Orthogonal Nonlinear Least-Squares Regression in R

Andrej-Nikolai Spiess
Soilytix GmbH, Hamburg, Germany
draspiess@gmail.com

November 24, 2024

Abstract

Orthogonal nonlinear least squares (ONLS) regression is a not so frequently applied and largely overlooked regression technique that comes into question when one encounters an "error in variables" problem. While classical nonlinear least squares (NLS) aims to minimize the sum of squared vertical residuals, ONLS minimizes the sum of squared orthogonal residuals. The method is based on finding points on the fitted line that are orthogonal to the data by minimizing for each (x_i, y_i) the Euclidean distance $\|D_i\|$ to some point (x_{0i}, y_{0i}) on the fitted curve. There is a 25 year old FORTRAN implementation for ONLS available (ODRPACK, http://www.netlib.org/toms/869.zip), which has been included in the 'scipy' package for Python (http://docs.scipy.org/doc/scipy-0.14.0/reference/odr.html). Here, `onls` has been developed for easy future algorithm tweaking in R. The results obtained from `onls` are exactly similar to those found in [1, 4]. The implementation is based on an inner loop using `optimize` for each (x_i, y_i) to find $\min \|D_i\|$ within some border $[x_{i-w}, x_{i+w}]$ and an outer loop for the fit parameters using `nls.lm` of the 'minpack' package.

Overview

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

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

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

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

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

$$\min \|D_i\| \equiv \min \sqrt{(x_i - x_{0i})^2 + (y_i - y_{0i})^2} \tag{3}$$

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

$$\underset{x_{0i} \in [a,b]}{\operatorname{argmin}} \sqrt{(x_i - x_{0i})^2 + (y_i - f(x_{0i}, \hat{\boldsymbol{\beta}}))^2} \tag{4}$$

Algorithm and Implementation

In detail, `onls` conducts the following steps:

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

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

How to use

1. Building the model

As in the 'Examples' section of `nls` (here with 10% added error), we supply a formula, data environment and starting parameters to the model:

```
> library(onls)
> 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))
```

```
Obtaining starting parameters from ordinary NLS...
  Passed...
 Relative error in the sum of squares is at most `ftol'.
Optimizing orthogonal NLS...
  Passed... Relative error in the sum of squares is at most `ftol'.
```

2. Looking at the model and checking orthogonality

`print`ing the model will give us the estimated coefficients, the (classical) vertical residual sum-of-squares, the orthogonal residual sum-of-squares, and **most importantly**, information on how many points (x_i, y_i) are orthogonal to (x_{0i}, y_{0i}) on the fitted curve $f(x_i + \delta_i, \boldsymbol{\beta}) + \varepsilon_i$. This is accomplished using the independent checking routine `check_o` which calculates the angle between the slope m_i of the tangent obtained from the first derivative at (x_{0i}, y_{0i}) and the slope n_i of the `onls`-minimized Euclidean distance between (x_{0i}, y_{0i}) and (x_i, y_i):

$$\tan(\alpha_i) = \left|\frac{\mathrm{m}_i - \mathrm{n}_i}{1 + \mathrm{m}_i \cdot \mathrm{n}_i}\right|, \ \mathrm{m}_i = \frac{df(x,\beta)}{dx_{0i}}, \ \mathrm{n}_i = \frac{y_i - y_{0i}}{x_i - x_{0i}}$$
$$=> \alpha_i[^\circ] = \tan^{-1}\left(\left|\frac{\mathrm{m}_i - \mathrm{n}_i}{1 + \mathrm{m}_i \cdot \mathrm{n}_i}\right|\right) \cdot \frac{360}{2\pi} \qquad (5)$$

which should be 90°, if the Euclidean distance has been minimized.

```
> mod1
```

```
Nonlinear orthogonal regression model
  model: density ~ Asym/(1 + exp((xmid - log(conc))/scal))
   data: DNase1
 Asym  xmid  scal
2.158 1.287 1.029
 vertical residual sum-of-squares: 0.06665
 orthogonal residual sum-of-squares: 0.06368
 PASSED: 16 out of 16 fitted points are orthogonal.

Number of iterations to convergence: 2
Achieved convergence tolerance: 1.49e-08
```

In this case, all points have been fitted orthogonal, giving the `PASSED` message and all is well. If a `FAILED` message is given, not all of the points are orthogonal and some tweaking is necessary, see next chapter.

