José C. Pinheiro
Douglas M. Bates

Mixed-Effects Models in S and S-PLUS

Springer

José C. Pinheiro
Department of Biostatistics
Novartis Pharmaceuticals
One Health Plaza
East Hanover, NJ 07936-1080
USA
jose.pinheiro@pharma.novartis.com

Douglas M. Bates
Department of Statistics
University of Wisconsin
Madison, WI 53706-1685
USA
bates@stat.wisc.edu

Library of Congress Cataloging-in-Publication Data
Pinheiro, José C.
Mixed-effects models in S and S-PLUS/José C. Pinheiro, Douglas M. Bates
p. cm.—(Statistics and computing)
Includes bibliographical references and index.
ISBN 0-387-98957-9 (alk. paper)
I. Bates, Douglas M. II. Title. III. Series.
QA76.73.S15P56 2000
005.13'3—dc21 99-053566

Printed on acid-free paper.

Printed in the United States of America. (HAM/BP)

9 8 7

springer.com

8

Fitting Nonlinear Mixed-Effects Models

As shown in the examples in Chapter 6, nonlinear mixed-effects models offer a flexible tool for analyzing grouped data with models that depend nonlinearly upon their parameters. As nonlinear models are usually based on a *mechanistic* model of the relationship between the response and the covariates, their parameters can have a theoretical interpretation and are often of interest in their own right. In this chapter, we describe in detail the facilities in the nlme library for fitting nonlinear mixed-effects models.

The first section presents a brief review of the standard nonlinear regression function in S, `nls`, to illustrate the use of nonlinear formulas and the derivation of starting estimates for the model parameters. We describe the selfStart class of nonlinear model functions that can calculate initial values for their parameters from the data. The `nlsList` function, which produces separate `nls` fits for each level of a grouping variable, is described and illustrated through examples.

Section 8.2 describes the `nlme` function for fitting nonlinear mixed-effects models with single or multiple levels of nesting. Method functions for displaying, plotting, updating, making predictions, and obtaining confidence intervals from a fitted nlme object are described and illustrated with examples.

The use of correlation structures and variance functions to extend the basic nonlinear mixed-effects model is discussed in Section 8.3. The `gnls` function for fitting the extended nonlinear regression model without random effects, presented in §7.5.1, is described, together with its associated methods.

8.1 Fitting Nonlinear Models in S with `nls` and `nlsList`

In this section we describe the use of the nonlinear least squares (`nls`) function for fitting a single nonlinear regression model and the `nlsList` function for fitting a set of nonlinear regression models to grouped data. Especially with `nlsList`, it is very helpful to have the model function itself defined as a *self-starting model*, as described in §8.1.2.

8.1.1 Using the `nls` Function

The S function `nls` uses a Gauss–Newton algorithm, described in §7.5.2, to determine the *nonlinear least squares* estimates of the parameters in a nonlinear regression model. A typical call to `nls` is of the form

```
nls( formula, data, start )
```

where `formula` is a two-sided nonlinear formula specifying the model, `data` is a data frame with the variables used in `formula`, and `start` is a named vector or list containing the starting estimates for the model parameters. Several other arguments to `nls` are available, as described in Bates and Chambers (1992) and Venables and Ripley (1999, Chapter 8).

We illustrate the use of `nls` to fit nonlinear regression models with the `Orange` data from a study of the growth of orange trees, reported in Draper and Smith (1998, Exercise 24.N, p. 559). The data, shown in Figure 8.1 and described in Appendix A.16, are the trunk circumferences (in millimeters) of each of five trees at each of seven occasions.

```
> ## outer = ~1 is used to display all five curves in one panel
> plot( Orange, outer = ~1 )                              # Figure 8.1
```

Because all trees are measured on the same occasions these are balanced, longitudinal data. It is clear from Figure 8.1 that a "tree effect" is present, and we will take this into account when fitting `nlme` or `nlsList` models. To illustrate some of the details of fitting `nls` models, we will temporarily ignore the tree effect and fit a single logistic model to all the data. Recall from §6.1 that this model expresses the trunk circumference y_{ij} of tree i at age x_{ij} for $i = 1, \ldots, 5$ $j = 1, \ldots, 7$ as

$$y_{ij} = \frac{\phi_1}{1 + \exp\left[-\left(t_{ij} - \phi_2\right)/\phi_3\right]} + \epsilon_{ij}, \tag{8.1}$$

where the error terms ϵ_{ij} are assumed to be distributed independently as $\mathcal{N}(0, \sigma^2)$. As explained in §6.1, the model parameters are the asymptotic trunk circumference ϕ_1, the age at which the tree attains half of its asymptotic trunk circumference ϕ_2, and the growth scale ϕ_3. This function is nonlinear in ϕ_2 and ϕ_3.

Model (8.1) can be represented in S by the nonlinear formula

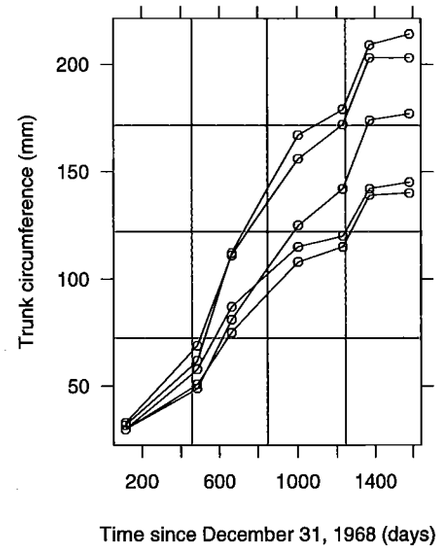

FIGURE 8.1. Circumference of five orange trees from a grove in southern California over time. The measurement is probably the "circumference at breast height" commonly used by foresters. Points corresponding to the same tree are connected by lines.

```
circumference ~ Asym/(1 + exp(-(age - xmid)/scal)),
```

where `Asym` $= \phi_1$, `xmid` $= \phi_2$, and `scal` $= \phi_3$. Unlike in the linear case, the parameters must be declared explicitly in a nonlinear model formula and an intercept is not assumed by default.

An alternative approach is to write an S function representing the logistic model as, say,

```
> logist <-
+    function(x, Asym, xmid, scal) Asym/(1 + exp(-(x - xmid)/scal))
```

and then use it in the nonlinear formula

```
circumference ~ logist(age, Asym, xmid, scal)
```

An advantage of this latter approach is that we can modify our `logist` function to include a `gradient` attribute with its returned value. This would then be used as the gradient matrix in the Gauss–Newton nonlinear least-squares optimization, increasing the numerical stability and the rate of convergence of the algorithm, compared to the default use of numerical derivatives. The `deriv` function can be used to produce a function that returns a `gradient` attribute with its value.

```
> logist <- deriv( ~Asym/(1+exp(-(x-xmid)/scal)),
```

```
+     c("Asym", "xmid", "scal"), function(x, Asym, xmid, scal){} )
> Asym <- 180; xmid <- 700; scal <- 300
> logist( Orange$age[1:7], Asym, xmid, scal )
[1]  22.617  58.931  84.606 132.061 153.802 162.681 170.962
attr(, "gradient"):
        Asym      xmid      scal
[1,] 0.12565 -0.065916  0.127878
[2,] 0.32739 -0.132124  0.095129
[3,] 0.47004 -0.149461  0.017935
[4,] 0.73367 -0.117238 -0.118802
[5,] 0.85446 -0.074616 -0.132070
[6,] 0.90378 -0.052175 -0.116872
[7,] 0.94979 -0.028614 -0.084125
```

As mentioned in §6.1, one important difference between linear and nonlinear regression is that the nonlinear models require starting estimates for the parameters. Determining reasonable starting estimates for a nonlinear regression problem is something of an art, but some general recommendations are available (Bates and Watts, 1988, §3.2). We return to this issue in §8.1.2, where we describe the selfStart class of model functions.

Because the parameters in the logistic model (8.1) have a graphical interpretation, we can determine initial estimates from a plot of the data. In Figure 8.1 it appears that the mean asymptotic trunk circumference is around 170 mm and that the trees attain half of their asymptotic trunk circumference at about 700 days of age. Therefore, we use the inital estimates of $\tilde{\phi}_1 = 170$ for the asymptotic trunk circumference and $\tilde{\phi}_2 = 700$ for the location of the inflection point. To obtain an initial estimate for ϕ_3, we note that the logistic curve reaches approximately 3/4 of its asymptotic value when $x = \phi_2 + \phi_3$. Inspection of Figure 8.1 suggests that the trees attain 3/4 of their final trunk circumference at about 1200 days, giving an intial estimate of $\tilde{\phi}_3 = 500$.

We combine all this information in the following call to `nls`

```
> fm1Oran.nls <- nls(circumference ~ logist(age, Asym, xmid, scal),
+     data = Orange, start = c(Asym = 170, xmid = 700, scal = 500) )
```

Our initial estimates are reasonable and the call converges. Following the usual framework for modeling functions in S, the object `fm1Oran.nls` produced by the call to `nls` is of class nls, for which several methods are available to display, plot, update, and extract components from a fitted object. For example, the `summary` method provides information about the estimated parameters.

```
> summary( fm1Oran.nls )
Formula: circumference ~ logist(age, Asym, xmid, scal)
Parameters:
     Value Std. Error t value
Asym 192.68     20.239  9.5203
xmid 728.71    107.272  6.7931
scal 353.49     81.460  4.3395

Residual standard error: 23.3721 on 32 degrees of freedom

Correlation of Parameter Estimates:
     Asym  xmid
xmid 0.922
scal 0.869 0.770
```


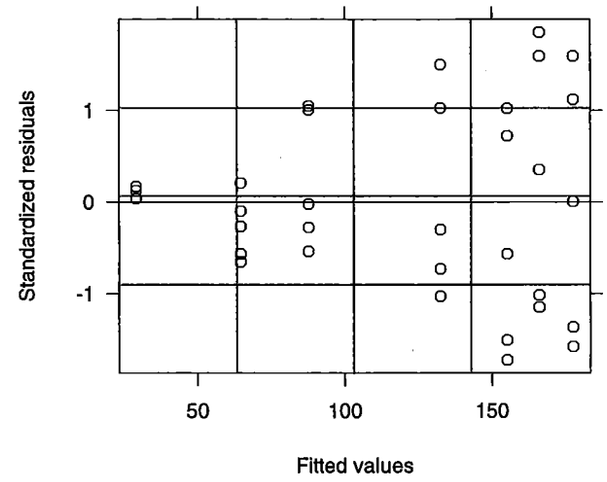

FIGURE 8.2. Scatter plots of standardized residuals versus fitted values for `fm1Oran.nls`, a nonlinear least squares fit of the logistic growth model to the entire orange tree data set.

The final estimates are close to the initial values derived from Figure 8.1. The standard errors for the parameter estimates are relatively large, suggesting that there is considerable variability in the data.

The `plot` method for nls objects (which is included with the nlme library), has a syntax similar to the `lme` and `gls` plot methods described in §4.3.1 and §5.4. By default, the plot of the standardized residuals versus fitted values, shown in Figure 8.2, is produced.

```
> plot( fm1Oran.nls )                                    # Figure 8.2
```

The variability in the residuals increases with the fitted values, but, in this case, the wedge-shaped pattern is due to the correlation among observations in the same tree and not to heteroscedastic errors. We can get a better understanding of the problem by looking at the plot of the residuals by tree presented in Figure 8.3.

```
> plot(fm1Oran.nls, Tree ~ resid(.), abline = 0)         # Figure 8.3
```

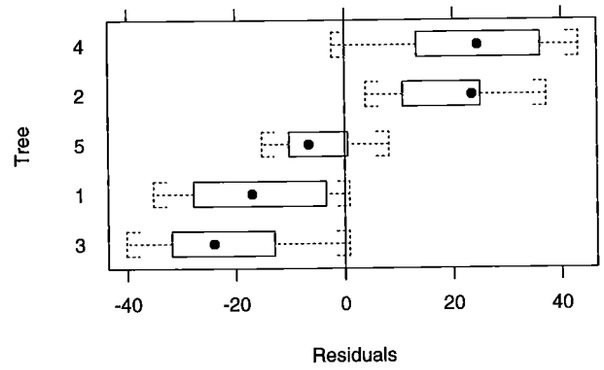

FIGURE 8.3. Boxplots of residuals by tree for `fm1Oran.nls`, a nonlinear least squares fit of the logistic growth model to the entire orange tree data set.

The residuals are mostly negative for trees 1 and 3 and mostly positive for trees 2 and 4, giving strong evidence that a "tree effect" should be included in the model.

A simple approach to account for a tree effect is to allow different parameters for each tree, resulting in separate `nls` fits. This is the approach used in the `nlsList` function, described in §8.1.3, which provides a valuable tool for model building, but usually produces overparametrized models. As illustrated in Chapter 6, nonlinear mixed-effects models strike a balance between the simple `nls` model and the overparametrized `nlsList` model, by allowing random effects to account for among-group variation in some of the parameters, while preserving a moderate number of parameters in the model.

8.1.2 Self-Starting Nonlinear Model Functions

Bates and Watts (1988, §3.2) describe several techniques for determining starting estimates in a nonlinear regression. Some of the more effective techniques are:

- Take advantage of partially linear models, as described in Bates and Chambers (1992, §10.2.5), so that initial estimates are needed only for those parameters that enter the model nonlinearly.
- Choose parameters that have meaningful graphical interpretations, as we did for the logistic model.
- Refine the estimates of some of the parameters by iterating on them while holding all the other parameters fixed at their current values.

