance matrix, only the diagonal entries of the precision matrix are used for this per-axis check; the true joint stationarity condition couples all predictor axes simultaneously, so this diagnostic is an approximation in that case. The overall `Ortho` flag for an observation is `TRUE` only if every predictor axis passes its individual check.

Slopes. The slopes m_i (or partial derivatives) are computed inside `check_o` 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 α_i is highly sensitive to the exact position of the foot point (an error in the foot point of relative size ε produces an angle error of roughly ε divided by the length of the residual vector), 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 columns depending on whether the model has a single predictor or several, and on whether the fit is unweighted or weighted (see 'Details' for when each regime applies): For single-predictor models ($p = 1$): the observed predictor x_i, the foot point x_{0i}, the observed response y_i, the fitted foot-point response y_{0i}, either `alpha` (unweighted; α_i in degrees, `NA` for an observation that lies on the fitted curve) or `rel_resid` (weighted; the relative KKT residual), the model slope `df/dx` at the foot point, and a logical `Ortho` that is `TRUE` when $|\alpha_i - 90°| <$ `tol_deg` (unweighted; also `TRUE` if `alpha` is `NA`) or when the relative KKT residual is $<$ `tol_kkt` (weighted). For multivariate models ($p > 1$): the observed predictors $x_{i1}, \ldots, x_{ip}$, the foot-point predictors `x0_<name>` for each predictor, the observed response, the fitted foot-point response `y0`, either `alpha_<name>` or `rel_resid_<name>` for each predictor axis, the model partial derivative `df/dx_<name>` at the foot point for each predictor axis, and an overall logical `Ortho` that is `TRUE` only if the per-axis criterion passes on *every* predictor axis. If `plot = TRUE`, the diagnostic is additionally plotted, in black where `Ortho = TRUE` and dark red otherwise.

For single-predictor models, rows are returned in the internally-used sorted-predictor order (matching `object$pred`/`object$resp`/`x0`/`y0`), NOT the original row order of the input data; for multivariate models, sorting is a no-op and the original observation order is used.

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` *Bootstrapped confidence intervals for 'onls' model parameters*

Description

Computes bootstrap confidence intervals for all parameters of an `onls` model. Confidence limits are obtained from the empirical distribution of parameter estimates generated by nonparametric case-resampling. Unlike `confint.nls`, which uses profile likelihoods, `confint.onls` is based on repeated refitting of the orthogonal nonlinear least-squares model to bootstrap samples and therefore remains fully consistent with the estimation criterion used by `onls`.

Usage

```
## S3 method for class 'onls'
confint(object, parm, level = 0.95, k = 200, ...)
```

Arguments

`object`	an object returned from `onls`.
`parm`	just for S3 purposes, as all parameters are bootstrapped.
`level`	the confidence level required.
`k`	the number of bootstrap samples.
`...`	additional argument(s) passed to `update.onls`.

Details

Confidence intervals are obtained by nonparametric bootstrap resampling.
Let $\hat{\theta}$ denote the vector of parameter estimates from the original fit. For each bootstrap replicate,

1. rows of the original data set are sampled with replacement to generate a bootstrap sample of the same size as the original data
2. the model is refitted to the bootstrap sample using `onls`
3. parameter estimates from successful fits are stored
4. After k successful bootstrap fits have been collected, the confidence interval for each parameter is obtained from the empirical quantiles of its bootstrap distribution

Fits that fail to converge or violate internal orthogonality checks are discarded. Additionally, bootstrap estimates whose absolute deviation from the original estimate exceeds twenty times the corresponding estimated standard error are treated as pathological solutions and excluded.

Value

A matrix (or vector) with columns giving lower and upper confidence limits for each parameter.

