the key point is the connection between R and Java (which underlies CDK).

Ron Wehrens

Institute for Molecules and Materials
Analytical Chemistry
The Netherlands
R.Wehrens@science.ru.nl

Non-linear regression for optimising the separation of carboxylic acids

by Peter Watkins and Bill Venables

In analytical chemistry, models are developed to describe a relationship between a response and one or more stimulus variables. The most frequently used model is the linear one where the relationship is linear in the parameters that are to be estimated. This is generally applied to instrumental analysis where the instrument response, as part of a calibration process, is related to a series of solutions of known concentration. Estimation of the linear parameters is relatively simple and is routinely applied in instrumental analysis. Not all relationships though are linear with respect to the parameters. One example of this is the Arrhenius equation which relates the effect of temperature on reaction rates:

$$k = A \exp\left(-E_a/RT\right) \times \exp(\epsilon) \qquad (1)$$

where k is the rate coefficient, A is a constant, E_a is the activation energy, R is the universal gas constant, and T is the temperature in degrees Kelvin. As k is the measured response at temperature T, A and E_a are the parameters to be estimated. The last factor indicates that there is an error term, which we assume is multiplicative on the response. In this article we will assume that the error term is normal and homoscedastic, that is, $\epsilon \sim \mathrm{N}(0, \sigma^2)$

One way to find estimates for A and E_a is to transform the Arrhenius equation by taking logarithms of both sides. This converts the relationship from a multiplicative one to a linear one with homogeneous, additive errors. In this form linear regression may be used to estimate the coefficients in the usual way.

Note that if the original error structure is not multiplicative, however, and the appropriate model is, for example, as in the equation

$$k = A \exp\left(-E_a/RT\right) + \epsilon \qquad (2)$$

then taking logarithms of both sides does not lead to a linear relationship. While it may be useful to ignore this as a first step, the optimum estimates can only be obtained using non-linear regression techniques, that is by least squares on the original scale and not in the logarithmic scale. Starting from initial values for the unknown parameters, the estimates are iteratively refined until, it is hoped, the process converges to the maximum likelihood estimates.

This article is intended to show some of the powerful general facilities available in R for non-linear regression, illustrating the ideas with simple, yet important non-linear models typical of those in use in Chemometrics. The particular example on which we focus is one for the response behaviour of a carboxylic acid using reverse-phase high performance liquid chromatography and we use it to optimise the separation of a mixture of acids.

Non-linear regression in general is a very unstructured class of problems as the response function of the regression may literally be any function at all of the parameters and the stimulus variables. In specific applications, however, certain classes of non-linear regression models are typically of frequent occurrence. The general facilities in R allow the user to build up a knowledge base in the software itself that allows the fitting algorithm to find estimates for initial values and to find derivatives of the response function with respect to the unknown parameters automatically. This greatly simplifies the model fitting process for such classes of models and usually makes the process much more stable and reliable.

The working example

Aromatic carboxylic acids are an important class of compounds since many are pharmacologically and biologically significant. Thus it is useful to be able to separate, characterise and quantify these types of compounds. One way to do this is with chromatography, more specifically, reverse phase high performance liquid chromatography (RP-HPLC). Due to the ionic nature of these compounds, analysis by HPLC can be complicated as the hydrogen ion concentration is an important factor for separation of these compounds. Waksmundzka-Hajnos (1998) reported a widely used equation that models the separation of monoprotic carboxylic acids (i.e. containing a single hydrogen ion) depending on the hydrogen ion concentration, $[H^+]$. This is given by

$$k = \frac{(k_{-1} + k_0([H^+]/K_a))}{(1 + [H^+]/K_a)} + \epsilon \qquad (3)$$

where k is the capacity factor, k_0 and k_{-1} are the k values for the non-ionised and ionised forms of the

carboxylic acid, and K_a is the acidity constant. In this form, non-linear regression can be applied to Equation 3 to estimate the parameters, k_0, k_{-1} and K_a. Deming and Turoff (1978) reported HPLC measurements for four carboxylic acids containing a single hydrogen ion; benzoic acid (BA), *o*-aminobenzoic acid (OABA), *p*-aminobenzoic acid (PABA) and *p*-hydroxybenzoic acid (HOBA).