The application of these techniques to a particular nonlinear regression model applied to a given set of data can be straightforward but tedious.

Especially when the same model will be applied to several similar sets of data, as is the case in many of the examples we consider here, we do not want to have to manually repeat the same series of steps in determining starting estimates. A more sensible approach is to encapsulate the steps used to derive initial estimates for a given nonlinear model into a function that can be used to generate intial estimates from any dataset. Self-starting nonlinear regression models are S functions that contain an auxillary function to calculate the initial parameter estimates. They are represented in S as selfStart objects.

The S objects of the selfStart class are composed of two functions; one that evaluates the nonlinear regression model itself and an auxillary function, called the initial attribute, that determines starting estimates for the model's parameters from a set of data. When a selfStart object for a model is available, there is no need to determine starting values for the parameters. The user can simply specify the formula for the model and the data to which it should be applied. From a user's point of view, fitting self-starting nonlinear regression models is nearly as easy as fitting linear regression models.

We illustrate the construction and use of self-starting models by building a function for the logistic model. The basic steps in the calculation of initial estimates for the logistic model (8.1) from the `Orange` dataset are:

1. *Sort/average*: sort the data according to the `x` variable and obtain the average response `y` for each unique `x`.
2. *Asymptote*: use the maximum `y` as an initial value $\tilde{\phi}_1$ for the asymptote.
3. *Inflection point*: use the `x` corresponding to $0.5\tilde{\phi}_1$ as an initial value $\tilde{\phi}_2$ for the inflection point.
4. *Scale*: use the difference between the `x` corresponding to $0.75\tilde{\phi}_1$ and $\tilde{\phi}_2$ as an initial value $\tilde{\phi}_3$ for the growth scale

Step 1, *sort/average*, was carried out implicitly in our graphical derivation of initial values for the orange trees example, but is now explicitly incorporated in the algorithm.

Two auxillary functions, `sortedXyData` and `NLSstClosestX`, included in the nlme library, are particularly useful for constructing self-starting models. The `sortedXyData` function performs the *sort/average* step. It takes the arguments `x`, `y`, and `data` and returns a `data.frame` with two columns: `y`, the average `y` for each unique value of `x`, and `x`, the unique values of `x`, sorted in ascending order. The arguments `x` and `y` can be numeric vectors or they can be expressions or strings to be evaluated in `data`. The returned object is of class sortedXyData. For example, the pointwise averages of the growth curves of the orange trees are obtained with

```
> Orange.sortAvg <- sortedXyData( "age", "circumference", Orange )
> Orange.sortAvg
     x     y
1  118  31.0
2  484  57.8
3  664  93.2
4 1004 134.2
5 1231 145.6
6 1372 173.4
7 1582 175.8
```

The `NLSstClosestX` function estimates the value of `x` corresponding to a given `y`, using linear interpolation. It takes two arguments: `xy`, a sortedXyData object, and `yval`, the desired value of `y`, and returns a numeric value. For example, the estimated age at which the average growth curve reaches 130 mm is

```
> NLSstClosestX(Orange.sortAvg, 130)
[1] 969.17
```

A function to serve as an `initial` attribute of a selfStart function must be defined with the arguments `mCall`, `LHS`, and `data`, in that order, and must return the initial values as a vector, or list, with elements named according to the actual parameters in the model function's call. The first argument, `mCall`, is a *matched call* to the `selfStart` model. It contains the arguments of the function call (as expressions), matched with the formal parameters, which are the argument names used in the original definition of the model. In the case of the `logist` function, defined before as

```
function(x, Asym, xmid, scal)
```

the formal parameters are `x`, `Asym`, `xmid`, and `scal`. If the function is called as

```
logist(age, A, xmid, scal)
```

then `mCall` will have components `x = age`, `Asym = A`, `xmid = xmid` and `scal = scal`. As described above, the names of these components are the formal parameters in the model function definition. The values of these components are the names (or, more generally, the expressions) that are the actual arguments in the call to the model function.

The `LHS` argument is the expression on the left-hand side of the model formula in the call to `nls`. It determines the response vector. The `data` argument gives a `data.frame` in which to find the variables named in the other two arguments. The function `logistInit` below implements a slightly more general version of the algorithm described above for calculating initial estimates in the logistic model, using the required syntax.


```
> logistInit
function(mCall, LHS, data)
{
  xy <- sortedXyData(mCall[["x"]], LHS, data)
  if(nrow(xy) < 3) {
    stop("Too few distinct input values to fit a logistic")
  }
  Asym <- max(abs(xy[,"y"]))
  if (Asym != max(xy[,"y"])) Asym <- -Asym  # negative asymptote
  xmid <- NLSstClosestX(xy, 0.5 * Asym)
  scal <- NLSstClosestX(xy, 0.75 * Asym) - xmid
  value <- c(Asym, xmid, scal)
  names(value) <- mCall[c("Asym", "xmid", "scal")]
  value
}
```

The algorithm in `logistInit` includes a check for an adequate number of distinct observations to fit a logistic model and allows negative as well as positive asymptotes.

Before the starting values are returned, we ensure that the components are named according to the actual parameters in the model function call. As described above, the object `mCall` provides the correspondence between the formal parameter names and the actual parameter names, so we assign the names indirectly through `mCall`. This step is important. All functions to be used as the `initial` attribute of a selfStart model should indirectly assign the names to the result in this way.

The `selfStart` constructor is used to create a self-starting model. It can be called with two functions, the model function itself and the `initial` attribute, as in

```
> logist <- selfStart( logist, initial = logistInit )
> class( logist )
[1] "selfStart"
```

Alternatively, it can be called with a one-sided formula defining the nonlinear model, the function for the `initial` attribute, and a character vector giving the parameter names.

```
> logist <- selfStart( ~ Asym/(1 + exp(-(x - xmid)/scal)),
+    initial = logistInit, parameters = c("Asym", "xmid", "scal"))
```

When `selfStart` is called like this, the model function is produced by applying `deriv` to the right-hand side of the model formula.

The `getInitial` function is used to extract the initial parameter estimates from a given dataset when using a selfStart model function. It takes two arguments: a two-sided model formula, which must include a selfStart function on its right-hand side, and a `data.frame` in which to evaluate the variables in the model formula.

TABLE 8.1. Standard selfStart functions in the NLME 3.0 distribution.

SSasymp	asymptotic regression
SSasympOff	asymptotic regression with an offset
SSasympOrig	asymptotic regression through the origin
SSbiexp	biexponential
SSfol	first-order compartment
SSfpl	four-parameter logistic
SSlogis	logistic
SSmicmen	Michaelis–Menten

```
> getInitial(circumference ~ logist(age, Asym, xmid, scal), Orange)
  Asym   xmid   scal
 175.8 637.05 347.46
```

As expected, the initial values produced by `logist` are similar to those obtained previously using the graphical interpretation of the parameters and Figure 8.1.

When `nls` is called without initial values for the parameters and a selfStart model function is provided, `nls` calls `getInitial` to provide the initial values. In this case, only the model formula and the data are required to fit the model, making the call nearly as simple as an `lm` call. For example, the logistic model can be fit to the orange tree data with

```
> nls( circumference ~ logist(age, Asym, xmid, scal), Orange )
Residual sum of squares : 17480
parameters:
   Asym   xmid   scal
 192.69 728.77 353.54
formula: circumference ~ logist(age, Asym, xmid, scal)
35 observations
```

The nlme library includes several self-starting model functions that can be used to fit nonlinear regression models in S without specifying starting estimates for the parameters. They are listed in Table 8.1 and are described in detail in Appendix B. The `SSlogis` function is a more sophisticated version of our simple `logist` self-starting model, but with the same argument sequence. It uses several techniques, such as the algorithm for partially linear models, to refine the starting estimates so the returned values are actually the converged estimates.

```
> getInitial(circumference ~ SSlogis(age,Asym,xmid,scal), Orange)
   Asym   xmid  scal
 192.68 728.72 353.5
> nls( circumference ~ SSlogis(age, Asym, xmid, scal), Orange )
Residual sum of squares : 17480
parameters:
   Asym   xmid  scal
 192.68 728.72 353.5
formula: circumference ~ SSlogis(age, Asym, xmid, scal)
35 observations
```


We can see that selfStart model objects relieve the user of much of the effort required for a nonlinear regression analysis. If a versatile, effective strategy for determining starting estimates is represented carefully in the object, it can make the use of nonlinear models nearly as simple as the use of linear models. If you frequently use a nonlinear model not included in Table 8.1, you should consider writing your own self-starting model to represent it. The selfStart functions in Table 8.1 and the `logist` function described in this section can be used as templates.

8.1.3 *Separate Nonlinear Fits by Group: The `nlsList` Function*

In the indomethacin and soybean examples of Chapter 6 we saw how the `nlsList` function can be used to produce separate fits of a nonlinear model for each group in a groupedData object. These separate fits by group are a powerful tool for model building with nonlinear mixed-effects models because the individual estimates can suggest the type of random-effects structure to use. Also, these estimates provide starting values for the parameters in the mixed-effects model. In this section we provide more detail on the use of `nlsList` function itself and the methods for the nlsList objects that it creates.

The `nls` fits performed within `nlsList` will require starting estimates for the parameters. Although it is sometimes possible to use a single set of starting estimates for all the groups, we recommend using a `selfStart` function, as described in §8.1.2, to automatically generate individual initial estimates for each group.

A typical call to `nlsList` is

```
nlsList( model, data )
```

where `model` is a two-sided formula whose right-hand side consists of two parts separated by the | operator. The first part defines the nonlinear model, generally involving a selfStart function to be fitted to each subset of `data`, and the second part specifies the grouping factor. Any nonlinear model formula allowed in `nls` can also be used with `nlsList`. The `data` argument gives a data frame in which the variables in `model` can be evaluated. The grouping factor can be omitted from the `model` formula when `data` is a `groupedData` object.

To illustrate the use of `nlsList`, let us continue with the analysis of the orange trees data, fitting a separate logistic curve to each tree and using the selfStart function `SSlogis` to produce initial estimates.

```
> fm1Oran.lis <-
+    nlsList(circumference ~ SSlogis(age, Asym, xmid, scal) | Tree,
+            data = Orange)
```

Because `Orange` is a groupedData object, the grouping factor `Tree` could have been omitted from the model formula. When a `selfStart` function depends on only one covariate, as does `SSlogis`, and `data` is a groupedData object, the entire form of the model formula can be inferred from the display formula stored with the groupedData object. In this case, only the selfStart function and the groupedData object need to be passed to the `nlsList` function. For example, we could use

```
> fm1Oran.lis <- nlsList( SSlogis, Orange )
```

to obtain the same `nlsList` fit as before.

The `nlsList` function can also be used with regular, non-self-starting nonlinear functions, but in this case the same set of starting values, specified in the `start` argument, will be used for every group. The use of common starting estimates may not be a sensible choice in many applications, so we strongly encourage the use of `selfStart` functions with `nlsList`. In the orange trees example, the individual growth patterns are similar enough that a common set of starting estimates can be used. For example, using the same initial estimates as used to fit `fm1Oran.nls` in §8.1.1, we have

```
> fm1Oran.lis.noSS <-
+    nlsList( circumference ~ Asym/(1+exp(-(age-xmid)/scal)),
+             data = Orange,
+             start = c(Asym = 170, xmid = 700, scal = 500) )
```

Because the model fits `fm1Oran.lis` and `fm1Oran.lis.noSS` are derived from different starting values, the parameter estimates will differ slightly.

Objects returned by `nlsList` are of class nlsList which inherits from class lmList. Therefore, all summary and display methods, as well as methods for extracting components from the fitted object, for class lmList can also be applied to an nlsList object. Table 8.2 lists the most commonly used methods for nlsList objects. We illustrate the use of some of these methods below.

The `print` method gives minimal information about the individual `nls` fits.

```
> fm1Oran.lis
 Call:
  Model: circumference ~ SSlogis(age, Asym, xmid, scal) | Tree
   Data: Orange

Coefficients:
    Asym   xmid   scal
3 158.83 734.85 400.95
1 154.15 627.12 362.50
5 207.27 861.35 379.99
2 218.99 700.32 332.47
4 225.30 710.69 303.13

Degrees of freedom: 35 total; 20 residual
Residual standard error: 7.98
```


TABLE 8.2. Main nlsList methods.

