Experimental methods in agriculture

Andrea Onofri and Dario Sacco

Update: v. 0.99 (2021-12-01), compil. 2021-12-01

Contents

Introduction

Placeholder

Aims

How this book is organised

Statistical software

The authors

Chapter 1

Science and pseudoscience

Placeholder

1.1 Science needs data

1.2 Not all data support science

1.3 Good data is based on good 'methods'

1.4 The 'falsification' principle

1.5 Trying to falsify a result

1.6 The basic principles of experimental design

1.6.1 Control

1.6.2 Replication

1.6.3 Randomisation

1.7 Invalid experiments

1.7.1 Lack of good control

1.7.2 'Confounding' and spurious correlation

1.7.3 Lack of true-replicates or careless randomisation

1.8 How can we assess whether the data is valid?

1.9 Conclusions

1.10 Further readings

Chapter 2

Designing experiments

Placeholder

2.1 The elements of research

2.2 Hypothesis and objectives

2.3 The experimental treatments

2.3.1 Factorial experiments

2.3.2 The control

2.4 The experimental units

2.5 The allocation of treatments

2.6 The variables

2.6.1 Nominal variables

2.6.2 Ordinal variables

2.6.3 Count and ratio variables

2.6.4 Continuous variables

2.6.5 Sensory and visual assessments

2.7 Setting up a field experiment

2.7.1 Selecting the field

2.7.2 Selecting the units within the field

2.7.3 Number of replicates

2.7.4 The field map

2.7.5 The experimental lay-out

Completely randomised design (CR)

Randomised complete block design (RCBD)