The `R` data set we use here has a stimulus variable named `pH` and four responses named `BA`, `OABA`, `PABA` and `HOBA`, the measured capacity factors for the acids at different pH values. The data set is given as an appendix to this article.

Initial values

While non-linear regression is appropriate for the final estimates, we can pay less attention to the error structure when trying to find initial values. In fact we usually disregard it and simply manipulate the equation to a form where we can find initial values by simple approximation methods. One way to do this for this example, suggested to us by Petra Kuhnert, is to multiply the equation by the denominator and re-arrange to give

$$k[H^+] \approx k_{-1}K_a + kK_a + [H^+]k_0 = \theta + kK_a + [H^+]k_0 \quad (4)$$

where the three unknowns on the right hand side are $\theta = k_{-1}K_a$, K_a and k_0. The initial values can found by ignoring the error structure and simply regressing the artificial response, $k[H^+]$ on the notional 'stimulus' variables k and $[H^+]$, with an intercept term. The initial values for k_{-1}, k_0 and K_a then follow simply.

We can in fact do slightly better than this by noting that if the value for the parameter K_a were known, the original equation is then linear in k_{-1} and k_0. In principle, then, we could use the approximate idea to obtain an initial value for K_a only and then use the entire data set to find initial values for the other two parameters by linear regression with the original equation. Intuitively we might expect this to yield slightly better initial estimates as while the value for K_a is still somewhat doubtful, at least those for k_{-1} and k_0 do in fact respect the appropriate error structure, but sadly there is no guarantee that this will be the case.

Partially linear models

The non-linear fitting function supplied with `R` is `nls`. Normally the user has to supply initial values for all unknown parameters. However if the linear regression has a form like the one above where, if some of the parameters are known, the regression becomes linear in the others, a special algorithm allows us to capitalise on this fact. The benefit is two-fold: firstly, we need only supply initial values for some of the parameters and secondly, the algorithm is typically more stable than it would otherwise be.

Parameters in the first set, (for our example, just K_a) are called the "non-linear parameters" while the second set are the "linear parameters" (though a better term might be "conditionally linear parameters"). The so-called "partially linear" fitting algorithm requires only that initial values be supplied for the non-linear parameters. To illustrate the entire fitting process, we note that when `BA` is the response, a reasonable initial value is $K_a = 0.0001$. Using this we can fit the regression as follows:

```
> tmp <- transform(ba, H = 10^(-pH))
> ba.nls <- nls(BA ~ cbind(1, H/Ka)/(1 + H/Ka),
   data = tmp, algorithm = "plinear",
   start = c(Ka = 0.0001), trace = T)
13.25806 :  0.000100  3.531484 54.819291
7.180198 : 4.293692e-05 5.890920e-01 4.197178e+01
1.441950 : 5.600297e-05 1.635335e+00 4.525067e+01
1.276808 : 5.906698e-05 1.831871e+00 4.597552e+01
1.276494 : 5.920993e-05 1.840683e+00 4.600901e+01
1.276494 : 5.921154e-05 1.840783e+00 4.600939e+01
> coef(ba.nls)
          Ka        .lin1        .lin2
5.921154e-05 1.840783e+00 4.600939e+01
```

The first step is for convenience only. Notice that the regression function is specified as a matrix whose columns are, in general, functions of the non-linear parameters. The coefficients for this non-linear "model matrix" are then the estimates of the linear parameters.

The trace output shown above shows at each step the current residual sum of squares and the parameter estimates, beginning with the non-linear parameters. As the model specification makes no explicit reference to the linear parameters, the estimates are named using a simple naming convention, as shown in the names of the final coefficient vector.

The trace output shows that the initial value is not particularly good and the iterative process is struggling, somewhat.

Self-starting non-linear regressions

The idea behind a self-starting non-linear regression model is that we supply to the fitting algorithm enough information for the process to generate a starting value from the data itself. This involves two steps. Firstly we need to write an *initial value routine* which calculates the initial values from the data set. Secondly we need to generate the self-starting object itself, which will use this initial value routine.