3. Tweaking the model in case of non-orthogonality

Two arguments to the `onls` function mainly influence the success of overall orthogonal fitting:

extend: By default, it is set to `c(0.2, 0.2)`, which means that (x_{0i}, y_{0i}) in the inner loop

are also optimized in an extended predictor value x range of $[\min(x) - 0.2 \cdot \text{range}(x), \max(x) + 0.2 \cdot \text{range}(x)]$. This is important for the values at the beginning and end of the data, because the resulting model can display significantly different curvature if (x_{0i}, y_{0i}) are forced to be within the predictor range, often resulting in a loss of orthogonality at the end points.
In the following, we will take an example from the ODRPACK implementation [1].

```
> x <- c(0, 10, 20, 30, 40, 50, 60, 70, 80, 85, 90, 95, 100, 105)
> y <- c(4.14, 8.52, 16.31, 32.18, 64.62, 98.76, 151.13, 224.74, 341.35,
+        423.36, 522.78, 674.32, 782.04, 920.01)
> DAT <- data.frame(x, y)
> mod4 <- onls(y ~ b1 * 10^(b2 * x/(b3 + x)), data = DAT,
+              start = list(b1 = 1, b2 = 5, b3 = 100))
```

```
Obtaining starting parameters from ordinary NLS...
  Passed...
 Relative error in the sum of squares is at most `ftol'.
Optimizing orthogonal NLS...
  Passed... Relative error in the sum of squares is at most `ftol'.
```

With

```
> coef(mod4)
```

```
        b1         b2         b3
  4.487871   7.188156 221.837834
```

we get the same coefficients as in the ODRPACK implementation (4.4879/7.1882/221.8383) and with

```
> deviance_o(mod4)
```

```
[1] 15.26281
attr(,"label")
[1] "Deviance (RSS) of orthogonal residuals from orthogonal model"
```

the same orthogonal residual sum-of-squares (15.263), as both given on page 363.
However, if we **do not use** the (default) extended predictor range and set `extend = c(0, 0)`, x_1 and x_{14} are non-orthogonal, as analyzed by the `check_o` function. See α_1 and α_{14}:

```
> mod5 <- onls(y ~ b1 * 10^(b2 * x/(b3 + x)), data = DAT,
+              start = list(b1 = 1, b2 = 5, b3 = 100), extend = c(0, 0))
```

```
Obtaining starting parameters from ordinary NLS...
  Passed...
 Relative error in the sum of squares is at most `ftol'.
Optimizing orthogonal NLS...
  Passed... Relative error in the sum of squares is at most `ftol'.
```

```
> check_o(mod5)
```

```
    x           x0      y         y0     alpha      df/dx Ortho
1   0 6.268414e-09   4.14   4.519318 71.480996  0.3349642 FALSE
2  10 9.701772e+00   8.52   9.005910 89.999998  0.6137530  TRUE
3  20 1.934099e+01  16.31  16.928822 89.999997  1.0649358  TRUE
4  30 2.999583e+01  32.18  32.182244 89.998072  1.8596604  TRUE
5  40 4.244013e+01  64.62  63.893359 89.999997  3.3580942  TRUE
6  50 5.094935e+01  98.76  98.564953 90.000000  4.8672657  TRUE
7  60 5.988618e+01 151.13 151.146249 89.999558  7.0049486  TRUE
8  70 6.872289e+01 224.74 224.870231 89.999823  9.8067808  TRUE
9  80 7.862153e+01 341.35 341.448888 89.999995 13.9398022  TRUE
10 85 8.398467e+01 423.36 423.420823 89.999850 16.6939697  TRUE
```

```
11  90 8.942638e+01 522.78 522.808813 89.999959 19.9080963  TRUE
12  95 9.625199e+01 674.32 674.269107 89.999989 24.6004639  TRUE
13 100 1.003675e+02 782.04 782.026781 89.998525 27.8176041  TRUE
14 105 1.050000e+02 920.01 919.996237  1.793665 31.8165512 FALSE
```