Chapter 3

Describing the observations

Placeholder

3.1 Quantitative data

3.1.1 Statistics of location

3.1.2 Statistics of spread

3.1.3 Summing the uncertainty

3.1.4 Relationship between quantitative variables

3.2 Nominal data

3.2.1 Distributions of frequencies

3.2.2 Descriptive stats for distributions of frequencies

3.2.3 Contingency tables

3.2.4 Independence

3.3 Descriptive stats with R

3.4 Graphical representations

3.5 Further reading

Chapter 4

Modeling the experimental data

Placeholder

4.1 Deterministic models

4.2 Stochastic models

4.2.1 Probability functions

4.2.2 Density functions

4.2.3 The Gaussian PDF and CDF

4.3 A model with two components

4.4 And so what?

4.5 Monte Carlo methods to simulate an experiment

4.6 Data analysis and model fitting

4.7 Some words of warning

4.8 Further readings

Chapter 5

Estimation of model parameters

Placeholder

5.1 Example 1: a concentration value

5.1.1 The empirical sampling distribution

5.1.2 A theoretical sampling distribution

5.1.3 The frequentist confidence interval

5.2 Example 2: a proportion

5.3 Conclusions

5.4 Further readings

Chapter 6

Making Decisions under uncertainty

Placeholder

6.1 Comparing sample means: the Student's t-test

6.1.1 The dataset

6.1.2 Monte Carlo simulation

6.1.3 A formal solution

6.1.4 The t test with R

6.2 Comparing proportions: the χ^2 test

6.3 Correct interpretation of the P-value

6.4 Conclusions

6.5 Further readings

Chapter 7

One-way ANOVA models

Placeholder

7.1 Comparing herbicides in a pot-experiment

7.2 Data description

7.3 Model definition

7.3.1 Parameterisation

7.3.2 Treatment constraint

7.3.3 Sum-to-zero constraint

7.4 Basic assumptions

7.5 Fitting ANOVA models by hand

7.5.1 Parameter estimation

7.5.2 Residuals

7.5.3 Standard deviation σ

7.5.4 SEM and SED

7.5.5 Variance partitioning

7.5.6 Hypothesis testing

7.6 Fitting ANOVA models with R

7.7 Expected marginal means

7.8 Conclusions

7.9 Further readings

Chapter 8

Checking for the basic assumptions

Placeholder

8.1 Outlying observations

8.2 The inspection of residuals

8.2.1 Plot of residuals against expected values

8.2.2 QQ-plot

8.3 Formal hypotesis testing

8.4 What do we do, in practice?

8.5 Correcting measures

8.5.1 Removing outliers

8.5.2 Stabilising transformations

8.6 Examples with R

8.7 Example 1

8.8 Example 2

8.9 Other possible approaches

8.10 Further readings

Chapter 9

Contrasts and multiple comparison testing

Placeholder

9.1 Back to the 'mixture' example

9.2 Linear contrasts

9.2.1 The variance of contrasts

9.2.2 Testing linear contrasts with R

9.3 Pairwise comparisons

9.4 Letter display

9.5 Multiplicity correction

9.6 Multiple comparisons with transformed data

9.7 What about the traditional MCPs?

9.8 Some practical suggestions

9.9 Further readings

Chapter 10

Multi-way ANOVA models

Placeholder

10.1 Motivating example: a genotype experiment in blocks

10.2 Model definition

10.3 Model fitting by hand

10.4 Model fitting with R

10.4.1 Model checking

10.4.2 Variance partitioning

10.5 Another example: comparing working protocols

Chapter 11

Multi-way ANOVA models with interactions

Placeholder

11.1 The 'interaction' concept

11.2 Genotype by N interactions

11.2.1 Model definition

11.2.2 Model fitting by hand

11.2.3 Model fitting with R

11.2.4 Inferences and standard errors

11.2.5 Expected marginal means

11.3 Nested effects: maize crosses

11.3.1 Model definition

11.3.2 Parameter estimation

11.4 Further readings

Chapter 12

Plots of different sizes

Placeholder

12.1 Example 1: a split-plot experiment

12.1.1 Model definition

12.1.2 Model fitting with R

12.2 Example 2: a strip-plot design

12.2.1 Model definition

12.2.2 Model fitting with R

12.3 Further readings

Chapter 13

Simple linear regression

Placeholder

13.1 Case-study: N fertilisation in wheat

13.2 Preliminary analysis

13.3 Definition of a linear model

13.4 Parameter estimation

13.5 Goodness of fit

13.5.1 Graphical evaluation

13.5.2 Standard errors for parameter estimates

13.5.3 F test for lack of fit

13.5.4 F test for goodness of fit and coefficient of determination

13.6 Making predictions

13.7 Further readings

Chapter 14

Nonlinear regression

Biological processes, very rarely follow linear trends. Just think about how a crop grows, or responds to increasing doses of fertilisers/xenobiotics. Or, think about how an herbicide degrades in the soil, or about the germination pattern of a seed population. It is very easy to realise that curvilinear trends, asymptotes and/or inflection points are far more common, in nature. In practice, linear equations in biology are nothing more than a quick way to approximate a response over a very narrow range of the independent variable.

Therefore, we need to be able to fit simple nonlinear models to our experimental data. In this Chapter, we will see how to do this, starting from a simple, but realistic example.

14.1 Case-study: a degradation curve

A soil was enriched with the herbicide metamitron, up to a concentration of 100 ng g^1. It was put in 24 aluminium containers, inside a climatic chamber at 20°C. Three containers were randomly selected in eight different times and the residual concentration of metamitron was measured. The observed data are available in the 'degradation.csv' file, in an external repository. First of all we load and inspect the data.

```
fileName <- "https://www.casaonofri.it/_datasets/degradation.csv"
dataset <- read.csv(fileName, header=T)
head(dataset, 10)
```

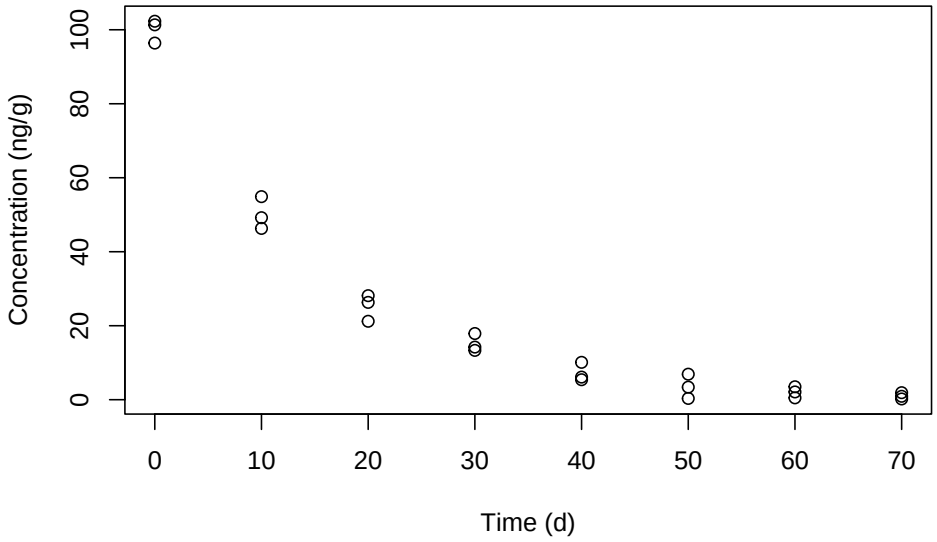

Figure 14.1: Degradation of metamitron in soil

```
##    Time   Conc
## 1     0  96.40
## 2    10  46.30
## 3    20  21.20
## 4    30  17.89
## 5    40  10.10
## 6    50   6.90
## 7    60   3.50
## 8    70   1.90
## 9     0 102.30
## 10   10  49.20
```

It is always very useful to take a look at the observed data, by plotting the response against the predictor (Figure 14.1); we see that the trend is curvilinear, which rules out the use of simple linear regression.

In order to build a nonlinear regression model, we can use the general form introduced in Chapter 4, that is:

$$Y_i = f(X_i, \theta) + \varepsilon_i$$

where X is the time, Y is the concentration, f is a nonlinear function, θ is the set of model parameters and ε are the residuals, assumed as independent, gaussian and homoscedastic.