`augPred`	predictions augmented with observed values
`coef`	coefficients from individual `nls` fits
`fitted`	fitted values from individual `nls` fits
`fixef`	average of individual `nls` coefficients
`intervals`	confidence intervals on coefficients
`nlme`	nonlinear mixed-effects model from `nlsList` fit
`logLik`	sum of individual `nls` log-likelihoods
`pairs`	scatter-plot matrix of coefficients or random effects
`plot`	diagnostic Trellis plots
`predict`	predictions for individual `nls` fits
`print`	brief information about the `lm` fits
`qqnorm`	normal probability plots
`ranef`	deviations of coefficients from average
`resid`	residuals from individual `nls` fits
`summary`	more detailed information about `nls` fits
`update`	update the individual `nls` fits

More detailed information on the coefficients is obtained with `summary`

```
> summary( fm1Oran.lis )
. . .
Coefficients:
   Asym
   Value Std. Error t value
3 158.83     19.235  8.2574
1 154.15     13.588 11.3446
5 207.27     22.028  9.4092
2 218.99     13.357 16.3959
4 225.30     11.838 19.0318
   xmid
   Value Std. Error t value
3 734.85    130.807  5.6178
1 627.12     92.854  6.7538
5 861.35    108.017  7.9742
2 700.32     61.342 11.4167
4 710.69     51.166 13.8899
```

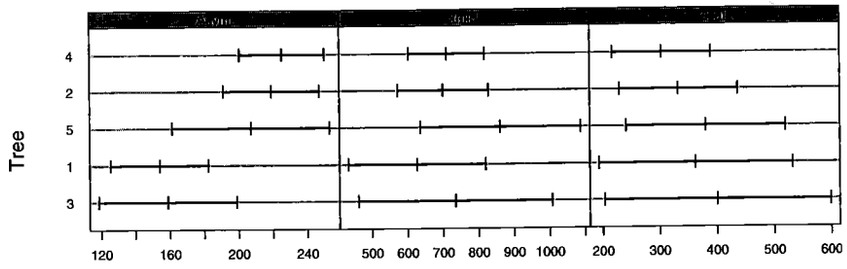

FIGURE 8.4. Ninety-five percent confidence intervals on the logistic model parameters for each tree in the orange trees data.

```
   scal
   Value Std. Error t value
3 400.95     94.776  4.2306
1 362.50     81.185  4.4652
5 379.99     66.761  5.6917
2 332.47     49.381  6.7327
4 303.13     41.608  7.2853
. . .
```

Although the estimates for all the parameters vary with tree, there appears to be relatively more variability in the `Asym` estimates. We can investigate this better with the `intervals` method.

```
> plot( intervals( fm1Oran.lis ), layout = c(3,1) )     # Figure 8.4
```

As mentioned in §6.2, confidence intervals for the same parameter in different groups within an `nlsList` fit do not necessarily have the same length, even with balanced data. This is evident in Figure 8.4.

The only parameter for which all the confidence intervals do not overlap in Figure 8.4 is `Asym`, suggesting that it is the only parameter for which random effects are needed to account for variation among trees.

The same `plot` method used for `lmList` objects is used to obtain diagnostic plots for an `nlsList` object. The boxplots of the residuals by tree, obtained with

```
> plot( fm1Oran.lis, Tree ~ resid(.), abline = 0 )     # Figure 8.5
```

and displayed in Figure 8.5, no longer indicate the "tree effect" observed in Figure 8.3. The basic drawback of the `nlsList` model is that uses 15 parameters to account for the individual tree effects. A more parsimonious representation is provided by the nonlinear mixed-effects model discussed in §8.2.

As a second example to illustrate the use of the `nlsList` function, we consider the theophylline data, which we used in §3.4. Recall that these data, displayed in Figure 8.6, are the serum concentrations of theophylline

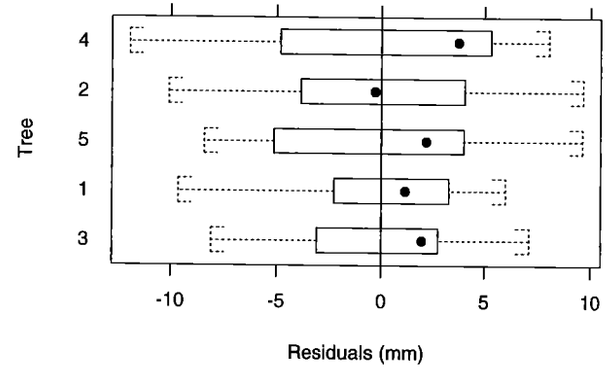

FIGURE 8.5. Boxplots of residuals by tree for `fm1Oran.lis`.

measured on twelve subjects at eleven times up to 25 hours after receiving an oral dose of the drug.

```
> Theoph[1:4,]
Grouped Data: conc ~ Time | Subject
  Subject   Wt Dose Time  conc
1       1 79.6 4.02 0.00  0.74
2       1 79.6 4.02 0.25  2.84
3       1 79.6 4.02 0.57  6.57
4       1 79.6 4.02 1.12 10.50
```

The column `Wt` in `Theoph` gives the `Subject`'s weight (in kilograms).

As when modeling the indomethacin data in §6.2 and the phenobarbital data in §6.4, we use a compartment model for these data. A first-order open-compartment model expresses the theophylline concentration c_t at time t after an initial dose D as

$$c_t = \frac{Dk_ek_a}{Cl\,(k_a - k_e)}\left[\exp\left(-k_et\right) - \exp\left(-k_at\right)\right]. \tag{8.2}$$

The parameters in the model are the *elimination* rate constant k_e, the *absorption* rate constant k_a, and the *clearance* Cl. For the model to be meaningful, all three parameters must be positive. To ensure ourselves of positive estimates while keeping the optimization problem unconstrained, we reparameterize model (8.2) in terms of the logarithm of the clearance and the rate constants.

$$c_t = \frac{D\exp\left(lKe + lKa - lCl\right)}{\exp\left(lKa\right) - \exp\left(lKe\right)}\left\{\exp\left[-\exp\left(lKe\right)t\right] - \exp\left[-\exp\left(lKa\right)t\right]\right\}, \tag{8.3}$$

where $lKe = \log(k_e)$, $lKa = \log(k_a)$, and $lCl = \log(Cl)$.

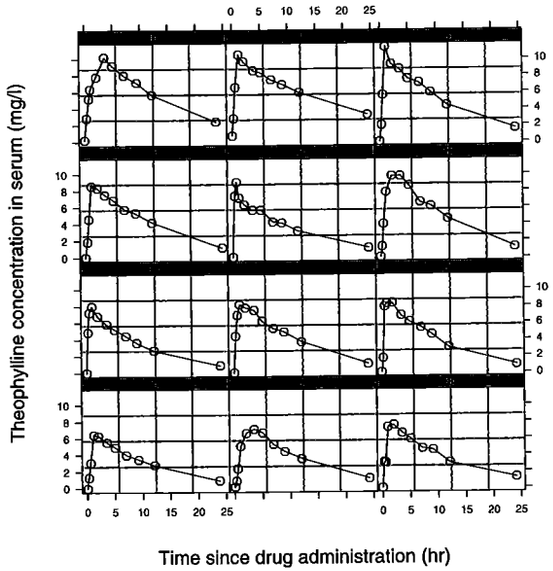

FIGURE 8.6. Serum concentrations of theophylline versus time since oral administration of the drug in twelve subjects.

The selfStart function `SSfol`, described in Appendix C.5, provides a self-starting implementation of (8.3). Because two covariates, *dose* and *time*, are present in (8.3), and hence also in the argument sequence of `SSfol`, we must specify the full model formula when calling `nlsList`.

```
> fm1Theo.lis <- nlsList( conc ~ SSfol(Dose, Time, lKe, lKa, lCl),
+    data = Theoph )
> fm1Theo.lis
Call:
  Model: conc ~ SSfol(Dose, Time, lKe, lKa, lCl) | Subject
   Data: Theoph

Coefficients:
       lKe      lKa     lCl
 6 -2.3074  0.15171 -2.9733
 7 -2.2803 -0.38617 -2.9643
 8 -2.3863  0.31862 -3.0691
11 -2.3215  1.34779 -2.8604
 3 -2.5081  0.89755 -3.2300
 2 -2.2862  0.66417 -3.1063
 4 -2.4365  0.15834 -3.2861
 9 -2.4461  2.18201 -3.4208
12 -2.2483 -0.18292 -3.1701
10 -2.6042 -0.36309 -3.4283
 1 -2.9196  0.57509 -3.9158
```


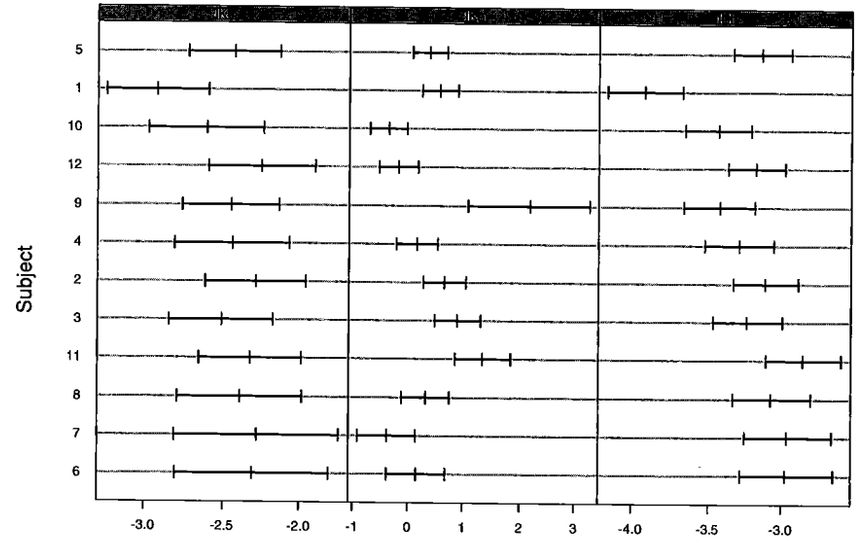

FIGURE 8.7. Ninety-five percent confidence intervals on the parameters in the first-order open-compartment model (8.3) for each subject in the theophylline data.

```
 5 -2.4254  0.38616 -3.1326

Degrees of freedom: 132 total; 96 residual
Residual standard error: 0.70019
```

The individual estimates suggest that the absorption rate constant is more variable among subjects than either the elimination rate constant or the clearance. As usual, we recommend using the plot of the confidence intervals on the individual parameters to analyze their between-group variation.

```
> plot( intervals( fm1Theo.lis ), layout = c(3,1) )     # Figure 8.7
```

The individual confidence intervals in Figure 8.7 indicate that there is substantial subject-to-subject variation in the absorption rate constant and moderate variation in the clearance. The elimination rate constant does not seem to vary significantly with subject.

The main purpose of the preliminary analysis provided by `nlsList` is to suggest a structure for the random effects to be used in a nonlinear mixed-effects model. We must decide which random effects to include in the model (`intervals` and its associated `plot` method are often useful for that) and what covariance structure these random effects should have. The `pairs` method, which is the same for lmList and nlsList objects, provides one view of the random effects covariance structure.

```
> pairs( fm1Theo.lis, id = 0.1 )                        # Figure 8.8
```

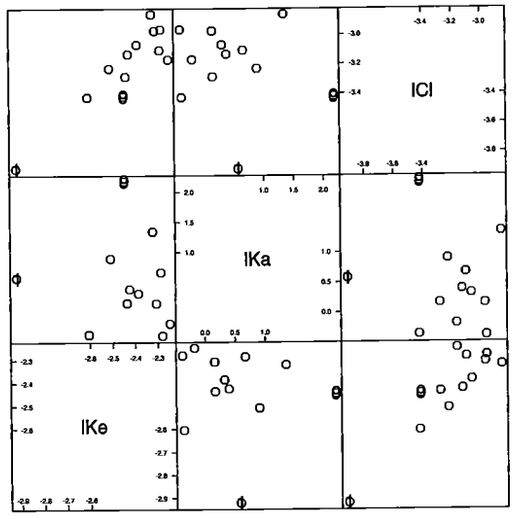

FIGURE 8.8. Pairs plot for the random effects estimates corresponding to `fm1Theo.lis`.

The scatter plots in Figure 8.8 suggest that Subject `1` has an unusually low elimination rate constant and clearance and that Subject `9` has an unusually high absorption rate constant. Overall, there do not appear to be significant correlations between the individual parameter estimates.

8.2 Fitting Nonlinear Mixed-Effects Models with `nlme`

The general formulation of a nonlinear mixed-effects model, and the estimation methods used to fit it, are described in §7.1. This section concentrates on the capabilities available in the nlme library for fitting nonlinear mixed-effects models with independent, homoscedastic within-group errors. Fitting `nlme` models to single-level grouped data is described in §8.2.1. The use of covariates to explain between-group variability and to reduce the number of random effects in an `nlme` model is presented in §8.2.2. In §8.2.3, we describe how to fit multilevel nonlinear mixed-effects models with `nlme`.

8.2.1 Fitting Single-Level `nlme` Models

Several optional arguments can be used with the `nlme` function, but a typical call looks like

```
nlme( model, data, fixed, random, groups, start )
```


The `model` argument is required and consists of either a two-sided formula specifying the nonlinear model to be fitted, or an `nlsList` object. Any S nonlinear formula can be used, giving the function considerable flexibility. For example, in the orange trees example one could either write down the logistic model explicitly

```
circumference ~ Asym/(1 + exp(-(age - xmid)/scal))
```

or use the selfStart function `SSlogis`

```
circumference ~ SSlogis(age, Asym, xmid, scal)
```

As with `nls` fits, there is an advantage of encapsulating the model expression in an S function when fitting an `nlme` model, in that it allows analytic derivatives of the model function to be passed to `nlme` and used in the optimization algorithm. The S function `deriv` can be used to create model functions that return the value of the model and its derivatives as a `gradient` attribute. If the value returned by the model function does not have a `gradient` attribute, numerical derivatives are used in the optimization.