window: is the window $[x_{i-w}, x_{i+w}]$ in which `optimize` searches for (x_{0i}, y_{0i}) to minimize $\|D_i\|$. The default of `window = 12` works quite well with sample sizes $n > 25$, but may be tweaked, as in the following example when the x values are very close in a region:

```
> x <- 1:100
> y <- x^2
> set.seed(123)
> y <- sapply(y, function(a) rnorm(1, a, 0.1 * a))
> DAT <- data.frame(x, y)
> mod6 <- onls(y ~ x^a, data = DAT, start = list(a = 1))

Obtaining starting parameters from ordinary NLS...
  Passed...
 Relative error in the sum of squares is at most `ftol'.
Optimizing orthogonal NLS...
  Passed... Relative error in the sum of squares is at most `ftol'.

> mod6

Nonlinear orthogonal regression model
  model: y ~ x^a
   data: DAT
    a
2.005
 vertical residual sum-of-squares: 17496215
 orthogonal residual sum-of-squares: 675.3
 FAILED: Only 98 out of 100 fitted points are orthogonal.

Number of iterations to convergence: 2
Achieved convergence tolerance: 1.49e-08
```

Here fitting fails, while it passes when using a larger window size:

```
> mod7 <- onls(y ~ x^a, data = DAT, start = list(a = 10), window = 17)

Obtaining starting parameters from ordinary NLS...
  Passed...
 Conditions for `info = 1' and `info = 2' both hold.
Optimizing orthogonal NLS...
  Passed... Conditions for `info = 1' and `info = 2' both hold.

> mod7

Nonlinear orthogonal regression model
  model: y ~ x^a
   data: DAT
    a
2.005
 vertical residual sum-of-squares: 17496215
 orthogonal residual sum-of-squares: 675.3
 PASSED: 100 out of 100 fitted points are orthogonal.

Number of iterations to convergence: 2
Achieved convergence tolerance: 1.49e-08
```

4. Analysing the orthogonal model with classical `nls` functions

Plotting.

```
> plot(mod1)
```

Due to different scaling of x- and y-axes, orthogonality is often not evident (Figure 1). Scaling both axes equally resolves this issue (Figure 2):

```
> plot(mod1, xlim = c(0, 1), ylim = c(0, 1), asp = 1)
```

Fit features and summaries.
The usual **summary** as in `summary.nls` but with information for *vertical* and *orthogonal* residual standard errors:

```
> summary(mod1)
```

```
Formula: density ~ Asym/(1 + exp((xmid - log(conc))/scal))

Parameters:
     Estimate Std. Error t value Pr(>|t|)
Asym   2.1581     0.2362   9.138 5.06e-07 ***
xmid   1.2868     0.2737   4.701 0.000414 ***
scal   1.0294     0.1191   8.644 9.48e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error of vertical distances: 0.0716 on 13 degrees of freedom
Residual standard error of orthogonal distances: 0.06999 on 13 degrees of freedom

Number of iterations to convergence: 2
Achieved convergence tolerance: 1.49e-08
```

Coefficients:

```
> coef(mod1)
```

```
    Asym     xmid     scal
2.158063 1.286767 1.029365
```