14.2 Model selection

The first task is to select a function f that matches the observed response shape. You can find full detail elsewhere (go to this link). For now, we can look at Figure 14.2, where we have displayed the shapes of some useful functions for nonlinear regression analysis in agricultural studies. We distinguish:

1. Convex/concave shapes (e.g., quadratic polynomial, exponential growth/decay, asymptotic growth, power curve and rectangular hyperbola)
2. Sigmoidal shapes (e.g., logistic growth, Gompertz growth and log-logistic dose-response)
3. Curves with maxima/minima (e.g., Bragg's function)

In the above list, the quadratic polynomial is a curvilinear model, but it is linear in the parameters and, therefore, it should have been better included in the previous Chapter about linear regression. However, we decided to include it here, considering its concave shape.

Behind each of the above shapes, there is a mathematical function, which is shown in Figure 14.3).

For the dataset under study, considering the shapes in Figure 14.2 and, above all, considering the available literature information, we select an exponential decay equation, with the following form (see also Fig. 14.3):

$$Y_i = a\, e^{-k\, X_i} + \varepsilon_i$$

where Y is the concentration at time X, a is the initial metamitron concentration and k is the constant degradation rate.

14.3 Parameter estimation

Now, we have to obtain the least squares estimates of the parameters a and k. Unfortunately, with nonlinear regression, the least squares problem has no closed form solution and, therefore, we need to tackle the estimation problem in a different way. In general, there are three possibilities

1. linearise the nonlinear function;
2. approximate the nonlinear function by using a polynomial;
3. use numerical methods for the minimisation.