The arguments `fixed` and `random` are formulas, or lists of formulas, defining the structures of the fixed and random effects in the model. In these formulas a `1` on the right-hand side of a formula indicates that a single parameter is associated with the effect, but any linear formula in S can be used. This gives considerable flexibility to the model, as time-dependent parameters can be incorporated easily (e.g., when a formula in the `fixed` list involves a covariate that changes with time). Each parameter in the model will usually have an associated fixed effect, but it may, or may not, have an associated random effect. Because the `nlme` model assumes that all random effects have expected value zero, the inclusion of a random effect without a corresponding fixed effect would be unusual. Any covariates defined in the `fixed` and `random` formulas can, alternatively, be directly incorporated in the `model` formula. However, declaring the covariates in `fixed` and `random` allows for more efficient calculation of derivatives and is useful for `update` methods. `Fixed` is required when `model` is declared as a formula. By default, when `random` is omitted, all fixed effects in the model are assumed to have an associated random effect.

In the theophylline example, to have random effects only for the log of the absorption rate constant, `lKa`, and the log-clearance, `lCl`, we use

```
fixed = list(lKe ~ 1, lKa ~ 1, lCl ~ 1),
random = list(lKa ~ 1, lCl ~ 1)
```

Model parameters with common right-hand side expressions in the `fixed` or `random` formulas can be collapsed into a single formula, in which the left-hand side lists the parameters separated by the + operator. For example, `fixed` and `random` in the theophylline example can be expressed more compactly as

```
fixed = lKe + lKa + lCl ~ 1
random = lKa + lCl ~ 1
```

If we wanted to allow the log-clearance fixed effect to depend linearly on the subject's weight while retaining a single component for the other fixed effects, we would use

```
fixed = list(lKe + lKa ~ 1, lCl ~ Wt)
```

Note that there would be two fixed effects associated with `lCl` in this case: an intercept and a slope with respect to `Wt`.

`Data` names a data frame in which any variables used in `model`, `fixed`, `random`, and `groups` are to be evaluated. By default, `data` is set to the environment from which `nlme` was called.

The `groups` argument is a one-sided formula, or an S expression, which, when evaluated in `data`, returns a factor with the group label of each observation. This argument does not need to be specified when `object` is an nlsList object, or when `data` is a groupedData object, or when `random` is a named list (in which case the name is used as the grouping factor).

The `start` argument provides a list, or a vector, of starting values for the iterative algorithm. When given as a vector, it is used as starting estimates for the fixed effects only. It is only required when `model` is given as a formula and the model function is not a selfStart object. In this case, starting values for the fixed effects must be specified. Starting estimates for the remaining parameters are generated automatically. By default, the random effects are initialized to zero.

Objects returned by `nlme` are of class nlme, which inherits from class lme. As a result, all the methods available for `lme` objects can also be applied to an `nlme` object. In fact, most methods are common to both classes. Table 8.3 lists the most important methods for class nlme. We illustrate the use of these methods through the examples in the next sections.

Growth of Orange Trees

The nonlinear mixed-effects model corresponding to the logistic model 8.1, with random effects for all parameters, is

$$
\begin{aligned}
y_{ij} &= \frac{\phi_{1i}}{1+\exp\left[-\left(t_{ij}-\phi_{2i}\right)/\phi_{3i}\right]} + \epsilon_{ij}, \\
\boldsymbol{\phi}_i &= \begin{bmatrix}\phi_{1i}\\ \phi_{2i}\\ \phi_{3i}\end{bmatrix} = \begin{bmatrix}\beta_1\\ \beta_2\\ \beta_3\end{bmatrix} + \begin{bmatrix}b_{1i}\\ b_{2i}\\ b_{3i}\end{bmatrix} = \boldsymbol{\beta} + \boldsymbol{b}_i, \\
\boldsymbol{b}_i &\sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{\Psi}\right), \qquad \epsilon_{ij} \sim \mathcal{N}\left(0, \sigma^2\right).
\end{aligned}
\tag{8.4}
$$

The model parameters $\boldsymbol{\phi}_i$ have the same interpretation as in (8.1), but are now allowed to vary with tree. The fixed effects, $\boldsymbol{\beta}$, represent the average value of the individual parameters, $\boldsymbol{\phi}_i$, in the population of orange trees and the random effects, $\boldsymbol{b}_i$, represent the deviations of the $\boldsymbol{\phi}_i$ from their population average. The random effects are assumed to be independent for different trees and the within-group errors ϵ_{ij} are assumed to be independent for different i, j and to be independent of the random effects.

TABLE 8.3. Main nlme methods.

`ACF`	empirical autocorrelation function of within-group residuals
`anova`	likelihood ratio or Wald-type tests
`augPred`	predictions augmented with observed values
`coef`	estimated coefficients for different levels of grouping
`fitted`	fitted values for different levels of grouping
`fixef`	fixed-effects estimates
`intervals`	confidence intervals on model parameters
`logLik`	log-likelihood at convergence
`pairs`	scatter-plot matrix of coefficients or random effects
`plot`	diagnostic Trellis plots
`predict`	predictions for different levels of grouping
`print`	brief information about the fit
`qqnorm`	normal probability plots
`ranef`	random-effects estimates
`resid`	residuals for different levels of grouping
`summary`	more detailed information about the fit
`update`	update the `nlme` fit
`Variogram`	semivariogram of within-group residuals

The nonlinear mixed-effects model (8.4) uses ten parameters to represent the fixed effects (three parameters), the random-effects variance–covariance structure (six parameters), and the within-group variance (one parameter). These numbers remain unchanged if we increase the number of trees being analyzed. In comparison, the number of parameters in the corresponding `nlsList` model, described in §8.1.3, is equal to three times the number of trees (15, in the orange trees example).

A nonlinear mixed-effects fit of model (8.4) can be obtained with

```
> ## no need to specify groups, as Orange is a groupedData object
> ## random is omitted - by default it is equal to fixed
> fm1Oran.nlme <-
+  nlme( circumference ~ SSlogis(age, Asym, xmid, scal),
+        data = Orange,
+        fixed = Asym + xmid + scal ~ 1,
+        start = fixef(fm1Oran.lis) )
```

but a much simpler, equivalent form is

```
> fm1Oran.nlme <- nlme( fm1Oran.lis )
```

The fm1Oran.lis object stores information about the model function, the parameters in the model, the groups formula, and the data used to fit the model. These are retrieved by nlme, allowing a more compact call. Another important advantage of using an nlsList object as the first argument to nlme is that it automatically provides initial estimates for the fixed effects, the random effects, and the random-effects covariance matrix.

We can now use the nlme methods to display the results and to assess the quality of the fit. As with lme objects, the print method gives some brief information about the fitted object.

```
> fm1Oran.nlme
Nonlinear mixed-effects model fit by maximum likelihood
  Model: circumference ~ SSlogis(age, Asym, xmid, scal)
  Data: Orange
  Log-likelihood: -129.99
  Fixed: list(Asym ~ 1, xmid ~ 1, scal ~ 1)
   Asym   xmid   scal
 192.12 727.74 356.73

Random effects:
 Formula: list(Asym ~ 1, xmid ~ 1, scal ~ 1)
 Level: Tree
 Structure: General positive-definite
          StdDev   Corr
    Asym 27.0302 Asym   xmid
    xmid 24.3761 -0.331
    scal 36.7363 -0.992  0.447
Residual  7.3208

Number of Observations: 35
Number of Groups: 5
```

Note that the default estimation method in nlme is maximum likelihood (ML), while the default in lme is restricted maximum likelihood (REML). The reason for this, as described in §7.2.2, is because nested nlme models which differ in either their fixed effects, or their random effects, cannot be compared using their REML likelihoods, thus invalidating most REML likelihood ratio tests of practical interest. In the linear case, REML likelihoods of nested mixed-effects models with common fixed-effects structure are comparable.

More detailed information about the fit is provided by the summary method.

```
> summary(fm1Oran.nlme)
 Nonlinear mixed-effects model fit by maximum likelihood
  Model: circumference ~ SSlogis(age, Asym, xmid, scal)
 Data: Orange
     AIC    BIC  logLik
  279.98 295.53 -129.99
```



```
Random effects:
 Formula: list(Asym ~ 1, xmid ~ 1, scal ~ 1)
 Level: Tree
 Structure: General positive-definite
          StdDev   Corr
    Asym 27.0302 Asym   xmid
    xmid 24.3761 -0.331
    scal 36.7363 -0.992  0.447
Residual  7.3208

Fixed effects: list(Asym ~ 1, xmid ~ 1, scal ~ 1)
      Value Std.Error DF t-value p-value
Asym 192.12    14.045 28  13.679  <.0001
xmid 727.74    34.618 28  21.022  <.0001
scal 356.73    30.537 28  11.682  <.0001
 Correlation:
       Asym   xmid
xmid  0.275
scal -0.194  0.666
. . .
```

It is interesting, at this point, to compare the nlme fit with the simple nls fit that ignores the grouped structure of the data, obtained in §8.1.1.

```
> summary(fm1Oran.nls)
. . .
Parameters:
      Value Std. Error t value
Asym 192.68     20.239  9.5203
xmid 728.71    107.272  6.7931
scal 353.49     81.460  4.3395

Residual standard error: 23.3721 on 32 degrees of freedom
. . .
```

The fixed-effects estimates are similar, but the standard errors are much smaller in the nlme fit. The estimated within-group standard error is also considerably smaller in the nlme fit. This is because the between-group variability is not incorporated in the nls model, being absorbed in the standard error. This pattern is generally observed when comparing mixed-effects versus fixed-effects fits.

In the fm1Oran.nlme fit the estimated correlation of -0.992 between Asym and xmid suggests that the estimated variance–covariance matrix is ill-conditioned and that the random-effects structure may be over-parameterized. The scatter-plot matrix of the estimated random effects, produced by the pairs method, provides a useful diagnostic plot for assessing over-parameterization problems.

```
> pairs( fm1Oran.nlme )                                  # Figure 8.9
```

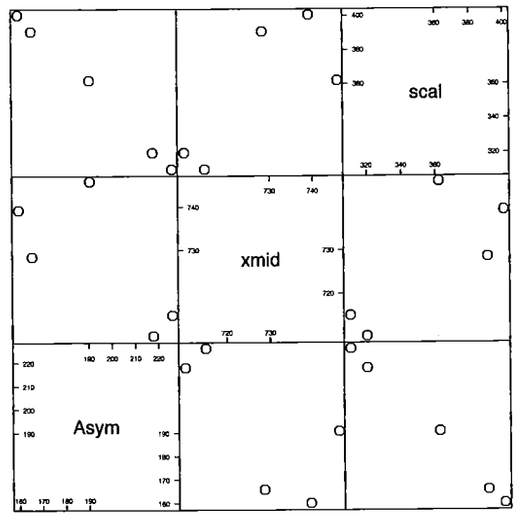

FIGURE 8.9. Pairs plot for the random-effects estimates corresponding to `fm1Oran.nlme`.

The nearly perfect alignment between the `Asym` random effects and the `scal` random effects further indicates that the model is over-parameterized.

The individual confidence intervals for the parameters in the `nlsList` model described by `fm1Oran.lis`, displayed in Figure 8.4 and discussed in §8.1.3, suggested that `Asym` was the only parameter requiring a random effect to account for its variation among trees. We can fit the corresponding model using the `update` method and compare it to the *full* model `fm1Oran.nlme` using the `anova` method.

```
> fm2Oran.nlme <- update( fm1Oran.nlme, random = Asym ~ 1 )
> anova( fm1Oran.nlme, fm2Oran.nlme )
             Model df    AIC    BIC  logLik   Test L.Ratio p-value
fm1Oran.nlme     1 10 279.98 295.53 -129.99
fm2Oran.nlme     2  5 273.17 280.95 -131.58 1 vs 2  3.1896  0.6708
```

The large p-value for the likelihood ratio test confirms that the `xmid` and `scal` random effects are not needed in the `nlme` model and that the simpler model `fm2Oran.nlme` is to be preferred.

As in the linear case, we must check if the assumptions underlying the nonlinear mixed-effects model appear valid for the model fitted to the data. The two basic distributional assumptions are the same as for the `lme` model:

> *Assumption 1*: the within-group errors are independent and identically normally distributed, with mean zero and variance σ^2, and they are independent of the random effects.

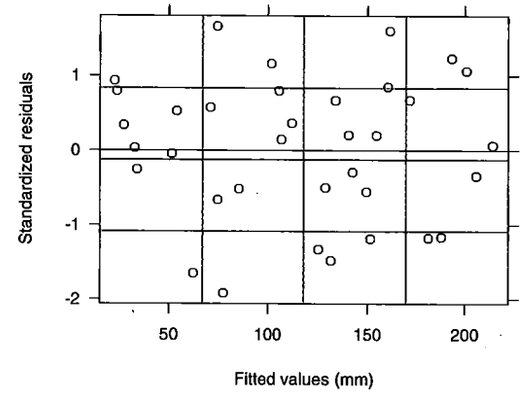

FIGURE 8.10. Scatter plot of standardized residuals versus fitted values for `fm1Oran.nlme`.