Variance-Covariance matrix:

```
> vcov(mod1)
```

```
           Asym       xmid       scal
Asym 0.05576852 0.06348146 0.02508376
xmid 0.06348146 0.07491901 0.02905975
scal 0.02508376 0.02905975 0.01417978
```

Response value prediction:

```
> predict(mod1, newdata = data.frame(conc = 6))
```

```
[1] 1.338527
```

Profiling confidence intervals:

```
> confint(mod1)
```

```
          2.5%    97.5%
Asym 1.7867613 3.122423
xmid 0.8307868 2.257017
scal 0.7985603 1.370012
```

5. Extracting information based on vertical residuals

Models fitted with `onls` incorporate information with respect to the vertical residuals using the classical `S3` functions.

Vertical residuals:

```
> residuals(mod1)
```

```
 [1] -0.015212734 -0.016965496 -0.002470757  0.007019025 -0.034659599
 [6]  0.051457795 -0.023373864  0.003534017 -0.101099950  0.040960449
[11]  0.188945323 -0.041102160 -0.091141962 -0.042432394  0.047561750
[16] -0.002783375
attr(,"label")
[1] "Vertical residuals from orthogonal model"
```

Fitted values corresponding to x:

```
> fitted(mod1)
```

```
 [1] 0.03240982 0.03240982 0.11950663 0.11950663 0.22249500 0.22249500
 [7] 0.39695431 0.39695431 0.66145991 0.66145991 1.00194214 1.00194214
[13] 1.35858684 1.35858684 1.65991222 1.65991222
attr(,"label")
[1] "Fitted values from orthogonal model"
```

Vertical residual sum-of-squares:

```
> deviance(mod1)
```

```
[1] 0.06664856
attr(,"label")
[1] "Deviance (RSS) of vertical residuals from orthogonal model"
```

Log-likelihood of model using vertical residuals:

```
> logLik(mod1)
```

```
'log Lik.' 21.14427 (df=4)
"Log-likelihood using vertical residuals from orthogonal model"
```

6. Extracting information based on orthogonal residuals

The following functions are meant to extract `S3`-type values based on orthogonal residuals. The naming convention is `function_o`.

Orthogonal residuals:

```
> residuals_o(mod1)
```

```
 [1] 0.012826034 0.014301739 0.002154180 0.006121746 0.031021360 0.046147306
 [7] 0.021674908 0.003278201 0.097183574 0.039396104 0.186391310 0.040540529
[13] 0.090863943 0.042303057 0.047540679 0.002782142
attr(,"label")
[1] "Orthogonal residuals from orthogonal model"
```

Orthogonal residual sum-of-squares:

```
> deviance_o(mod1)
```

```
[1] 0.06367923
attr(,"label")
[1] "Deviance (RSS) of orthogonal residuals from orthogonal model"
```

Log-likelihood of model using orthogonal residuals:

```
> logLik_o(mod1)

'log Lik.' 21.50887 (df=4)
"Log-likelihood using orthogonal residuals from orthogonal model"
```

7. Extracting information about x_{0i} and y_{0i}

Orthogonal fitting is based on finding some pair (x_{0i}, y_{0i}) on the fitted curve that is orthogonal to (x_i, y_i). Values for x_{0i} and y_{0i} can be extracted with `x0` and `y0`:

Extracting $\mathbf{x_{0i}}$:

```
> x0(mod1)

 [1]  0.04190991  0.04110828  0.19425722  0.19830497  0.37674385  0.41094562
 [7]  0.77312338  0.78247423  1.53557868  1.57326227  3.15540931  3.11831456
[13]  6.24290324  6.24669850 12.50141482 12.49991719
attr(,"label")
[1] "x0 values from orthogonal model"
```

Extracting $\mathbf{y_{0i}}$:

```
> y0(mod1)

 [1] 0.02799734 0.02748358 0.11891387 0.12118516 0.21557776 0.23252036
 [7] 0.39367421 0.39744729 0.65374031 0.66452278 1.00699348 1.00082547
[13] 1.35803126 1.35832847 1.65995435 1.65990975
attr(,"label")
[1] "y0 values from orthogonal model"
```