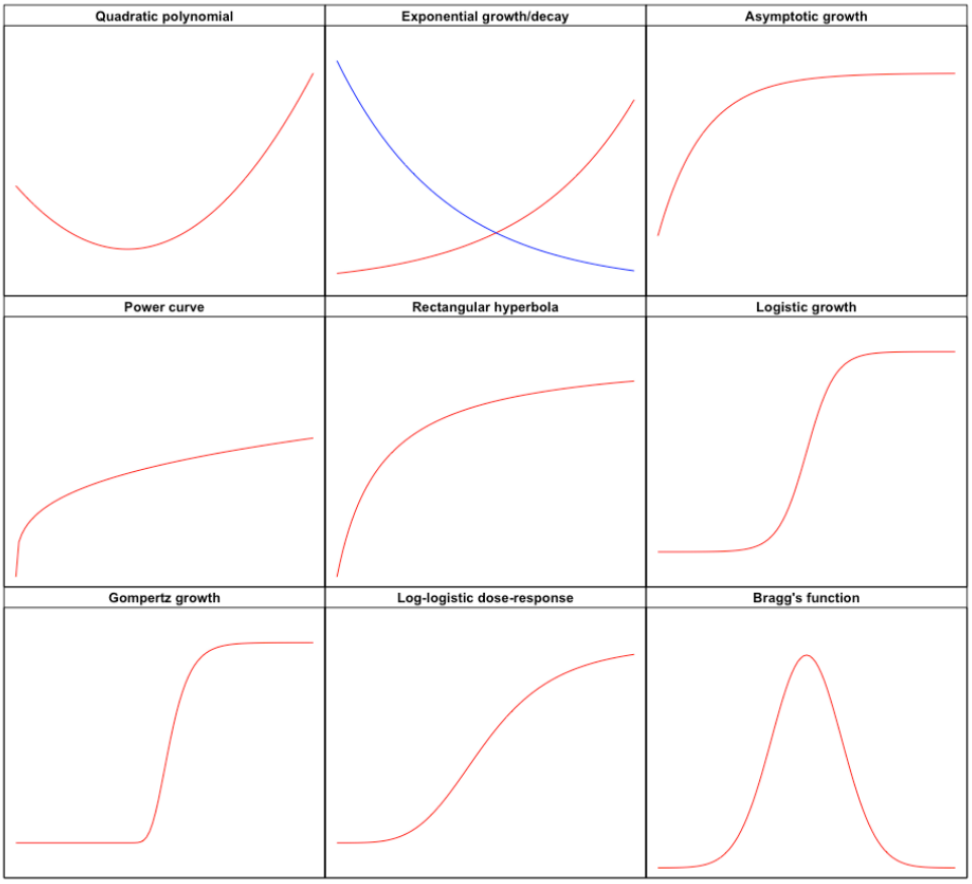

Figure 14.2: The shapes of the most important functions

Name	Equation	R function
Straight line	$Y = b_0 + b_1\,X$	`NLS.linear()`
Quadratic polynomial	$Y = b_0 + b_1\,X + b_2\,X^2$	`NLS.poly2()`
Exponential	$Y = a\,e^{kX}$	`NLS.expoGrowth()` `NLS.expoDecay()`
Asymptotic	$Y = a - (a - b)\,\exp(-cX)$	`NLS.asymReg()`
Power	$Y = a\,X^b$	`NLS.powerCurve()`
Logarithmic	$Y = a + b\,\log(X)$	`NLS.logCurve()`
Rectangular hyperbola	$Y = \frac{a\,X}{b+X}$	`SSmicmen()`
Logistic	$Y = c + \frac{d-c}{1+exp(-b(X-e))}$	`NLS.L4()` `NLS.L3()` `NLS.L2()`
Gompertz	$Y = c + (d-c)\,\exp\{-\exp[-b\,(X-e)]\}$	`NLS.G4()` `NLS.G3()` `NLS.G2()`
Modified Gompertz	$Y = c + (d-c)\,\{1-\exp\{-\exp[b\,(X-e)]\}\}$	`NLS.E4()` `NLS.E3()` `NLS.E2()`
Log-logistic	$Y = c + \frac{d-c}{1+\exp\{-b[\log(X)-\log(e)]\}}$	`NLS.LL4()` `NLS.LL3()` `NLS.LL2()`
Weibull I	$Y = c + (d-c)\,\exp\{-\exp[-b\,(\log(X)-\log(e))]\}$	`NLS.W1.4()` `NLS.W1.3()` `NLS.W1.2()`
Weibull II	$Y = c + (d-c)\,\{1-\exp\{-\exp[b\,(\log(X)-\log(e))]\}\}$	`NLS.W2.4()` `NLS.W2.3()` `NLS.W2.2()`
Bragg	$Y = c + (d-c)\,\exp\left[-b(X-e)^2\right]$	`NLS.bragg.3()` `NLS.bragg.4()`
Lorentz	$Y = c + \frac{d-c}{1+b(X-e)^2}$	`NLS.lorentz.3()` `NLS.lorentz.4()`
Beta	$Y = d\left\{\left(\frac{X-X_b}{X_o-X_b}\right)\left(\frac{X_c-X}{X_c-X_o}\right)^{\frac{X_c-X_b}{X_o-X_b}}\right\}^b$	`NLS.beta()`

Figure 14.3: Some useful functions for nonlinear regression analysis (equation and R function).

Let's see advantages and disadvantages of all methods.

14.3.1 Linearisation

In some cases, we can modify the function to make it linear. In this case, we can take the logarithm of both sides and define the following equation:

$$\log(Y) = \log(A) - k\,X$$

which is, indeed, linear. Therefore, we can take the logarithms of the concentrations and fit a linear model, by using the `lm()` function:

```
mod <- lm(log(Conc) ~ Time, data=dataset)
summary(mod)
```

```
##
## Call:
## lm(formula = log(Conc) ~ Time, data = dataset)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.11738 -0.09583  0.05336  0.31166  1.01243
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.662874   0.257325   18.12 1.04e-14 ***
## Time        -0.071906   0.006151  -11.69 6.56e-11 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6905 on 22 degrees of freedom
## Multiple R-squared:  0.8613, Adjusted R-squared:  0.855
## F-statistic: 136.6 on 1 and 22 DF,  p-value: 6.564e-11
```

We see that the slope has a negative sign (the line is decreasing) and the intercept is not the initial concentration value, but its logarithm.

The advantage of this process is that the calculations are very easily performed with every spreadsheet or simple calculator. However, we need to

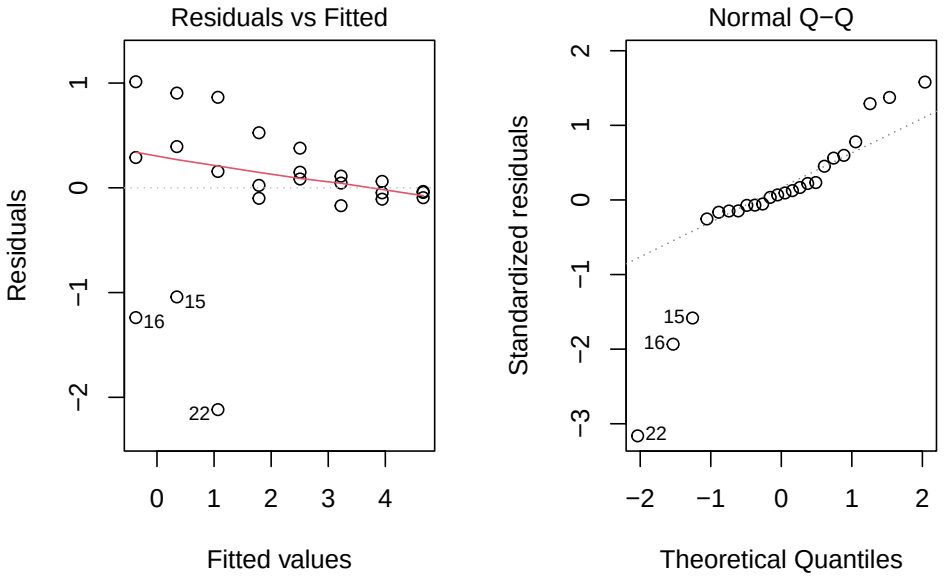

Figure 14.4: Graphical analyses of residuals for nonlinear regression with linearisation of the mean function

check that the basic assumptions of normal and homoscedastic residuals hold in the logarithmic scale. We do this in the following box.

```
par(mfrow = c(1,2))
plot(mod, which = 1)
plot(mod, which = 2)
```

The plots show dramatic deviations from homoscedasticity (Figure 14.4), which rule out the use of the linearisation method. Please, consider that this conclusion is specific to our dataset, while, in general, the linearisation method can be successfully used, whenever the basic assumptions for linear models appear to hold for the transformed response.