We begin with the initial value routine. As this will be called by the fitting algorithm and not directly by the user, it has to use a somewhat obscure convention for both the parameter list it uses and the way it supplies its values. This convention initially appears to be obscure, but it becomes less so with time. Such an initial value routine can be defined for this kind of regression model as follows:

```
SSba.init <- function(mCall, data, LHS) {
#
# k ~ (k_1 + k_0*H/Ka)/(1 + H/Ka); H = 10^(-pH)
#
      H <- 10^(-eval(mCall[["pH"]], data))
      k <- eval(LHS, data)

      Ka <- as.vector(coef(lsfit(cbind(H, -k),
                H * k, int = TRUE))[3])
      b <- coef(nls(k ~ cbind(1, H/Ka)
                           /(1 + H/Ka),
                data = data.frame(k = k, H = H),
                algorithm = "plinear",
                start = c(Ka = Ka)))
      names(b) <- mCall[c("Ka", "k_1", "k_0")]
      b
}
```

The initial comment is a reminder of how the model will be specified in stage two of the process, and is optional. The way the parameters are accessed is conventional.

This initial value function actually goes a step further and fits the non-linear regression itself using the `plinear` algorithm, so the initial values should be very good indeed! This is the simplest way to capitalise both on the self-starting model and using the `plinear` model along the way. The model fitting process at the second stage should then converge in one iteration, which can act as a check on the process.

Note that the value supplied by the function is a vector with names. These names are not necessarily the same as in the defining formula, so the final step of giving names to the value object must again do so in a conventional way, which will ensure that actual names used by the user of the self-starting regression will be correctly specified. In other words, the choice of names for the parameters remains open to the user and is not fixed in advance.

To specify a self-starting model, we now proceed as follows:

```
SSba <- selfStart( ~ (k_1 + k_0*10^(-pH)/Ka)
                      /(1 + 10^(-pH)/Ka),
        initial = SSba.init,
        parameters = c("Ka", "k_1", "k_0"),
        template = function(pH, k_1, k_0, Ka) {})
```

The first argument is a one-sided formula giving the response function. This now has to respect the name choices used in the initial value routine, of course. The remaining arguments are the initial value routine itself, the parameters as a character vector and a "function template", that is, a dummy function that specifies, in effect, just the argument list. The `selfStart` function itself supplies the innards of this function, so at the end of the call, `SSba` is a function with this argument list that may be used to specify the right hand side of a non-linear regression.

Fitting the regressions is now a simple task. We fit all four as follows:

```
ba.SS <- nls(BA ~ SSba(pH, k_1, k_0, Ka),
             data = ba, trace = T)
oaba.SS <- update(ba.SS, OABA ~ .)
paba.SS <- update(ba.SS, PABA ~ .)
hoba.SS <- update(ba.SS, HOBA ~ .)
```

We have suppressed the trace output, but in fact all four (appear to) converge in a single step. Of course the hard work is done inside the initial value function.

Comparison with published values

We can compare the estimates we get from this rather limited data set with published values, at least for the acidity constant, K_a. These values can be found in Morrison and Boyd (1993).

```
> Kas <- sapply(list(ba = ba.SS, oaba = oaba.SS,
        paba = paba.SS, hoba = hoba.SS), coef)[1,]
> signif(Kas, 4)
       ba      oaba      paba      hoba
5.921e-05 6.449e-06 9.631e-06 2.397e-05
> pKas <- scan(quiet = TRUE)
1: 5.8e-05 1.6e-05 1.4e-05 2.6e-05
5:
> KAvalues <- cbind(Calculated = Kas,
                    Published = pKas)
> signif(KAvalues, 2)
     Calculated Published
ba      5.9e-05   5.8e-05
oaba    6.4e-06   1.6e-05
paba    9.6e-06   1.4e-05
hoba    2.4e-05   2.6e-05
```

In most cases the agreement is quite good.

Thus, the agreement between K_a values suggests that the calculated parameter estimates can be used for modelling the HPLC retention behaviour of the carboxylic acids. To verify this, the predicted values can be compared with the experimental data. Figure 1 shows the predicted and measured data as a function of $[H^+]$ or $pH = -\log_{10}[H^+]$.