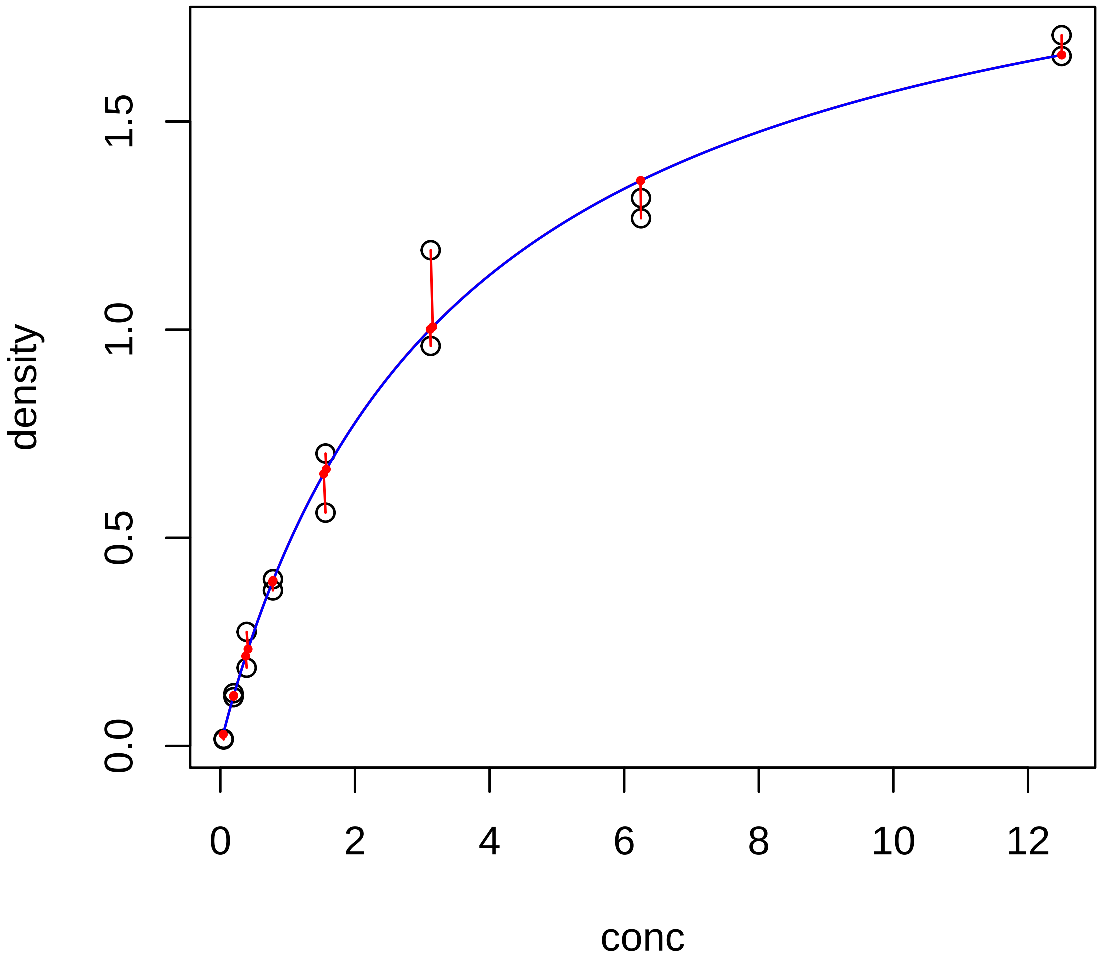

Figure 1: Plot of `mod1` showing the (x_i, y_i) values as black circles, (x_{0i}, y_{0i}) values as red circles and orthogonal lines in red.