Author(s)

Andrej-Nikolai Spiess

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, k = 100) # 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'.
```

`logLik_o` *Log-Likelihood for the orthogonal residuals*

Description

Returns the log-likelihood as calculated from the orthogonal residuals obtained by `residuals_o`, including a Jacobian/normalizing-constant correction for the response and predictor precisions (`weights`, `sigma_x`, `sigma_y`) used by `onls`, so that the resulting value (and hence `AIC`/`BIC`) is comparable across `onls` fits made with different weighting schemes.

Usage

```
logLik_o(object)
```

Arguments

`object`	an object returned from `onls`.

Details

Let d_i be the weighted orthogonal distance of observation i at the fitted parameters (`residuals_o`, i.e. `object$resid_o`), and N the number of observations. A naive treatment of the d_i as i.i.d. $N(0, \sigma^2)$ draws gives the usual concentrated Gaussian log-likelihood

$$\ell_0 = -\frac{N}{2}\left(\log(2\pi) + 1 - \log(N) + \log\left(\sum_i d_i^2\right)\right),$$

which is what earlier versions of `logLik_o` returned. However, since `onls` already folds the response precision Qyy_i and predictor precision Qx_i into d_i itself (see 'Details' in `onls`), ℓ_0 alone omits the normalizing-constant (Jacobian) term that these precisions contribute to the underlying Gaussian density – exactly as plain `logLik`/`AIC` on *unweighted* residuals from a *weighted* `lm`/`nls` fit would, if computed without R's own $+\frac{1}{2}\sum_i \log(w_i)$ correction. `logLik_o` therefore adds the corresponding correction,

$$\ell = \ell_0 \; + \; \frac{1}{2}\sum_{i=1}^{N} \log(Qyy_i) \; + \; \frac{1}{2}\sum_{i=1}^{N} \log|Qx_i|,$$

where $|Qx_i|$ is the determinant of the (possibly per-observation) predictor precision matrix used by `onls` – a scalar for single-predictor models, and the full $p \times p$ determinant (correctly accounting for any predictor correlation) for multivariate models. For the default, fully unweighted `onls` fit, $Qyy_i = 1$ and $Qx_i = I_p$ for every i, so both correction terms are exactly 0 and $\ell = \ell_0$; the correction only changes the value for fits using `weights`, `sigma_x`, and/or `sigma_y`.

Note that d_i itself remains a simplification: it is the square root of a sum of $p+1$ squared, precision-weighted Gaussian terms (one from the response, p from the predictors), not a single univariate normal draw, so treating $\sum_i d_i^2/N$ as the MLE of a common residual variance (as ℓ_0 does) is itself an approximation, inherited unchanged from the original (unweighted) formula. The correction above addresses only the missing weighting/precision Jacobian term, not this deeper simplification.

Value

The log-likelihood as calculated from the orthogonal residuals, with attributes `"df"` ($1 + q_{free}$, the number of free model parameters plus one for the residual scale) and `"nobs"`/`"nall"` (the number of observations).

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. Because `logLik_o` includes the response/predictor precision correction described in 'Details', its value (and `AIC`/`BIC` derived from it) can be validly compared between two `onls` fits on the same data even if they used different `weights`, `sigma_x`, or `sigma_y`, provided the response variable and the number of observations are the same.

Author(s)

Andrej-Nikolai Spiess

Examples

```
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)

## compare AIC of vertical versus orthogonal residuals.
AIC(mod1)
AIC(logLik_o(mod1))

## Comparing an unweighted and a weighted onls() fit on the same data:
## logLik_o() includes the precision Jacobian correction, so AIC/BIC
## from the two fits are validly comparable.
mod2 <- 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)
AIC(logLik_o(mod1))
AIC(logLik_o(mod2))
```

NIST *Complicated nonlinear models from the NIST database*

Description

Fits, by onls, a few complicated nonlinear models from the NIST database (https://www.itl.nist.gov/div898/strd/nls/nls_main.shtml), *i.e.* 'Chwirut2', 'Gauss1', 'ENSO' and 'Rat43'.

Usage

```
NIST(verbose = FALSE)
```

Arguments

verbose	logical. If TRUE, the onls output is printed to the console.

Value

Output of the models

Author(s)

Andrej-Nikolai Spiess

Examples

```
NIST()
```

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, Rogers and Schnabel; 1987): 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)
```

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. 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.