To display the fitted self-starting functions we evaluate them on a finer scale of pH values than we have in the observations themselves. This can be done using the `predict` function in R, but we find it useful here to evaluate the predictions directly using `eval`, as we needed to use in the initial value routine.

```
form <- Quote((k_1 + k_0*10^(-pH)/Ka)/
                (1 + 10^(-pH)/Ka))
## evaluate on a fine grid of pH values

at <- function(x, m) c(x, as.list(coef(m)))
pHData <- list(pH = seq(3.5, 6, len = 250))
pHData <- transform(pHData,
    ba = eval(form, at(pHData,   ba.SS)),
  oaba = eval(form, at(pHData, oaba.SS)),
  paba = eval(form, at(pHData, paba.SS)),
  hoba = eval(form, at(pHData, hoba.SS)))


## first plot the fitted lines
with(pHData,
```

```
  matplot(pH, cbind(ba, oaba, paba, hoba),
          col = 16:19, type = "l", lty = 1,
          ylab = "Capacity factor",
          main = "Retention behaviour"))

## then add the observed points
with(ba,
     matpoints(pH, cbind(BA, OABA, PABA, HOBA),
     col = 16:19, pch = 16:19, cex = 0.8))

legend(5.5, 35, c("BA", "OABA", "PABA", "HOBA"),
       text.col = 16:19, col = 16:19, lty = 1,
       pch = 16:19, bty = "n")
```

Notice how `eval` may be used to evaluate a quoted expression, using a list with names as the second argument to provide values for the variables and parameters.

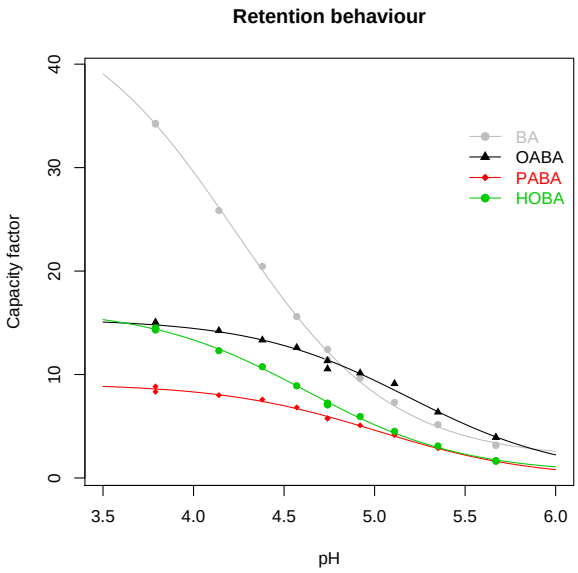

Figure 1: Plot of capacity factor for selected aromatic carboxylic acids against pH

Figure 1 shows that the fit of the predicted values from the self-starting models appears to be quite good with the experimental data.

Diagnostics

The residual plot (predicted - measured *vs* predicted) (Figure 2) suggests that there may be some outliers. This is particularly notable for BA and OABA.

Further presentation of results

A 'window diagram' is a graphical method for showing chromatographic data (Cooper and Hurtubise, 1985), and is suitable for visualing the degree of chromatographic separation. The degree of separation can be optimised by maximising the selectivity, α, of the most difficult to resolve peak pairs as a function of pH. The selectivity factor, α_{ij}, is defined by:

$$\alpha_{ij} = k_i / k_j \tag{5}$$

where k_i and k_j are the respective capacity factors. For their data, Deming and Turoff (1978) ensured that α was greater than or equal to 1. In the cases where α was found to be less than 1 then the reciprocal was taken and used for α. This amounts to defining α_{ij} as

$$\alpha_{ij} = \max(k_i, k_j) / \min(k_i, k_j) \tag{6}$$

```
alpha <- function(ki, kj) pmax(ki,kj)/pmin(ki,kj)
pHData <- transform(pHData,
          aBO = alpha(  ba, oaba),
          aBP = alpha(  ba, paba),
          aBH = alpha(  ba, hoba),
          aOP = alpha(oaba, paba),
          aOH = alpha(oaba, hoba),
          aPH = alpha(paba, hoba))
```

The window diagram (Figure 3) can be produced by plotting the selectivity factor, α_{ij}, as a function of pH.

```
with(pHData,
  matplot(pH, cbind(aBO, aBP, aBH, aOP, aOH, aPH),
          type = "l", col = 16:21, lty = 1:6,
          main = "Window diagram",
          ylab = "Selectivity factor"))
abline(v = 4.5, lty = 4)

leg.txt <-
    c("aBO", "aBP", "aBH", "aOP", "aOH", "aPH")
legend(5.5, 4.5, leg.txt, col = 16:21,
          text.col = 16:21, lty = 1:6, bty = "n")
```