> *Assumption 2*: the random effects are normally distributed, with mean zero and covariance matrix Ψ (not depending on the group) and are independent for different groups;

The `plot` and `qqnorm` methods provide the most useful tools for assessing these assumptions. For example, a plot of the standardized residuals versus the fitted values in Figure 8.10

```
> plot( fm1Oran.nlme )                                  # Figure 8.10
```

shows that the residuals are distributed symmetrically around zero, with an approximately constant variance. It does not indicate any violations of the assumptions for the within-group error:

The adequacy of the fitted model is better visualized by displaying the fitted and observed values in the same plot. The `augPred` method, which produces the *augmented predictions*, and its associated `plot` method are used for that. For comparison, both the population predictions (obtained by setting the random effects to zero) and the within-group predictions (using the estimated random effects) are displayed in Figure 8.11, produced with

```
> ## level = 0:1 requests fixed (0) and within-group (1) predictions
> plot( augPred(fm2Oran.nlme, level = 0:1),              # Figure 8.11
+   layout = c(5,1) )
```

The tree-specific predictions follow the observed values closely, indicating that the logistic mixed-effects model (8.4) explains the trunk circumference growth in the orange trees data well.

The normal plot of the standardized residuals, shown in Figure 8.12, does not indicate any violations of the normality assumption for the within-group errors.

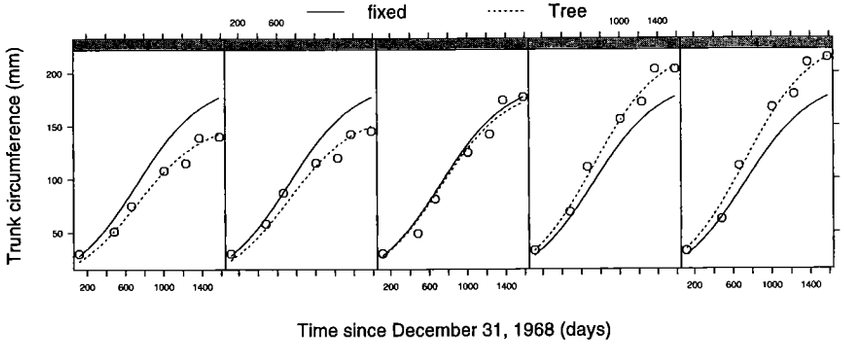

FIGURE 8.11. Population predictions (`fixed`), within-group predictions (`Tree`), and observed trunk circumferences (circles) versus age of tree, for the `fm2Oran.nlme` model .

```
> qqnorm( fm2Oran.nlme, abline = c(0,1) )                    #Figure 8.12
```

Because there are only five trees in the data, with just seven observations each, we cannot reliably test assumptions about the random-effects distribution and the independence of the within-group errors.

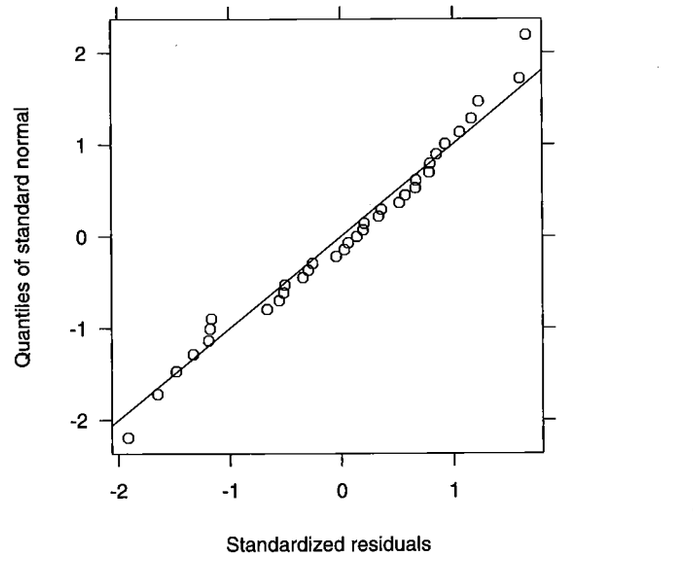

FIGURE 8.12. Normal probability plot of the standardized residuals from a nonlinear mixed-effects model fit `fm1Oran.nlme` to the orange data.

Theophylline Kinetics

The nonlinear mixed-effects version of the first-order open-compartment model (8.3), with all parameters as mixed-effects, is

$$
\begin{aligned}
c_{ij} = & \frac{D_i \exp\left(lKe_i + lKa_i - lCl_i\right)}{\exp\left(lKa_i\right) - \exp\left(lKe_i\right)} \\
& \times \left\{\exp\left[-\exp\left(lKe_i\right) t_{ij}\right] - \exp\left[-\exp\left(lKa_i\right) t_{ij}\right]\right\} + \epsilon_{ij}, \\
\boldsymbol{\phi}_i = & \begin{bmatrix} lKe_i \\ lKa_i \\ lCl_i \end{bmatrix} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} + \begin{bmatrix} b_{1i} \\ b_{2i} \\ b_{3i} \end{bmatrix} = \boldsymbol{\beta} + \boldsymbol{b}_i, \\
& \boldsymbol{b}_i \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{\Psi}\right), \quad \epsilon_{ij} \sim \mathcal{N}\left(0, \sigma^2\right),
\end{aligned}
\tag{8.5}
$$

where c_{ij} is the theophylline concentration for patient i measured at time t_{ij}, with an initial dose D_i. The pharmacokinetic parameters $\boldsymbol{\phi}_i$ are interpreted as in (8.3), but are now allowed to vary with subject. The fixed effects, $\boldsymbol{\beta}$, represent the population average of the $\boldsymbol{\phi}_i$ and the random effects, $\boldsymbol{b}_i$, their deviations from the population average.

Fitting the nonlinear mixed-effects model (8.5) to the theophylline data is done with the simple call

```
> fm1Theo.nlme <- nlme( fm1Theo.lis )
> fm1Theo.nlme
Nonlinear mixed-effects model fit by maximum likelihood
  Model: conc ~ SSfol(Dose, Time, lKe, lKa, lCl)
  Data: Theoph
  Log-likelihood: -173.32
  Fixed: list(lKe ~ 1, lKa ~ 1, lCl ~ 1)
     lKe      lKa      lCl
-2.4327 0.45146 -3.2145

Random effects:
 Formula: list(lKe ~ 1, lKa ~ 1, lCl ~ 1)
 Level: Subject
 Structure: General positive-definite
          StdDev   Corr
     lKe 0.13104 lKe    lKa
     lKa 0.63783  0.012
     lCl 0.25118  0.995 -0.089
Residual 0.68183

Number of Observations: 132
Number of Groups: 12
```

The estimated correlation between the `lKe` and `lCl` random effects is near one, indicating that the estimated random-effects variance–covariance matrix is ill-conditioned. We can investigate the precision of the correlation estimates with the `intervals` method.

```
> intervals( fm1Theo.nlme, which = "var-cov" )
Approximate 95% confidence intervals

 Random Effects:
  Level: Subject
                  lower      est.   upper
     sd(lKe)  0.058102  0.131142 0.29600
     sd(lKa)  0.397740  0.637838 1.02287
     sd(lCl)  0.156811  0.251216 0.40246
cor(lKe,lKa) -0.399511  0.011723 0.41903
cor(lKe,lCl) -0.995248  0.994868 1.00000
cor(lKa,lCl) -0.520140 -0.089480 0.37746

 Within-group standard error:
   lower    est.   upper
 0.59598 0.68183 0.78005
```

All three confidence intervals for the correlations include zero. The interval on the correlation between `lKe` and `lCl` shows this quantity is not estimated with any precision whatsoever. It must lie in the interval $[-1, 1]$ and the confidence interval is essentially that complete range.

As a first attempt at simplifying model (8.5), we investigate the assumption that the random effects are independent, that is, the matrix Ψ is diagonal. Structured random-effects variance—covariance matrices are specified in `nlme` in the same way as in `lme`: by using a `pdMat` constructor to specify the desired class of positive-definite matrix. The `pdMat` classes and methods are described in §4.2.2 and the standard pdMat classes available in the nlme library are listed in Table 4.3. By default, when no `pdMat` class is specified, a general positive-definite matrix (pdSymm class) is used to represent the random-effects variance–covariance structure. Alternative pdMat classes are specified by calling the corresponding constructor with the random effects formula, or list of formulas, as its first argument. For example, we specify a model with independent random effects for the theophylline data with either

```
> fm2Theo.nlme <- update( fm1Theo.nlme,
+   random = pdDiag(list(lKe ~ 1, lKa ~ 1, lCl ~ 1)) )
```

or

```
> fm2Theo.nlme <- update( fm1Theo.nlme,
+   random = pdDiag(lKe + lKa + lCl ~ 1) )
> fm2Theo.nlme
. . .
  Log-likelihood: -177.02
  Fixed: list(lKe ~ 1, lKa ~ 1, lCl ~ 1)
     lKe      lKa      lCl
 -2.4547  0.46554 -3.2272
```



```
Random effects:
 Formula: list(lKe ~ 1, lKa ~ 1, lCl ~ 1)
 Level: Subject
 Structure: Diagonal
              lKe      lKa      lCl Residual
StdDev: 1.9858e-05 0.64382 0.16692  0.70923
. . .
```

The very small estimated standard deviation for `lKe` suggests that the corresponding random effect could be omitted from the model.

```
> fm3Theo.nlme <-
+   update( fm2Theo.nlme, random = pdDiag(lKa + lCl ~ 1) )
```

We use the `anova` method to test the equivalence of the different `nlme` models used so far for the theophylline data. By including `fm3Theo.nlme` as the second model, we obtain the p-values comparing this model with each of the other two.

```
> anova( fm1Theo.nlme, fm3Theo.nlme, fm2Theo.nlme )
             Model df    AIC    BIC  logLik   Test L.Ratio p-value
fm1Theo.nlme     1 10 366.64 395.47 -173.32
fm3Theo.nlme     2  6 366.04 383.34 -177.02 1 vs 2  7.4046  0.1160
fm2Theo.nlme     3  7 368.05 388.23 -177.02 2 vs 3  0.0024  0.9611
```

The simpler `fm3Theo.nlme` model, with two independent random effects for `lKa` and `lCl`, has the smallest AIC and BIC. Also, the large p-values for the likelihood ratio tests comparing it to the other two models indicate that it should be preferred.

The plot of the standardized residuals versus the fitted values in Figure 8.13 gives some indication that the within-group variability increases with the drug concentration.

```
> plot( fm3Theo.nlme )                                    # Figure 8.13
```

The use of variance functions in `nlme` to model within-group heteroscedasticity is described in §8.3.1. We postpone further investigation of the within-group error assumptions in the theophylline until that section.

The `qqnorm` method is used to investigate the normality of the random effects.

```
> qqnorm( fm3Theo.nlme, ~ ranef(.) )                      # Figure 8.14
```

Figure 8.14 does not indicate any violations of the assumption of normality for the random effects.

8.2.2 Using Covariates with `nlme`

The random effects in a mixed-effects model represent deviations of the individual parameters from the fixed effects. In some applications, these

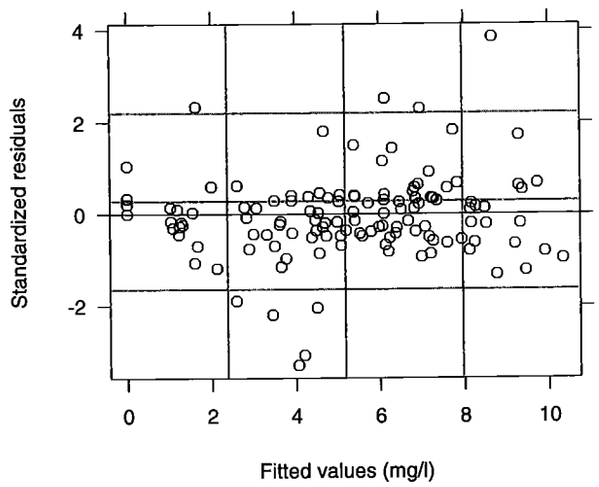

FIGURE 8.13. Scatter plot of standardized residuals versus fitted values for `fm3Theo.nlme`.

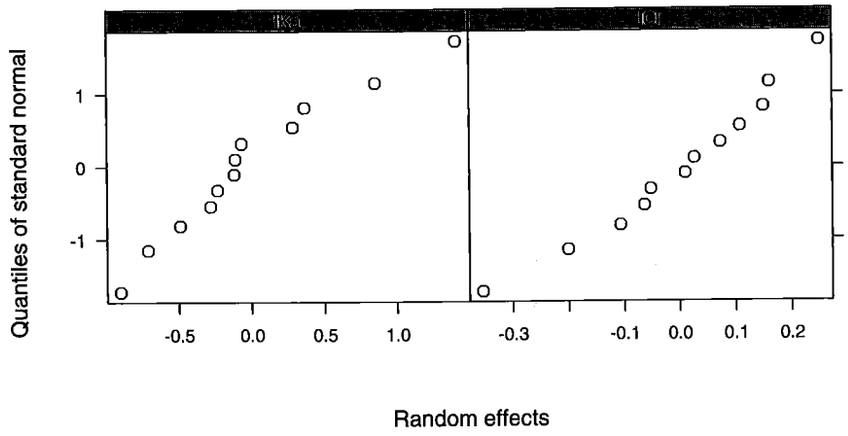

FIGURE 8.14. Normal plot of the estimated random effects corresponding to `fm3Theo.nlme` fit.

deviations arise from unexplained intergroup variation but, frequently, they can be at least partially explained by differences in covariate values among groups. For example, differences in the absorption rate constant among the subjects in the theophylline example of §8.2.1 may be attributed to differences in age, weight, blood pressure, etc.

Including covariates in the model to explain intergroup variation generally reduces the number of random effects in the model and leads to a better understanding of the mechanism producing the response. We have seen this in the soybean and the phenobarbital examples in Chapter 6.