`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`, spent in restarts of at most 1024 iterations each, see 'Details'); `ftol` and `ptol`, the relative-reduction and relative-step convergence tolerances passed to `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. At least one parameter must be free.

`subset` Optional specification of a subset of 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 (including restarts) are printed.

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

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

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` and `resid_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`). See example 12.

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 (MorÃ© 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 scaling D. 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. ODRPACK exploits this structure to make each step cheap; `onls` delegates the step to the general dense solver of **minpack.lm**, so the cost per iteration grows faster with n than in ODRPACK.
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) *Restarts and convergence.* **minpack.lm** caps a single `nls.lm` call at 1024 iterations. The solver is restarted from its last iterate (which also resets the trust region) until it converges (`nls.lm` codes 1 to 4, giving `convergence = 0`), a restart no longer reduces the objective, or the budget is exhausted (`convergence = 1`, with a warning). 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 an angle or, for weighted fits, through the relative residual of this equation).

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. `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`.

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 $\hat{\sigma}^2 = \left(\sum_i d_i^2\right)/\mathrm{df_resid}$ 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 (`residONLS`) with those of the NLS fit (`residNLS`), 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, predictor corrections 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 $y = b_0 + b_1 x1 + b_2 x2$ or $y = b_0 + b_1 x1 + b_2 x2 + b_3 x1 x2$, respectively.

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`	Original `data` argument.
`call`	Matched function call.
`convInfo`	Convergence information from the joint optimization: `isConv` (logical), `finIter` (total number of Levenberg-Marquardt iterations over all restarts), `finTol` (the `ftol` used), `stopCode` and `stopMessage` (termination code and message of the last `nls.lm` call).
`na.action`	Information on omitted observations.
`dataClasses`	Classes of model variables.
`model`	Model frame used for fitting.
`formula`	Model formula.
`parNLS`	Parameter estimates from the initial nonlinear least-squares fit (the starting values, if the warm start failed; fixed parameters at their starting values).
`parONLS`	Final orthogonal-distance parameter estimates.
`x0`, `y0`	Coordinates $\hat{\xi}_i, f(\hat{\xi}_i, \hat{\theta})$ of the fitted foot points on the model curve/surface, in sorted predictor order (see 'Details'; for $p > 1$, sorting is a no-op and the original order is used).

`fittedONLS`	Model evaluated at the *observed* predictors, $f(x_i, \hat{\theta})$, using the final ODR parameter estimates.
`fittedNLS`	Model evaluated at the observed predictors using the initial NLS warm-start estimates.
`residONLS`	Vertical residuals $y_i - f(x_i, \hat{\theta})$ from the orthogonal model (observed x, not the foot point).
`residNLS`	Vertical residuals from the initial nonlinear least-squares fit.
`resid_o`	Weighted orthogonal distances d_i (see 'Details') at the final estimates, in sorted predictor order. Numerically equivalent to `orth_dist`, but reordered; compare to `orth_dist` for the original observation order.
`pred`, `resp`	Predictor and response values used internally during fitting (sorted order for $p = 1$).
`grad`	Jacobian $\partial f(\hat{\xi}_i, \theta)/\partial\theta$ of the model with respect to *all* model parameters (fixed and free), evaluated at the final foot points $\hat{\xi}_i$ and estimates $\hat{\theta}$ (not at the observed predictors), in the original observation order. Only its free-parameter columns are used to construct `vcov`.
`QR`	QR decomposition of `grad`.
`weights`	Observation weights used in fitting, in original observation order.
`control`	The resolved control settings actually passed to `nls.lm` (via `nls.lm.control`) for each restart, not the raw `control` argument.
`coefficients`	Named vector of final parameter estimates.
`std_errors`	Approximate standard errors of parameter estimates (`0` for `fixed` parameters); see 'Details' for the covariance used.
`vcov`	Estimated parameter covariance matrix; see 'Details'.
`xi`	Estimated foot-point coordinates $\hat{\xi}_i$ in the original observation order.
`x_corrections`	Estimated predictor corrections $\hat{\delta}_i = \hat{\xi}_i - x_i$.
`orth_dist`	Final weighted orthogonal distances d_i, in original observation order (see `resid_o` for the sorted-order equivalent).
`objective`	Minimized objective value $S(\hat{\theta}, \hat{\delta}) = \sum_i d_i^2$.
`Q_x`	Predictor precision structure used in fitting.
`Sigma_x`	Predictor covariance structure corresponding to `Q_x`.
`Q_yy`	Response precision vector.
`is_diag_x`	Logical indicating whether predictor precision is diagonal.
`per_obs_x`	Logical indicating whether predictor precision varies by observation.
`sigma_x`	User-supplied predictor error specification (after `subset`/`na.action`).
`sigma_y`	User-supplied response error specification (after `subset`/`na.action`).
`known_sigma`	Logical indicating whether measurement variances were treated as known.
`reduced_chisq`	Reduced chi-square statistic based on the orthogonal-distance criterion.
`pred_names`	Predictor variable names.
`param_names`	Model parameter names.