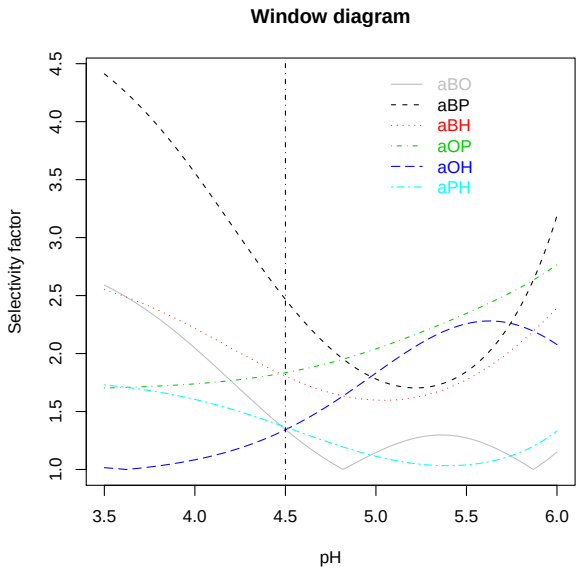

Figure 3: Window diagram for all six combinations of carboxylic acids

The conditions that give α as unity represent a minimum performance for the chromatographic separation while separations with $\alpha > 1$ represent better separation for that pair of compounds. Thus, we

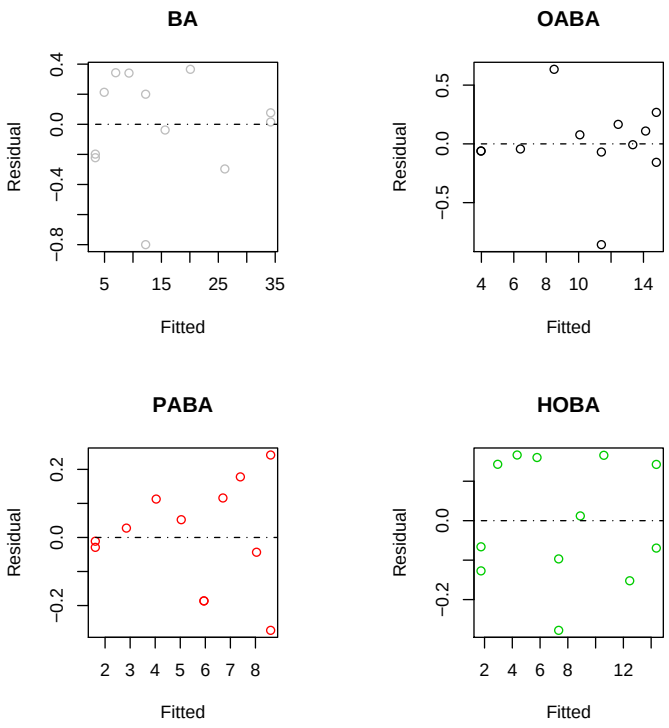

Figure 2: Residual plots for non-linear regression for BA, OABA, PABA and HOBA

seek the point where the best separation occurs for the worst separated compounds. Inspection of Figure 3 shows that this occurs at pH 4.5.

Final remarks

In conclusion, non-linear regression has been used to model the retention behaviour of four related carboxylic acids using Equation 1. Using the predicted values for each acid, the optimum point for separation of the four compounds was identified and used to find the optimal pH.

The example is simple, but sufficient to show the power of non-linear regression in Chemometric applications, and, more importantly, the value of investing some time and effort to produce self-starting model functions for specific classes of regressions. The example also shows some of the power and flexibility of R itself to investigate the data, to fit the models, to check the diagnostics, to predict from the estimates and to present the results.