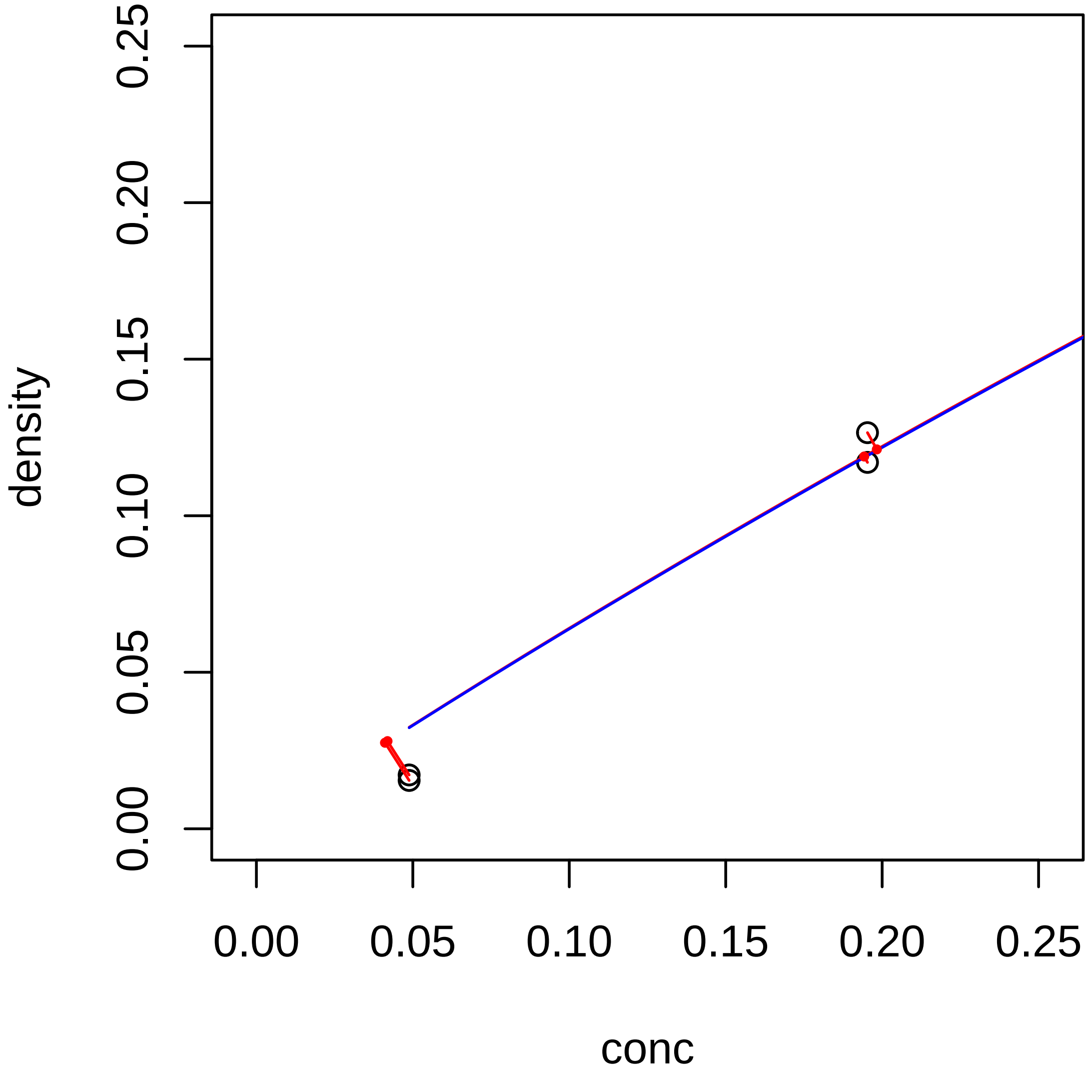

Figure 2: Plot of `mod1` as in Figure 1 with equal scaling for better visualization of orthogonality.

References

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

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

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

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