`start`	Starting parameter values.
`fixed`	Logical vector identifying fixed parameters.
`lower`, `upper`	Parameter bounds supplied to the optimizer.
`y_obs`, `X_obs`	Observed response and predictor values.
`resp_name`	Response variable name.
`n`, `p`, `q`	Numbers of observations, predictors and model parameters.
`df_resid`	Residual degrees of freedom, n minus the number of *free* (non-`fixed`) parameters.
`convergence`	Convergence code, `0` if the joint optimization converged (`nls.lm` termination codes 1 to 4), `1` otherwise.
`iters`	Total number of Levenberg-Marquardt iterations, summed over all restarts (`NA` if the joint optimization failed to run).
`type`	Model type, currently `"nonlinear"`.
`ortho`	Orthogonality diagnostics returned by `check_o` (`NA` if they could not be computed).

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)


## Reference tests comparing to pivotal literature
## 4. 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)
mod4 <- onls(y ~ b1 + b2 * (exp(b3 * x) -1)^2, data = DAT,
             start = list(b1 = 1500, b2 = -50, b3 = -0.1))
deviance_o(mod4) # 21.445 as on page 47
summary(mod4) # 1264.65481 (1.03492) / -54.01838 (1.583992) / -0.08785 (6.33222E-3) as on page 48

## 5. 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)
mod5 <- onls(y ~ b1 * 10^(b2 * x/(b3 + x)), data = DAT,
             start = list(b1 = 1, b2 = 5, b3 = 100))
deviance_o(mod5) # 15.263 as on page 363
summary(mod5) # 4.4879 (0.56876) / 7.1882 (0.69504) / 221.8383 (37.2313) as on page 363

## 6. Example with bounds from simple_example.f90
## in https://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)
mod6 <- 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(mod6) # 1.4376 / 0.9 ## Different to reference 1.6334 / 0.9
deviance_o(mod6) # 0.1919 => lower RSS than original ODRPACK with 0.2674!

## 7. Example similar to Deming regression
## Comparison to XLstat
## https://help.xlstat.com/6650-run-deming-regression-compare-methods-excel
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)

## 8. Linear multivariate model, using the closed-form Total Least Squares
## (TLS) solution from Golub & Van Loan (1980)
tls_fit <- function(X, y) {
  X <- as.matrix(X)
  n <- nrow(X); p <- ncol(X)
  Xc <- scale(X, center = TRUE, scale = FALSE)
  yc <- y - mean(y)
  xbar <- colMeans(X); ybar <- mean(y)
  Z <- cbind(Xc, yc)
  SVD <- svd(Z)
  v <- SVD$v[, p + 1L]
  v_x <- v[1:p]; v_y <- v[p + 1L]
  slope <- -v_x / v_y; intercept <- ybar - sum(slope * xbar)
    list(intercept = intercept, slope = setNames(slope, colnames(X)),
       singular_values = SVD$d)
}