Some of the questions that need to be addressed in the covariate modeling process are the following:

1. Among the candidate covariates, which are potentially useful in explaining the random-effects variation?
2. Which random effects have their variation best explained by covariates?
3. How should the potentially useful covariates be tested for inclusion in the model?
4. Should random effects be included in, or eliminated from, the modified model?

This section describes a model-building strategy for addressing these questions in the context of nonlinear mixed-effects models and using the capabilities of the nlme library.

Our general approach to address questions 1 and 2 above is to start with an `nlme` model with no covariates to explain random-effects variation, and use plots of the estimated random effects versus the candidate covariates to identify interesting patterns. Because the random effects accommodate individual departures from the population mean, plotting the estimated random effects against the candidate covariates provides useful information for the model-building process. A systematic pattern in a given random effect with respect to a covariate would indicate that the covariate should be included in the model.

If no interesting patterns are observed, we keep the current model. Otherwise, we choose the covariate-coefficient pair with the most promising pattern and test for the inclusion of the covariate in the model. After a covariate has been included in the model, we repeat the procedure for the estimated random effects in the updated model and the remaining candidate covariates, searching for further useful covariates.

The number of additional parameters to be estimated tends to grow considerably with the inclusion of covariates and their associated random effects in the model. If the number of covariate-coefficient combinations is large, we suggest using a *forward stepwise* approach in which covariate-coefficient pairs are included in the model one at a time and the potential

importance of the remaining covariates is graphically assessed at each step. The significance of the fixed-effects associated with a covariate included in the model is assessed using the Wald-type tests that are described in §7.2.2 and are included in the output of the `summary` and `anova` methods for nlme objects.

The inclusion of new random effects in the model when a covariate is added is rare, but should be investigated. The more common situation is that random effects can be eliminated from the model after covariates are included to account for intergroup variation. In both cases we proceed by comparing nested models using either likelihood ratio tests, or information criterion statistics (AIC and BIC).

We illustrate the use of the proposed model-building strategy with two examples, one from an experiment to evaluate the cold tolerance of grass species and the other from a clinical study of the antiarrhytimic drug quinidine.

Carbon Dioxide Uptake

Potvin, Lechowicz and Tardif (1990) report data from a study of the cold tolerance of a C_4 grass species, *Echinochloa crus-galli*. A total of 12 four-week-old plants, 6 from Québec and 6 from Mississippi, were divided into two groups: control plants that were kept at 26°C and chilled plants that were subject to 14 h of chilling at 7°C. After 10 h of recovery at 20°C, carbon dioxide (CO_2) uptake rates (in μmol/m^{2}s) were measured for each plant at seven concentrations of ambient CO_2 (μL/L). The objective of the experiment was to evaluate the effect of plant type and chilling treatment on the CO_2 uptake. The CO_2 data, displayed in Figure 8.15 (and in Figure 3.6, p. 113), are available in the nlme library as the groupedData object `CO2`. More details about these data are given in Appendix A.5.

```
> CO2
Grouped Data: uptake ~ conc | Plant
   Plant        Type  Treatment conc uptake
 1   Qn1      Quebec nonchilled   95   16.0
 2   Qn1      Quebec nonchilled  175   30.4
. . .
83   Mc3 Mississippi    chilled  675   18.9
84   Mc3 Mississippi    chilled 1000   19.9
> plot(CO2, outer = ~Treatment*Type, layout = c(4,1))  # Figure 8.15
```

It is clear from Figure 8.15 that the CO_2 uptake rates of Québec plants are greater than those of Mississippi plants and that chilling the plants reduces their CO_2 uptake rates.

An asymptotic regression model with an offset is used in Potvin et al. (1990) to represent the expected CO_2 uptake rate $U(c)$ as a function of the ambient CO_2 concentration c:

$$U(c) = \phi_1 \{1 - \exp[-\exp(\phi_2)(c - \phi_3)]\}, \tag{8.6}$$

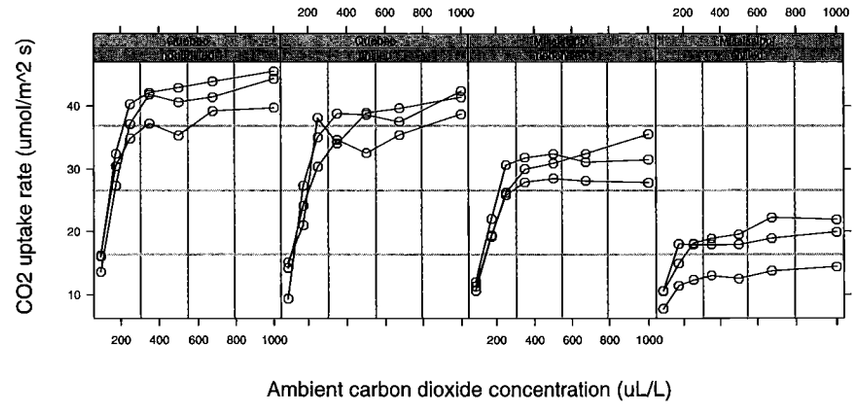

FIGURE 8.15. CO_2 uptake versus ambient CO_2 by treatment and type for *Echinochloa crus-galli* plants, 6 from Quebec and 6 from Mississippi. Half the plants of each type were chilled overnight before the measurements were taken.

where ϕ_1 is the asymptotic CO_2 uptake rate, ϕ_2 is the logarithm of the rate constant, and ϕ_3 is the maximum ambient concentration of CO_2 at which there is no uptake. The logarithm of the rate constant is used to enforce the positivity of the estimated rate constant, while keeping the optimization problem unconstrained.

The selfStart function `SSasympOff` gives a self-starting implementation of model (8.6), which is used to automatically generate starting estimates for the parameters in an `nlsList` fit.

```
> fm1CO2.lis <- nlsList( SSasympOff, CO2)
> fm1CO2.lis
Call:
  Model: uptake ~ SSasympOff(conc, Asym, lrc, c0) | Plant
   Data: CO2

Coefficients:
      Asym     lrc      c0
Qn1 38.140 -4.3807  51.221
Qn2 42.872 -4.6658  55.856
. . .
Mc3 18.535 -3.4654  67.843
Mc1 21.787 -5.1422 -20.395

Degrees of freedom: 84 total; 48 residual
Residual standard error: 1.7982
```

with $\phi_1 =$ `Asym`, $\phi_2 =$ `lrc`, and $\phi_3 =$ `c0`.

The plot of the individual confidence intervals from `fm1CO2.lis` (not shown) indicates that there is substantial between-plant variation in the

asymptote Asym and only moderate variation in the log-rate lrc and the offset c0. We initially consider a nonlinear mixed-effects version of the CO_2 uptake model (8.6) with all parameters as mixed effects and no treatment covariates. The corresponding model for the CO_2 uptake u_{ij} of plant i at ambient CO_2 concentration c_{ij} is

$$u_{ij} = \phi_{1i}\{1 - \exp[-\exp(\phi_{2i})(c_{ij} - \phi_{3i})]\} + \epsilon_{ij},$$
$$\boldsymbol{\phi}_i = \begin{bmatrix}\phi_{1i}\\ \phi_{2i}\\ \phi_{3i}\end{bmatrix} = \begin{bmatrix}\beta_1\\ \beta_2\\ \beta_3\end{bmatrix} + \begin{bmatrix}b_{1i}\\ b_{2i}\\ b_{3i}\end{bmatrix} = \boldsymbol{\beta} + \boldsymbol{b}_i, \tag{8.7}$$
$$\boldsymbol{b}_i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Psi}), \quad \epsilon_{ij} \sim \mathcal{N}(0, \sigma^2),$$

where ϕ_{1i}, ϕ_{2i}, and ϕ_{3i} have the same interpretation as in model (8.6), but are now allowed to vary with plant. The fixed effects, $\boldsymbol{\beta}$, represent the population average of the individual parameters, $\boldsymbol{\phi}_i$, and the random effects, $\boldsymbol{b}_i$, represent the deviations of the $\boldsymbol{\phi}_i$ from their population average. The random effects are assumed to be independent for different plots and the within-group errors ϵ_{ij} are assumed to be independent for different i, j and to be independent of the random effects.

The nlsList object fm1CO2.lis is used to produce starting estimates for the nlme fit of model (8.7).

```
> fm1CO2.nlme <- nlme( fm1CO2.lis )
> fm1CO2.nlme
Nonlinear mixed-effects model fit by maximum likelihood
  Model: uptake ~ SSasympOff(conc, Asym, lrc, c0)
  Data: CO2
  Log-likelihood: -201.31
  Fixed: list(Asym ~ 1, lrc ~ 1, c0 ~ 1)
   Asym      lrc      c0
 32.474 -4.6362 43.543

Random effects:
 Formula: list(Asym ~ 1, lrc ~ 1, c0 ~ 1)
 Level: Plant
 Structure: General positive-definite
           StdDev   Corr
    Asym  9.50999 Asym   lrc
     lrc  0.12828 -0.160
      c0 10.40519  0.999 -0.139
Residual  1.76641

Number of Observations: 84
Number of Groups: 12
```

The very high correlation between Asym and c0 suggests that the random-effects model is over-parameterized. The scatter plot matrix of the estimated random effects (not shown) confirms that Asym and c0 are in almost

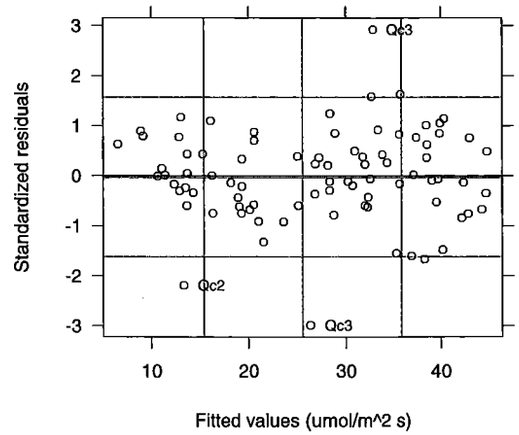

FIGURE 8.16. Scatter plot of standardized residuals versus fitted values for fm2CO2.nlme.

perfect linear alignment. A simpler model with just Asym and lrc as random effects gives an equivalent fit of the data.

```
> fm2CO2.nlme <- update( fm1CO2.nlme, random = Asym + lrc ~ 1 )
> fm2CO2.nlme
. . .
Random effects:
 Formula: list(Asym ~ 1, lrc ~ 1)
 Level: Plant
 Structure: General positive-definite
          StdDev   Corr
    Asym 9.65939 Asym
     lrc 0.19951 -0.777
Residual 1.80792
. . .
> anova( fm1CO2.nlme, fm2CO2.nlme )
           Model df    AIC    BIC  logLik   Test L.Ratio p-value
fm1CO2.nlme     1 10 422.62 446.93 -201.31
fm2CO2.nlme     2  7 419.52 436.53 -202.76 1 vs 2  2.8961  0.4079
```

The plot of the standardized residuals versus the fitted values in Figure 8.16 does not indicate any violations from the assumptions on the within-group error. The residuals are distributed symmetrically around zero, with uniform variance. Two large standardized residuals are observed for plant Qc3 and one for plant Qc2.

```
> plot( fm2CO2.nlme,id = 0.05,cex = 0.8,adj = -0.5 )  # Figure 8.16
```

The normal plot of the within-group residuals, not shown, does not indicate violations in the normality of the within-group errors.

The primary question of interest for the CO_2 data is the effect of plant type and chilling treatment on the individual model parameters ϕ_i. The random effects accommodate individual deviations from the fixed effects. Plotting the estimated random effects against the candidate covariates provides useful information for choosing covariates to include in the model. First, we need to extract the estimated random effects from the fitted model and combine them with the covariates. The `ranef` method accomplishes that.

```
> fm2CO2.nlmeRE <- ranef( fm2CO2.nlme, augFrame = T )
> fm2CO2.nlmeRE
         Asym        lrc        Type  Treatment conc uptake
Qn1   6.17160  0.0483563      Quebec nonchilled  435 33.229
Qn2  10.53264 -0.1728531      Quebec nonchilled  435 35.157
Qn3  12.21810 -0.0579930      Quebec nonchilled  435 37.614
Qc1   3.35213 -0.0755880      Quebec    chilled  435 29.971
Qc3   7.47431 -0.1924203      Quebec    chilled  435 32.586
Qc2   7.92855 -0.1803391      Quebec    chilled  435 32.700
Mn3  -4.07333  0.0334485 Mississippi nonchilled  435 24.114
Mn2  -0.14198  0.0056463 Mississippi nonchilled  435 27.343
Mn1   0.24056 -0.1938500 Mississippi nonchilled  435 26.400
Mc2 -18.79914  0.3193732 Mississippi    chilled  435 12.143
Mc3 -13.11688  0.2994393 Mississippi    chilled  435 17.300
Mc1 -11.78655  0.1667798 Mississippi    chilled  435 18.000
> class( fm2CO2.nlmeRE )
[1] "ranef.lme"  "data.frame"
```

The `augFrame` argument, when `TRUE`, indicates that summary values for all the variables in the data frame should be returned along with the random effects. The summary values are calculated as in the `gsummary` function (§3.4). When a covariate is constant within a group, such as `Treatment` and `Type` in the `CO2` data, its unique values per group are returned. Otherwise, if the covariate varies within the group and is numeric, such as `conc` and `uptake` in `CO2`, the group means are returned; if it is a categorical variable (factor or ordered), the most frequent values (modes) within each group are returned.