14.3.2 Approximation with a polynomial function

In some other cases, a curvilinear shape can be approximated by using a polynomial equation, that is, indeed, linear in the parameters. In our example, we could think of the descending part of a second order polynomial (see Figure 14.4) and, thus, we fit this model by the usual `lm()` function:

```
mod2 <- lm(Conc ~ Time + I(Time^2), data=dataset)
pred <- predict(mod2, newdata = data.frame(Time = seq(0, 70, by = 0.1)))
```

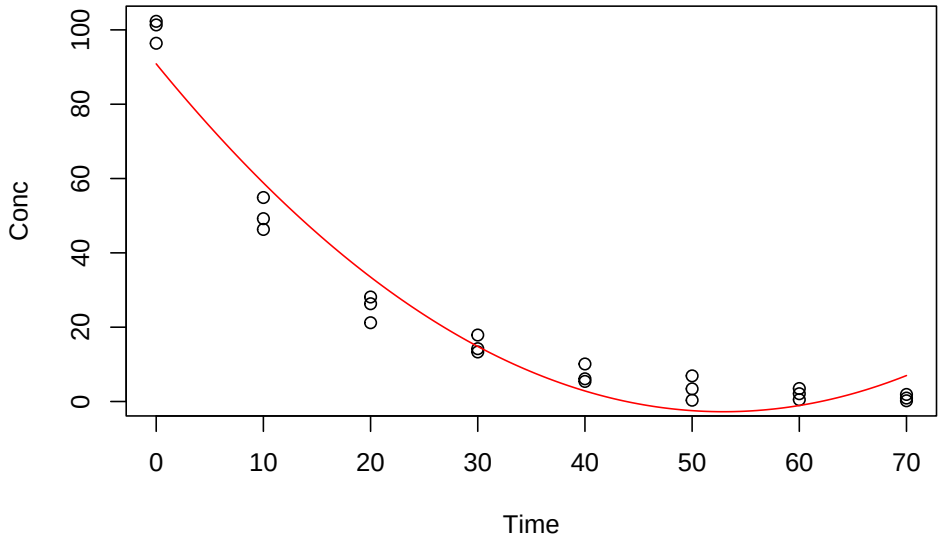

Figure 14.5: Approximating a degradation kinetic by using a second order polynomial

```
plot(Conc ~ Time, data=dataset)
lines(pred ~ seq(0, 70, by = 0.1), col = "red")
```

We see that the approximation is good up to 40 days after the beginning of the experiment. Later on, the fitted function appears to deviate from the observed pattern, depicting an unrealistic concentration increase from 55 days onwards (Figure 14.5). Clearly, the use of polynomials may be useful in some other instances, but it not a suitable solution for this case.

14.3.3 Nonlinear least squares

The third possibility consists of using nonlinear least squares, based on the Gauss-Newton iterative algorithm. We need to provide reasonable initial estimates of model parameters and the algorithm updates them, iteratively, until it converges on the approximate least squares solution. Nowadays, thanks to the advent of modern computers, nonlinear least squares have become the most widespread nonlinear regression method, providing very good approximations for most practical needs.

14.4 Nonlinear regression with R

Considering the base R installation, nonlinear least squares regression is implemented in the `nls()` function. The syntax is very similar to that of the `lm()` function, although we need to provide reasonable initial values for model parameters. In this case, obtaining such values is relatively easy: a is the initial concentration and, by looking at the data, we can set this value to 100. The parameter k is the constant degradation rate, i.e. the percentage daily reduction in concentration; we see that the concentration drops by 50% in ten days, thus we could assume that there is a daily 5% drop (k = 0.05). With such an information, we fit the model, as shown in the box below.

```
modNlin <- nls(Conc ~ A*exp(-k*Time),
               start=list(A=100, k=0.05),
               data = dataset)
summary(modNlin)
```

```
##
## Formula: Conc ~ A * exp(-k * Time)
##
## Parameters:
##    Estimate Std. Error t value Pr(>|t|)
## A 99.634902   1.461047   68.19   <2e-16 ***
## k  0.067039   0.001887   35.53   <2e-16 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.621 on 22 degrees of freedom
##
## Number of iterations to convergence: 5
## Achieved convergence tolerance: 4.33e-07
```

Instead of coding the mean model from the scratch, we can use one of the available self-starting functions (Fig. 14.3), which are associated to self-starting routines. This is very useful, as the self-starting functions do not need initial estimates and, thus, we are free from the task of providing them, which is rather difficult for beginners. Indeed, if our initial guesses are not close enough to least squares estimates, the algorithm may freeze during the estimation process and may not reach convergence. Self-starting functions are available within the 'aomisc' which needs to be installed from github prior to fitting the model.

```
# Installation is required only once
# library(devtools)
# install_github("onofriandreapg/aomisc")
library(aomisc)
modNlin2 <- nls(Conc ~ NLS.expoDecay(Time, a, k),
                data = dataset)
```

14.5 Checking the model

While the estimation of model parameters is performed by using numerical methods, checking a fitted model is done by using the very same methods as shown for simple linear regression, with slight differences. In the following section we will show some of the R methods for nonlinear models, which can be used on 'nls' objects.