set.seed(11)
n <- 40
x1_true <- runif(n, 0, 10)
x2_true <- runif(n, 0, 10)
b0_true <- 3; b1_true <- 1.5; b2_true <- -0.8
y_true  <- b0_true + b1_true * x1_true + b2_true * x2_true
x1 <- x1_true + rnorm(n, 0, 0.5)
x2 <- x2_true + rnorm(n, 0, 0.5)
y  <- y_true  + rnorm(n, 0, 0.5)
DAT <- data.frame(x1 = x1, x2 = x2, y = y)
TLS <- tls_fit(DAT[, c("x1", "x2")], DAT$y)
mod8 <- onls(y ~ b0 + b1 * x1 + b2 * x2, data = DAT, start = list(b0 = 1, b1 = 1, b2 = 1))
TLS_vec  <- c(b0 = TLS$intercept, b1 = TLS$slope[["x1"]], b2 = TLS$slope[["x2"]])
ONLS_vec <- coef(mod8)[c("b0", "b1", "b2")]
print(data.frame(TLS_closed_form = TLS_vec, onls = ONLS_vec,
      abs_diff = abs(TLS_vec - ONLS_vec))) # all equal

## 9. Pearson (1901) / York (1966) -> "Pearson's data with York's weights"
#   Intercept: 5.47991  (SE 0.29497)
#   Slope:    -0.48053  (SE 0.05799)
x <- c(0.0, 0.9, 1.8, 2.6, 3.3, 4.4, 5.2, 6.1, 6.5, 7.4)
y <- c(5.9, 5.4, 4.4, 4.6, 3.5, 3.7, 2.8, 2.8, 2.4, 1.5)
sd_x <- 1/sqrt(c(1000.0, 1000.0, 500.0, 800.0, 200.0, 80.0, 60.0, 20.0, 1.8, 1.0))
sd_y <- 1/sqrt(c(1.0, 1.8, 4.0, 8.0, 20.0, 20.0, 70.0, 70.0, 100.0, 500.0))
DAT <- data.frame(x = x, y = y)
mod9 <- onls(y ~ b0 + b1*x, data = DAT,
                    start = list(b0 = 5, b1 = -0.5),
                    sigma_x = sd_x, sigma_y = sd_y)
summary(mod9) # 5.47991 (0.29497) / -0.48053 (0.05799) as in paper
```

```
## 10. Daeron & Vermeesch (2024), Table 3 / Figure 2C toy example.
x <- c(9, 19, 31, 41)
y <- c(21, 31, 39, 49)
DAT <- data.frame(x = x, y = y)
mod10 <- onls(y ~ a + b * x, data = DAT, start = list(a = 10, b = 1), sigma_x = 1, sigma_y = 1)
summary(mod10) # 13.71 / 0.851 as in Table 3 of paper

## 11. Full predictor covariance (correlated predictor errors), compared to the closed-form
## generalized Total Least Squares (TLS) solution. Whitening the predictors with the Cholesky
## factor L of the precision matrix (t(L) %*% L = solve(Sigma)) and scaling y by 1/sigma_y
## turns the problem into plain TLS, which is solved by an SVD.
set.seed(123)
n <- 40
Sigma <- matrix(c(0.25, 0.15, 0.15, 0.16), 2)  # correlation of predictor errors = 0.75
sigma_y <- 0.3
xt <- cbind(runif(n, 0, 10), runif(n, 0, 10))
E <- matrix(rnorm(2 * n), n) %*% chol(Sigma)
DAT <- data.frame(x1 = xt[, 1] + E[, 1], x2 = xt[, 2] + E[, 2],
                  y = 3 + 1.5 * xt[, 1] - 0.8 * xt[, 2] + rnorm(n, 0, sigma_y))
mod12 <- onls(y ~ b0 + b1 * x1 + b2 * x2, data = DAT,
              start = list(b0 = 1, b1 = 1, b2 = 1), sigma_x = Sigma, sigma_y = sigma_y,
              control = list(ftol = 1e-13, ptol = 1e-13))
L <- chol(solve(Sigma))
U <- as.matrix(DAT[, c("x1", "x2")]) %*% t(L) # whitened predictors
v <- DAT$y / sigma_y
Z <- cbind(scale(U, scale = FALSE), v - mean(v))
V <- svd(Z)$v[, 3]
w <- -V[1:2] / V[3]
gTLS <- c(sigma_y * (mean(v) - sum(w * colMeans(U))), sigma_y * drop(t(L) %*% w))
print(data.frame(gen_TLS = gTLS, onls = coef(mod12),
                 abs_diff = abs(gTLS - coef(mod12)))) # all equal