The `plot` method for the ranef.lme class is the most useful tool for identifying relationships between individual parameters and covariates. The `form` argument is used to specify the desired covariates and plot type. A one-sided formula on the right-hand side, with covariates separated by the `*` operator, results in a `dotplot` of the estimated random effects versus all combinations of the unique values of the variables named in the formula. This plot is particularly useful for a moderate number of categorical covariates (factor or ordered variables) with a relatively small number of levels, as in the `CO2` example.

```
> plot( fm2CO2.nlmeRE, form = ~ Type * Treatment )     # Figure 8.17
```


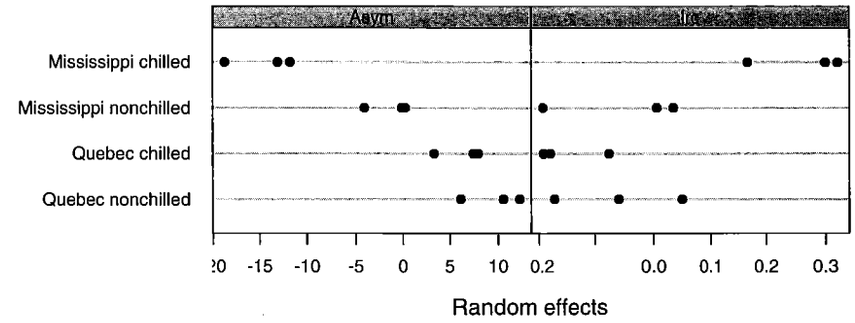

FIGURE 8.17. Dotplots of estimated random effects corresponding to `fm2CO2.nlme` versus all combinations of plant type and chilling treatment.

Figure 8.17 shows a strong relationship between the asymptotic uptake rate and the covariates: `Asym` decreases when the plants are chilled and is higher among Québec plants than Mississippi plants. The increase in `Asym` from chilled to nonchilled plants is larger among Mississippi plants than Québec plants, suggesting an interaction between `Type` and `Treatment`. There is also some evidence of a `Type:Treatment` interaction on the log-rate `lrc`, but it is less striking than in the case of `Asym`. We include both covariates in the model to explain the `Asym` plant-to-plant variation. The only change required in model (8.7) is in the formulation of ϕ_{1i}.

$$\begin{aligned}
\phi_{1i} &= \beta_1 + \gamma_1 x_{1i} + \gamma_2 x_{2i} + \gamma_3 x_{1i} x_{2i} + b_{1i}, \\
x_{1i} &= \begin{cases} -1, & \text{Type of Plant } i = \text{Québec}, \\ 1, & \text{Type of Plant } i = \text{Mississippi}, \end{cases} \\
x_{2i} &= \begin{cases} -1, & \text{Treatment of Plant } i = \text{nonchilled}, \\ 1, & \text{Treatment of Plant } i = \text{chilled}, \end{cases}
\end{aligned} \tag{8.8}$$

where β_1 represents the average asymptotic uptake rate, γ_1 and γ_2 represent, respectively, the plant type and chilling treatment main effects, and γ_3 represents the plant type–chilling treatment interaction. The parameterization used for x_{1i} and x_{2i} in (8.8) is consistent with the default parameterization for factors in S.

```
> contrasts(CO2$Type)
            [,1]
    Quebec    -1
Mississippi    1
> contrasts(CO2$Treatment)
           [,1]
nonchilled   -1
   chilled    1
```

The `update` method is used to fit the model with the covariate terms, which are specified through the `fixed` argument.

```
> fm3CO2.nlme <- update( fm2CO2.nlme,
+   fixed = list(Asym ~ Type * Treatment, lrc + c0 ~ 1),
+   start = c(32.412, 0, 0, 0, -4.5603, 49.344) )
```

Because the fixed-effects model has been reformulated, new starting values must be provided. We use the previous estimates for β_1, β_2 and β_3 and set the initial values for γ_1, γ_2 and γ_3 to zero. The fixed effects are represented internally in `nlme` in the same order they appear in `fixed`.

The `summary` method gives information about the significance of the individual fixed effects.

```
> summary( fm3CO2.nlme )
. . .
     AIC    BIC  logLik
  393.68 417.98 -186.84

Random effects:
 Formula: list(Asym ~ 1, lrc ~ 1)
 Level: Plant
 Structure: General positive-definite
                    StdDev   Corr
Asym.(Intercept) 2.92980 Asym.(
            lrc 0.16373 -0.906
       Residual 1.84957

Fixed effects: list(Asym ~ Type * Treatment, lrc + c0 ~ 1)
                     Value Std.Error DF t-value p-value
   Asym.(Intercept)  32.447    0.9359 67  34.670  <.0001
         Asym.Type  -7.108    0.5981 67 -11.885  <.0001
    Asym.Treatment  -3.815    0.5884 67  -6.483  <.0001
Asym.Type:Treatment -1.197    0.5884 67  -2.033   0.046
               lrc  -4.589    0.0848 67 -54.108  <.0001
                c0  49.479    4.4569 67  11.102  <.0001
. . .
```

The names of the fixed-effects terms include the parameter name. All fixed effects introduced in the model to explain the variability in `Asym` are significantly different from zero at the 5% level, confirming the previous conclusions from Figure 8.17. The *joint* significance of the fixed effects introduced in the model can be tested with the `anova` method.

```
> anova( fm3CO2.nlme, Terms = 2:4 )
F-test for: Asym.Type, Asym.Treatment, Asym.Type:Treatment
  numDF denDF F-value p-value
1     3    67  54.835  <.0001
```

As expected, the approximate F-test indicates that the added terms are highly significant.

The inclusion of the experimental factors in the model resulted in a reduction in the estimated standard deviation for the `Asym` random effects

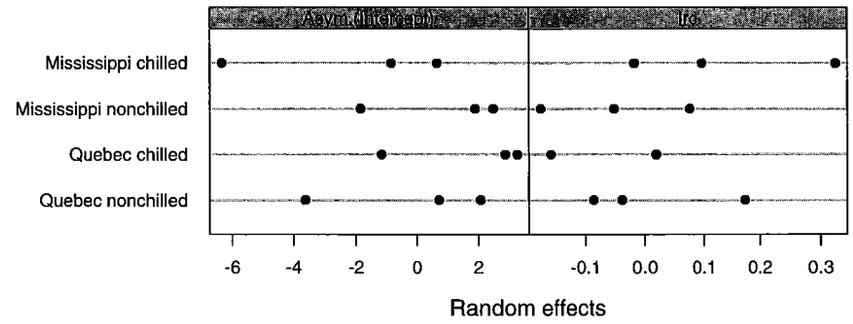

FIGURE 8.18. Dotplots of estimated random effects corresponding to `fm3CO2.nlme` versus all combinations of plant type and chilling treatment.

from 9.66 to 2.93, indicating that a substantial part of the plant-to-plant variation in the asymptotic uptake rate is explained by differences in plant type and chilling treatment. The standard deviations for the `lrc` random effects and the within-group error remained about the same.

We now investigate if any covariates should be included to account for the variability in the `lrc` random effects.

```
> fm3CO2.nlmeRE <- ranef( fm3CO2.nlme, aug = T )
> plot( fm3CO2.nlmeRE, form = ~ Type * Treatment )     # Figure 8.18
```

No systematic pattern can be observed for the estimated `Asym.(Intercept)` random effects in Figure 8.18, but it appears that the chilling treatment has opposite effects on Québec and Mississippi plants, suggesting an interaction. As before, we fit the augmented model with `update`, setting the initial values for the new fixed effects in the model to zero, and test the significance of the new terms using `summary`.

```
> fm3CO2.fix <- fixef( fm3CO2.nlme )          # for starting values
> fm4CO2.nlme <- update( fm3CO2.nlme,
+   fixed = list(Asym + lrc ~ Type * Treatment, c0 ~ 1),
+   start = c(fm3CO2.fix[1:5], 0, 0, 0, fm3CO2.fix[6]) )
> summary( fm4CO2.nlme )
. . .
Random effects:
 Formula: list(Asym ~ 1, lrc ~ 1)
 Level: Plant
 Structure: General positive-definite
                    StdDev   Corr
Asym.(Intercept) 2.349663 Asym.(
 lrc.(Intercept) 0.079608 -0.92
       Residual 1.791950

Fixed effects: list(Asym + lrc ~ Type * Treatment, c0 ~ 1)
```

```
                     Value Std.Error DF t-value p-value
Asym.(Intercept)    32.342    0.7849 64  41.208  <.0001
      Asym.Type    -7.990    0.7785 64 -10.264  <.0001
 Asym.Treatment    -4.210    0.7781 64  -5.410  <.0001
Asym.Type:Treatment -2.725  0.7781 64  -3.502  0.0008
lrc.(Intercept)    -4.509    0.0809 64 -55.743  <.0001
      lrc.Type      0.133    0.0552 64   2.417  0.0185
 lrc.Treatment      0.100    0.0551 64   1.812  0.0747
lrc.Type:Treatment  0.185    0.0554 64   3.345  0.0014
               c0  50.512    4.3646 64  11.573  <.0001
. . .
```

The `lrc.Type:Treatment` coefficient is highly significant and the `lrc.Type` coefficient is moderately significant. Even though `lrc.Treatment` is not significant (at a 5% level) we keep it in the model because the `Treatment` effect on `lrc` is involved in a highly significant interaction. The estimated standard deviation for the `lrc.(Intercept)` random effect is about 50% of the corresponding estimate in the `fm3CO2.nlme` fit. The remaining standard deviations are about the same as in the previous fit.

After covariates have been introduced in the model to account for intergroup variation, a natural question is which random effects, if any, are still needed. The ratio between a random-effects standard deviation and the absolute value of the corresponding fixed effect gives an idea of the relative intergroup variability for the coefficient, which is often useful in deciding which random effects should be tested for deletion from the model. For the `fm4CO2.nlme` fit these ratios are 7.3% for `Asym.(Intercept)` and 1.8% for `lrc.(Intercept)`, suggesting that the latter should be tested for exclusion first.

```
> fm5CO2.nlme <- update( fm4CO2.nlme, random = Asym ~ 1)
> anova( fm4CO2.nlme, fm5CO2.nlme )
            Model df    AIC    BIC  logLik   Test L.Ratio p-value
fm4CO2.nlme     1 13 388.42 420.02 -181.21
fm5CO2.nlme     2 11 387.06 413.79 -182.53 1 vs 2  2.6369  0.2675
```

The large p-value for the likelihood ratio test and the smaller AIC and BIC values for `fm5CO2.nlme` indicate that no random effects are needed for `lrc`.

To test if a random effect is needed for the asymptotic uptake rate, we need to fit a nonlinear fixed-effects model to the `CO2` data. The `nls` function can be used for that, though it is not designed to efficiently handle parameters that are expressed as linear combinations of covariates. (The `gnls` function, described in §8.3.3, is better suited for this type of model.) To use `nls`, we must first create variables representing the contrasts of interest

```
> CO2$type <- 2 * (as.integer(CO2$Type) - 1.5)
> CO2$treatment <- 2 * (as.integer(CO2$Treatment) - 1.5)
```

and then define all coefficients explicitly in the model formula.

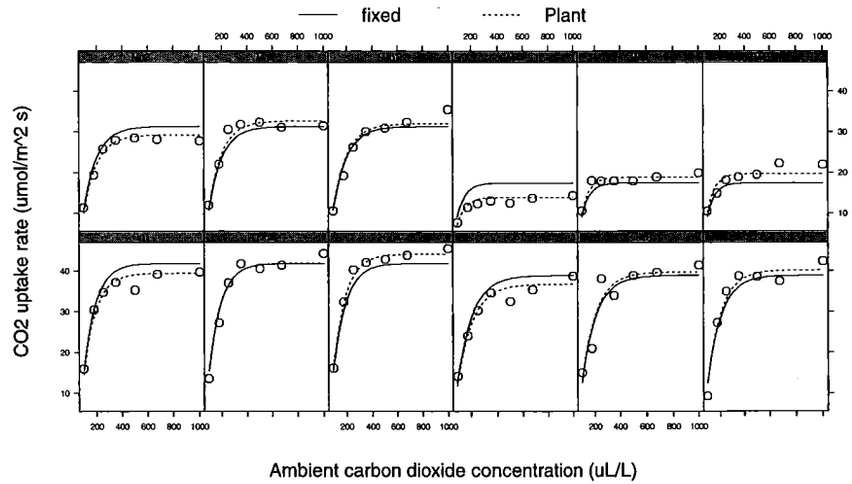

FIGURE 8.19. Plant-specific (`Plant`) and population average (`fixed`) predicted CO_2 uptake rates obtained from `fm5CO2.nlme`.

```
> fm1CO2.nls <- nls(uptake ~ SSasympOff(conc, Asym.Intercept +
+   Asym.Type * type + Asym.Treatment * treatment +
+   Asym.TypeTreatment * type * treatment, lrc.Intercept +
+   lrc.Type * type + lrc.Treatment * treatment +
+   lrc.TypeTreatment * type * treatment, c0), data = CO2,
+   start = c(Asym.Intercept = 32.371, Asym.Type = -8.0086,
+     Asym.Treatment = -4.2001, Asym.TypeTreatment = -2.7253,
+     lrc.Intercept = -4.5267, lrc.Type =  0.13112,
+     lrc.Treatment = 0.093928, lrc.TypeTreatment = 0.17941,
+     c0 = 50.126) )
```

The `anova` method can then be used to compare the models. (Note that the `nlme` object must appear first in the calling sequence to `anova`, so that the correct method is invoked.)

```
> anova( fm5CO2.nlme, fm1CO2.nls )
           Model df    AIC    BIC  logLik   Test L.Ratio p-value
fm5CO2.nlme     1 11 387.06 413.79 -182.53
 fm1CO2.nls     2 10 418.34 442.65 -199.17 1 vs 2  33.289  <.0001
```

The very significant p-value for the likelihood ratio test indicates that the `Asym.(Intercept)` random effect is still needed in the model.