14.5.1 Graphical analyses of residuals

First of all, we check the basic assumptions of normality and homoscedasticity of model residuals, by the usual diagnostic plots. We can us the `plotnls()` function, as available in the 'aomisc' package.

```
par(mfrow=c(1,2))
plotnls(modNlin, which = 1)
plotnls(modNlin, which = 2)
```

Figure 14.6 does not show any visible deviations and, thus, we proceed to plotting the observed data along with model predictions (Figure 14.7), e.g., by using the `plotnls()` function, in the 'aomisc' package.

```
plotnls(modNlin, type = "means",
        xlab = "Time (d)", ylab = "Concentration (ng/g)")
```

14.5.2 Approximate F test for lack of fit

If we have replicates, we can fit an ANOVA model and compare this latter model to the nonlinear regression model, as we have already shown in Chapter 13, for linear regression models (F test for lack of fit). With R, this test can be performed by using the `anova()` method and passing the two model objects

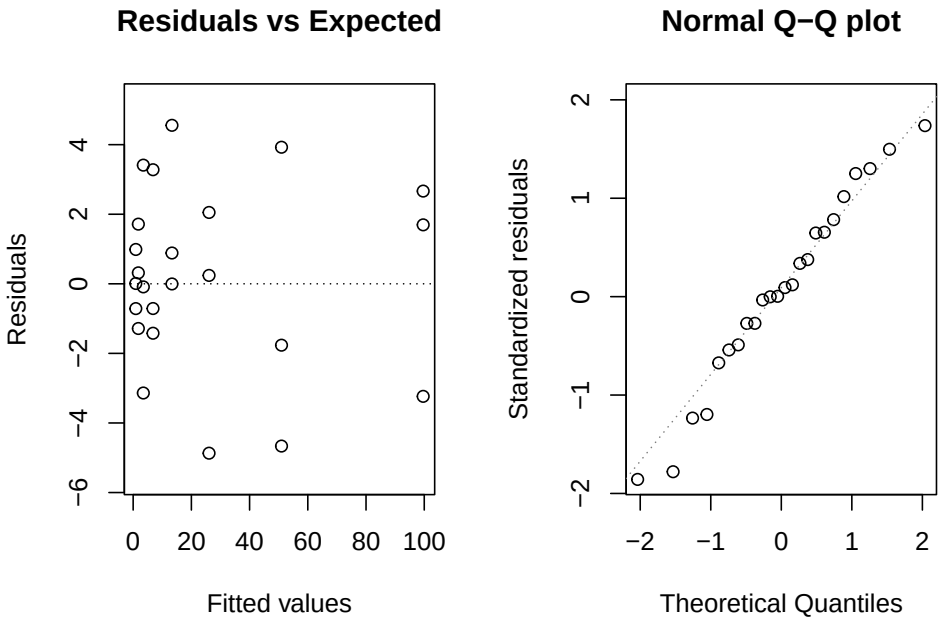

Figure 14.6: Graphical analyses of residuals relating to the degradation of metamitron in soil.

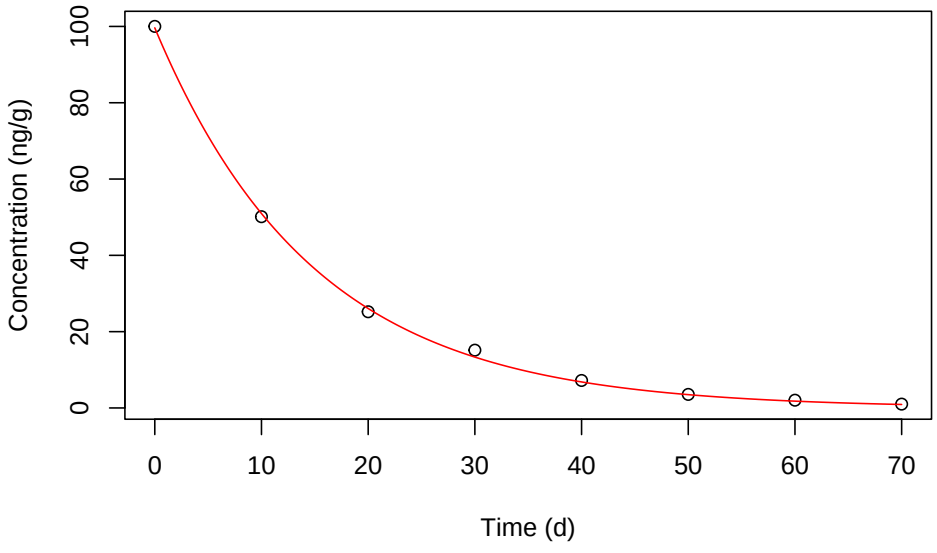

Figure 14.7: Degradation kinetic for metamitron in soil: symbols show the observed data, the red line shows the fitted equation

as arguments. The box below shows that the null hypothesis of no lack of fit can be accepted.

```
modAov <- lm(Conc ~ factor(Time), data=dataset)
anova(modNlin, modAov)
```

```
## Analysis of Variance Table
##
## Model 1: Conc ~ A * exp(-k * Time)
## Model 2: Conc ~ factor(Time)
##   Res.Df Res.Sum Sq Df Sum Sq F value Pr(>F)
## 1     22     151.18
## 2     16     135.94  6 15.238  0.2989 0.9284
```