Bibliography

H. A. Cooper and R. J. Hurtubise. Solubility parameter and window diagram optimization methods for the reversed-phase chromatographic optimization of complex hydroxyl aromatic mixtures. *J. Chrom. A*, 328, 81–91, 1985. 5

S. N. Deming and M. L. H. Turoff. Optimisation of reverse-phase liquid chromatographic separation of weak organic acids. *Anal. Chem.*, 58, 546–548, 1978. 3, 5, 7

R. T. Morrison and R. N. Boyd. *Organic Chemistry.* Allyn and Bacon, Newton, Mass, 5th edition, 1993. 4

M. Waksmundzka-Hajnos. Chromatographic separations of aromatic carboxylic acids. *J. Chrom. B*, 717, 93–118, 1998. 2

Peter Watkins
CSIRO Food Science Australia
peter.watkins@csiro.au

Bill Venables
CSIRO Mathematical and Information Sciences
bill.venables@csiro.au

Appendix

The data set used for this article is shown in the table. It is taken from Deming and Turoff (1978) where it was published in retention times. For this article, the data has been converted to their corresponding capacity factors.

pH	BA	OABA	PABA	HOBA
3.79	34.21	15.06	8.85	14.30
3.79	34.27	14.64	8.33	14.52
4.14	25.85	14.24	8.00	12.30
4.38	20.46	13.33	7.58	10.76
4.57	15.61	12.61	6.82	8.91
4.74	12.42	11.33	5.76	7.24
4.74	11.42	10.55	5.76	7.06
4.92	9.64	10.15	5.09	5.94
5.11	7.30	9.12	4.15	4.52
5.35	5.15	6.36	2.88	3.09
5.67	3.18	3.92	1.60	1.68
5.67	3.18	3.92	1.58	1.62

Fitting dose-response curves from bioassays and toxicity testing

by Johannes Ranke

Introduction

During the development of new chemicals, but also in risk assessment of existing chemicals, they have to be characterized concerning their potential to harm biological organisms. Characterizing chemicals according to this potential has many facettes and requires various types of experiments. One of the most important types is the dose-response experiment.

In such experiments, the responses of biological organisms to different doses[1] are observed in a quantitative way. Examples of the observed variables (endpoints of toxicity) are the length of wheat seedlings after being exposed to different concentrations of the chemical substance for a defined time interval, the activity of luminescent bacteria, the ability of cell cultures to reduce a specific dye, the growth rate according to number of individuals or biomass, the number of viable offspring and many others.

These observed variables have in common that a reference magnitude for healthy and viable organisms can be defined (normalised response level $r = 1$), and that the magnitude of the variable (response) is limited by a zero response ($r = 0$) where the maximum of the effect is observed. The **drfit** package covers the case where there is a continuum of possible response values between 0 and 1 (inclusive). Additionally, responses above 1 are frequently observed due to variability or as the result of stimulation by a subtoxic dose, and even responses below 0 may be present, depending on the type of data and the applied preprocessing.

If the responses are binomial, such as life and death for a number of individuals, it is adviseable to choose the readily available `glm` fitting procedures (generalized linear models), where the probit and logit links are already built-in (e.g. Chapter 7.2 in Venables and Ripley (2002)) or to look into the **drc** package.

Dose-response relationships for continuous response tests can generally be expressed as

$$r = f(d, \vec{p}) + \epsilon \quad (1)$$

where r is the normalised response at dose d, $f(d, \vec{p})$ is the model function with parameter vector $\vec{p}$, and ϵ is the error variable describing the variability in the observations not explainable by the model function $f(d, \vec{p})$.

This article shows how different model functions $f(d, \vec{p})$ can be conveniently fitted to such dose-response data using the R package **drfit**, yielding the vector of parameters $\vec{p}$ that gives the description of the data with the least residual error. The fitting can be carried out for many substances with a single call to the main function `drfit`.

The results that the user will probably be most interested in are the doses at which a response of 50 % relative to healthy control organisms is to be expected (termed ED_{50}), as this is a very robust parameter describing the toxicity of the substance toward the organism investigated.

The **drfit** package internally uses the R function `nls` for nonlinear regression analysis as detailed by Bates and Watts (1988). Confidence intervals for the model parameters are calculated by the `confint.nls` function from the **MASS** package as described in Venables and Ripley (2002).

drfit defines a dose-response data representation as a special case of an R dataframe, facilitates fitting standard dose-response models (probit, logit,

[1] The term dose is used here in a generalised way, referring to doses in the strict sense like mg oral intake per kg body weight as well as to measured concentrations in aquatic toxicity tests or nominal concentrations in cell culture assays.