## 12. ODRPACK's separate weights WE (response) and WD (predictor).
## ODRPACK takes both as precisions (inverse variances), per observation. In onls(), WE is
## passed as 'weights' (with the default sigma_y = 1) and WD through sigma_x = 1/sqrt(WD)
## (for p > 1: an n x p matrix 1/sqrt(WD)). known_sigma = FALSE gives ODRPACK's scaling of
## the standard errors by the residual variance.
set.seed(123)
n <- 30
xt <- seq(0.5, 10, length.out = n)
WD <- runif(n, 0.5, 4)   # predictor weights
WE <- runif(n, 0.5, 4)   # response weights
x <- xt + rnorm(n, 0, 0.3/sqrt(WD))
y <- 2 * exp(-0.3 * xt) + 0.5 + rnorm(n, 0, 0.03/sqrt(WE))
DAT <- data.frame(x, y)
mod12 <- onls(y ~ a * exp(-b * x) + c, data = DAT, start = list(a = 1.5, b = 0.2, c = 0.3),
              weights = WE, sigma_x = 1/sqrt(WD), known_sigma = FALSE)
summary(mod12)
```

`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 `asp`.

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 with x1/x2/z setup, and where the segments connect the estmated 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, per-axis 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 10% added noise.
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
plot(mod2)    # should render as 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)
```

`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, and the number of iterations to convergence.

Author(s)

Andrej-Nikolai Spiess

Examples

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

---

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 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, 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.

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).
`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.
`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 `NA`s.
`coefficients`, `parameters`	(identical) matrices of `Estimate`, `Std. Error`, `t value` and `Pr(>|t|)`, 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 given by `object$pred_names`) for multivariate models. `y0` returns $f(\hat{\xi}_i, \hat{\theta})$, always a length-$n$ vector, since the response is univariate regardless of p.

For single-predictor models, values are returned in the internally-used sorted-predictor order (matching `object$pred`/`object$resp`), not necessarily the original row order of the input data; for multivariate models, sorting is a no-op and the original observation order is used.

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

∗ **models**
  check_o, 2
  confint.onls, 5
  deviance_o, 6
  logLik_o, 7
  NIST, 9
  onls, 10
  plot.onls, 22
  print.onls, 24
  residuals_o, 25
  summary.onls, 26
  x0, y0, 28
∗ **nonlinear**
  check_o, 2
  confint.onls, 5
  deviance_o, 6
  logLik_o, 7
  NIST, 9
  onls, 10
  plot.onls, 22
  print.onls, 24
  residuals_o, 25
  summary.onls, 26
  x0, y0, 28
∗ **optimize**
  check_o, 2
  deviance_o, 6
  logLik_o, 7
  NIST, 9
  onls, 10
  residuals_o, 25
  x0, y0, 28

AIC, *7*, *8*

BIC, *7*

check_o, 2, *14*, *15*, *17*, *22*, *23*, *25*
confint.nls, *5*
confint.onls, 5

deriv, *13*
deviance, *15*
deviance_o, 6, *15*, *25*

extendrange, *22*

fitted, *15*
formula, *10*

lm, *8*
logLik, *8*, *15*
logLik_o, 7, *15*

napredict, *26*
NIST, 9
nls, *8*, *12*, *22*
nls.lm, *13*, *15*, *16*
nls.lm.control, *11*, *16*
nlsLM, *13*

onls, *2–9*, 10, *22–28*

plot, *22*
plot.onls, 21
print.onls, *15*, 24

residuals, *15*
residuals_o, *7*, *8*, *15*, 25

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

x0 *(x0, y0)*, 28
x0, y0, 28

y0 *(x0, y0)*, 28