14.5.3 The coefficient of determination (R^2)

The R^2 coefficient in nonlinear regression should not be used as a measure of goodness of fit, because it represents the ratio between the residual deviances for the model under comparison and for a model with the only intercept (see previous chapter). Some non-linear models do not have an intercept and, therefore, the R^2 value is not meaningful and can also assume values outside the range from 0 to 1.

Such an argument can be, at least partly, overcame by using the **Pseudo R^2** (Schabenberger and Pierce, 2002), that is the proportion of variance explained by the regression effect:

$$R_a^2 = 1 - \frac{MSE}{MST}$$

where MSE is the residual mean square and MST is total mean square. For our example:

```
MSE <- summary(modNlin)$sigma ^ 2
MST <- var(dataset$Conc)
1 - MSE/MST
```

```
## [1] 0.9936359
```

Whenever necessary, The Pseudo-R^2 value can be obtained by using the `R2nls()` function, in the 'aomisc' package, as shown in the box below.

```
R2nls(modNlin)$PseudoR2
```

```
## [1] 0.9939126
```

14.6 Stabilising transformations

In some cases, it may happen that the model does not fit the data (**lack of fit**) or that the residuals are not gaussian and homoscedastic. In the first case, we have to discard the model and select a new one, that fits better to the observed data.

In the other cases (heteroscedastic or non-normal residuals), similarly to linear regression, we should adopt some sort of stabilising transformations, possibly selected by using the Box-Cox method. However, nonlinear regression poses an additional issue, that we should keep into account. Indeed, in most cases, the parameters of nonlinear models are characterised by a clear biological meaning, such as, for example, the parameter a in the exponential equation above, that is the initial concentration value. If we transform the response into, e.g., its logarithm, the estimated a represents the logarithm of the initial concentration and its biological meaning is lost.

In order to avoid such a problem and obtain parameter estimates in their original scale, the 'transform-both-sides' approach is feasible, where we use the Box-Cox family of transformations (see Chapter 8) to transform both the response and the model:

$$Y_i^\lambda = f(X_i)^\lambda + \varepsilon_i$$

In order to use this technique in R, we can use the `boxcox.nls()` method in the 'aomisc' package, that is very similar to the 'boxcox()' method in the MASS package (see Chapter 8). The `boxcox.nls()` method returns the maximum likelihood value for λ with confidence intervals and a likelihood graph as a function of λ can also be obtained by using the argument 'plotit = T'

```
bc <- boxcox(modNlin)
bc$lambda
```

```
## $lambda
## [1] 0.8
```

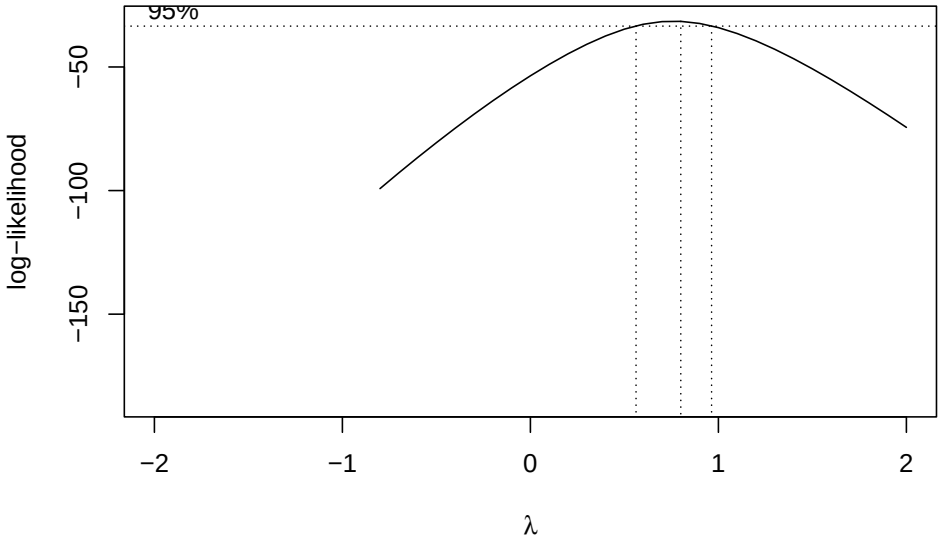

Figure 14.8: Selection of the 'lambda' value for the Box-Cox transformation of a nonlinear regression model

```
##
## $ci
## [1] 0.5619080 0.9637162
```

```
boxcox(modNlin, plotit = T)
```

From Figure 14.8, we see that the transformation is not required, although a value $\lambda = 0.5$ (square-root transformation; see previous chapter) might be used to maximise the likelihood of the data under the selected model. We can specify our favourite λ value and get the corresponding fit by using the same `boxcox.nsl()` function and passing an extra-argument, as shown in the box below.

```
modNlin3 <- boxcox(modNlin, lambda = 0.5)
summary(modNlin3)
```

```
##
## Formula: bcFct1(Conc) ~ bcFct2(A * exp(-k * Time))
##
## Parameters:
##    Estimate Std. Error t value Pr(>|t|)
## A 99.532761   4.625343   21.52 2.87e-16 ***
## k  0.067068   0.002749   24.40  < 2e-16 ***
```

```
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9196 on 22 degrees of freedom
##
## Number of iterations to convergence: 1
## Achieved convergence tolerance: 7.541e-06
```

We see that, in spite of the stabilising transformation, the estimated parameters have not lost their measurement unit.

14.7 Making predictions

As shown for linear regression in Chapter 13, every fitted model can be used to make predictions, i.e. to calculate the response for a given X-value or to calculate the X-value corresponding to a given response.

Analogously to linear regression, we could make predictions by using the `predict()` method, although, for nonlinear regression objects, this method does not return standard errors. Therefore, we can use the `gnlht()` function in the 'aomisc' package, which requires the following arguments:

1. a list of functions containing model parameters (with the same names as the fitted model) and, possibly, some other parameters
2. if the previous function contain further parameters with respect to the fitted model, their values need to be given in a data frame as an extra-argument

In the box below we make predictions for the response at 5, 10 and 15 days after the beginning of the experiment.

```
func <- list(~A * exp(-k * time))
const <- data.frame(time = c(5, 10, 15))
gnlht(modNlin, func,  const)
```

```
##                  Form time Estimate        SE  t-value
## 1 A * exp(-k * time)    5 71.25873 0.9505532 74.96553
## 2 A * exp(-k * time)   10 50.96413 0.9100611 56.00078
## 3 A * exp(-k * time)   15 36.44947 0.9205315 39.59611
##        p-value
```

```
## 1 5.340741e-28
## 2 3.163518e-25
## 3 6.029672e-22
```

The same method can be used to make inverse predictions. In our case, the inverse function is:

$$X = -\frac{log\left(\frac{Y}{A}\right)}{k}$$

We can calculate the times required for the concentration to drop to, e.g., 10 and 20 mg/g by using the code in the following box.

```
func <- list(~ -(log(Conc/A)/k))
const <- data.frame(Conc = c(10, 20))
gnlht(modNlin, func,  const)
```

```
##               Form Conc Estimate        SE  t-value
## 1 -(log(Conc/A)/k)   10 34.29237 0.8871429 38.65484
## 2 -(log(Conc/A)/k)   20 23.95291 0.6065930 39.48761
##        p-value
## 1 1.016347e-21
## 2 6.399715e-22
```

In the same fashion, we can calculate the half-life (i.e. the time required for the concentration to drop to half of the initial value), by considering that:

$$t_{1/2} = -\frac{\log\left(\frac{1}{2}\right)}{k}$$

The code is::

```
func <- list(~ -(log(0.5)/k))
gnlht(modNlin, func)
```

```
##            Form Estimate       SE t-value      p-value
## 1 -(log(0.5)/k) 10.33945 0.291017 35.5287 6.318214e-21
```

The 'gnlht()' function provides standard errors based on the delta method, which we have already introduced in a previous chapter.

14.8 Further readings

1. Bates, D.M., Watts, D.G., 1988. Nonlinear regression analysis & its applications. John Wiley & Sons, Inc., Books.
2. Bolker, B.M., 2008. Ecological models and data in R. Princeton University Press, Books.
3. Carroll, R.J., Ruppert, D., 1988. Transformation and weighting in regression. Chapman and Hall, Books.
4. Ratkowsky, D.A., 1990. Handbook of nonlinear regression models. Marcel Dekker Inc., Books.
5. Ritz, C., Streibig, J.C., 2008. Nonlinear regression with R. Springer-Verlag New York Inc., Books.
6. Schabenberger, O., Pierce, F.J., 2002. Contemporary statistical models for the plant and soil sciences. Taylor & Francis, CRC Press

Chapter 15

Exercises

Placeholder

15.1 Chapter 3

15.1.1 Exercise 1

15.1.2 Exercise 2

15.1.3 Exercise 3

15.2 Chapter 4

15.2.1 Exercise 1

15.2.2 Exercise 2

15.2.3 Exercise 3

15.2.4 Exercise 4

15.2.5 Exercise 5

15.3 Chapter 5

15.3.1 Exercise 1

15.3.2 Exercise 2

15.3.3 Exercise 3

15.3.4 Exercise 4

15.4 Chapter 6

15.4.1 Exercise 1

15.4.2 Exercise 2

15.4.3 Exercise 3

15.4.4 Exercise 4

15.4.5 Exercise 5

Chapter 16

APPENDIX: A very gentle introduction to R

Placeholder