A final assessment of the quality of the fitted model is provided by the plot of the augmented predictions included in Figure 8.19.

```
> plot( augPred(fm5CO2.nlme, level = 0:1),              # Figure 8.19
+    layout = c(6,2) )
```

The plant-specific predictions are in good agreement with the observed CO_2 uptake rates, attesting to the adequacy of the asymptotic regression

model. Note that the population average predictions vary with plant type and chilling treatment.

Clinical Study of Quinidine

The `Quinidine` data were described in §3.4 where we explored the structure of the data through summaries. Like the phenobarbital data analyzed in §6.4, the quinidine data are routine clinical pharmacokinetic data characterized by extensive dosage histories for each patient, but relatively sparse information on concentration. Recall that a total of 361 quinidine concentration measurements were made on 136 hospitalized patients under varying dosage regimens. The times since hospitalization at which the quinidine concentrations were measured varied between 0.13 and 8095.5 hours. Most patients have only a few concentration measurements: 34% have only one and 80% have three or fewer. Only 5% of the patients have seven or more observations.

Additional demographic and physiological data were collected for each subject. The additional available covariates are described in Table 8.4. Some of these covariates, such as age, body weight, and creatinine clearance, were "time-varying." That is, their value for a particular patient could change during the course of the study. Others, such as race, remained constant. One of the main objectives of the study was to investigate relationships between the individual pharmacokinetic parameters and the covariates. Statistical analyses of these data using alternative modeling approaches are given in Davidian and Gallant (1992) and in Wakefield (1996).

The model that has been suggested for the quinidine data is the one-compartment open model with first-order absorption. This model can be defined recursively as follows. Suppose that, at time t, a patient receives a dose d_t and prior to that time the last dose was given at time t'. The expected concentration in the serum compartment, C_t, and in the absorption

TABLE 8.4. Demographic and physiological covariates in the quinidine data.

Age (yr)	42–92
Glycoprotein concentration (mg/100 dL)	0.39–3.16
Body weight (kg)	41–119
Congestive heart failure	no/mild, moderate, severe
Creatinine clearance (ml/min)	< 50, ≥ 50
Ethanol abuse	none, current, former
Height (in.)	60–79
Race	Caucasian, Latin, Black
Smoking status	no, yes

compartment, Ca_t, are given by

$$\begin{aligned} C_t =& C_{t'} \exp\left[-k_e\,(t-t')\right] + \frac{Ca_{t'}\,k_a}{k_a - k_e} \\ & \times \left\{\exp\left[-k_e\,(t-t')\right] - \exp\left[-k_a\,(t-t')\right]\right\}, \\ Ca_t =& Ca_{t'} \exp\left[-k_a\,(t-t')\right] + \frac{d_t}{V}, \end{aligned} \tag{8.9}$$

where V is the apparent volume of distribution, k_a is the absorption rate constant, and k_e is the elimination rate constant.

When a patient receives the same dose d at regular time intervals Δ, model (8.9) converges to the steady state model

$$\begin{aligned} C_t &= \frac{d\,k_a}{V\,(k_a - k_e)}\left[\frac{1}{1-\exp\left(-k_e\Delta\right)} - \frac{1}{1-\exp\left(-k_a\Delta\right)}\right], \\ Ca_t &= \frac{d}{V\left[1-\exp\left(-k_a\Delta\right)\right]}. \end{aligned} \tag{8.10}$$

Finally, for a between-dosages time t, the model for the expected concentration C_t, given that the last dose was received at time t', is identical to (8.9).

Using the fact that the elimination rate constant k_e is equal to the ratio between the clearance (Cl) and the volume of distribution (V), we can reparameterize models (8.9) and (8.10) in terms of V, k_a, and Cl.

To ensure that the estimates of V, k_a, and Cl are positive, we can rewrite models (8.9) and (8.10) in terms of $lV = \log(V)$, $lKa = \log(k_a)$ and $lCl = \log(Cl)$.

The initial conditions for the recursive models (8.9) and (8.10) are $C_0 = 0$ and $Ca_0 = d_0/V$, with d_0 denoting the initial dose received by the patient. It is assumed in the model's definition that the bioavailability of the drug—the percentage of the administered dose that reaches the measurement compartment—is equal to one.

The function `quinModel` in the `nlme` library implements the recursive models (8.9) and (8.10) in S, parameterized in terms of lV, lKa and lCl. This is *not* a self-starting model, so initial values for the fixed effects need to be provided when calling `nlme`. We used values reported in the literature as starting estimates for the fixed effects.

Preliminary analyses of the data, without using any covariates to explain intersubject variation, indicates that only lCl and lV need random effects to account for their variability in the patient population, and that the corresponding random effects can be assumed to be independent. The corresponding model for the fixed and random effects is

$$\begin{aligned} lCl_i = \beta_1 + b_{1i}, \quad lV_i = \beta_2 + b_{2i}, \quad lKa_i = \beta_3, \\ \boldsymbol{b}_i = \begin{bmatrix} b_{1i} \\ b_{2i} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \psi_1 & 0 \\ 0 & \psi_2 \end{bmatrix}\right), \end{aligned} \tag{8.11}$$

which is fitted in S with

```
> fm1Quin.nlme <-
+   nlme(conc ~ quinModel(Subject, time, conc, dose, interval,
+                         lV, lKa, lCl),
+        data = Quinidine, fixed = lV + lKa + lCl ~ 1,
+        random = pdDiag(lV + lCl ~ 1), groups =  ~ Subject,
+        start = list(fixed = c(5, -0.3, 2)),
+        na.action = na.include, naPattern =  ~ !is.na(conc) )
> fm1Quin.nlme
Nonlinear mixed-effects model fit by maximum likelihood
  Model: conc ~ quinModel(Subject, time, conc, dose, interval, ...
  Data: Quinidine
  Log-likelihood: -495.77
  Fixed: lV + lKa + lCl ~ 1
     lV      lKa     lCl
 5.3796 -0.20535 2.4687

Random effects:
 Formula: list(lV ~ 1, lCl ~ 1)
 Level: Subject
 Structure: Diagonal
             lV     lCl Residual
StdDev: 0.31173 0.32276  0.73871

Number of Observations: 361
Number of Groups: 136
```

The `na.action` and `naPattern` arguments in this call to `nlme` are described in §6.4.

To investigate which covariates may account for patient-to-patient variation in the pharmacokinetic parameters, we first extract the estimated random effects, augmented with summary values for the available covariates (the modal value is used for time-varying factors and the mean for time-varying numeric variables).

```
> fm1Quin.nlmeRE <- ranef( fm1Quin.nlme, aug = T )
> fm1Quin.nlmeRE[1:3,]
            lV         lCl  time    conc dose interval Age Height
109  0.0005212 -0.0028369 61.58 0.50000   NA       NA  70     67
 70  0.0362214  0.3227614  1.50 0.60000   NA       NA  68     69
 23 -0.0254211  0.4402551 91.14 0.56667   NA       NA  75     72
    Weight      Race Smoke Ethanol   Heart Creatinine   glyco
109     58 Caucasian    no    none No/Mild      >= 50 0.46000
 70     75 Caucasian    no  former No/Mild      >= 50 1.15000
 23    108 Caucasian   yes    none No/Mild      >= 50 0.83667
```

The `dotplot` displays used to visualize the relationships between the estimated random effects and the covariates in the CO_2 example do not scale up well when there are a large number of groups, or a large number of covariates in the data, as in the quinidine study. Also, they cannot be used with numeric covariates, like `Weight` and `Age`. The `plot` method for class `ranef` allows a more flexible type of trellis display for these situations. Relationships between estimated random effects and factors are displayed using boxplots, while scatter plots are used for displaying the relationships between the estimated random effects and numeric covariates. Specifying a two-sided formula in the `form` argument, with the random effect on left-hand side and the desired covariates, separated by the + operator, on the right-hand side, indicates to the `plot` method that the more general trellis display should be used. For example, to plot the estimated `lCl` random effects against the available covariates we use


```
> plot( fm1Quin.nlmeRE, form = lCl ~  Age + Smoke + Ethanol +
+         Weight + Race + Height + glyco + Creatinine + Heart,
+       control = list(cex.axis = 0.7) )            #  Figure 8.20
```

The resulting plot, shown in Figure 8.20, indicates that clearance decreases with glycoprotein concentration and age, and increases with creatinine clearance and weight. There is also evidence that clearance decreases with severity of congestive heart failure and is smaller in Blacks than in both Caucasians and Latins. The glycoprotein concentration is clearly the most important covariate for explaining the `lCl` interindividual variation. A straight line seems adequate to model the observed relationship.

Figure 8.21 presents the plots of the estimated `lV` random effects versus the available covariates. None of the covariates seems helpful in explaining the variability of this random effect and we do not pursue the modeling of its variability any further.

Initially, only the glycoprotein concentration is included in the model to explain the `lCl` random-effect variation according to a linear model. This modification of model (8.11) is accomplished by writing

$$lCl_{ij} = (\beta_1 + b_{1i}) + \beta_2 \text{glyco}_{ij}. \tag{8.12}$$

Because the glycoprotein concentration may change with time on the same patient, the random effects for `lCl` need to be indexed by both patient i and time j. We fit the mixed-effects model corresponding to (8.12) with

```
> fm1Quin.fix <- fixef( fm1Quin.nlme)          # for initial values
> fm2Quin.nlme <- update( fm1Quin.nlme,
+   fixed = list(lCl ~ glyco, lKa + lV ~ 1),
+   start = c(fm1Quin.fix[3], 0, fm1Quin.fix[2:1]) )
> summary( fm2Quin.nlme )
. . .
    AIC    BIC  logLik
  891.3 918.52 -438.65
```

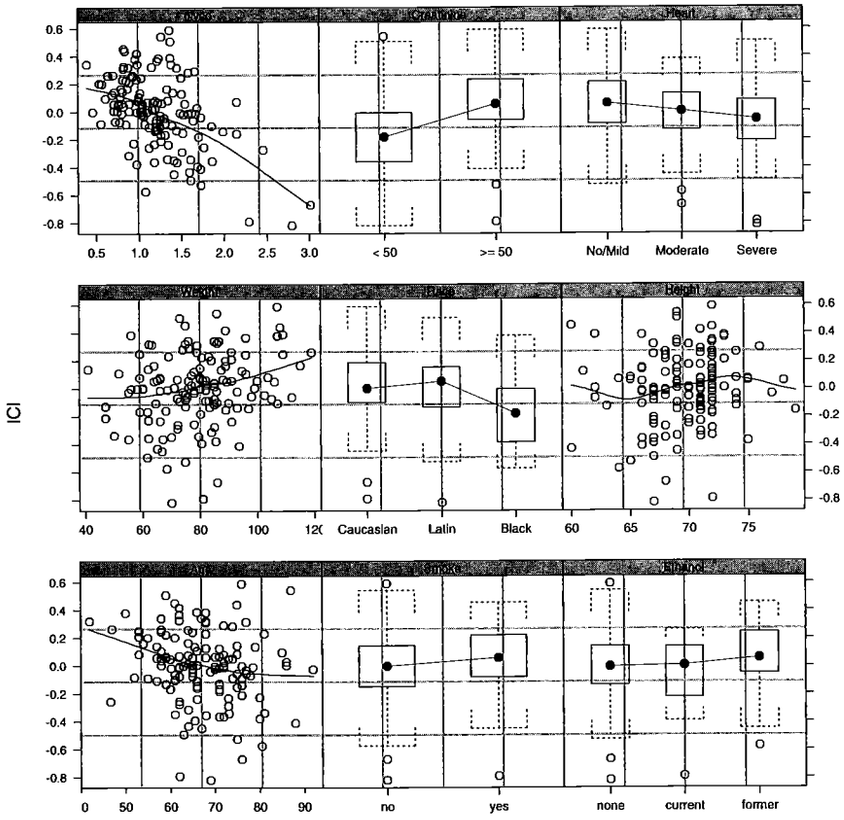

FIGURE 8.20. Estimated log-clearance random effects from model `fm1Quin.nlme` versus demographic and physiological covariates in the quinidine data. A `loess` smoother is included in the scatter plots of the continuous covariates to aid in visualizing possible trends.

```
Random effects:
 Formula: list(lV ~ 1, lCl ~ 1)
 Level: Subject
 Structure: Diagonal
            lV lCl.(Intercept) Residual
StdDev: 0.26764         0.27037  0.63746

Fixed effects: list(lCl ~ glyco, lKa + lV ~ 1)
                  Value Std.Error  DF t-value p-value
lCl.(Intercept)  3.1067   0.06473 222  47.997  <.0001
     lCl.glyco -0.4914   0.04263 222 -11.527  <.0001
           lKa -0.6662   0.30251 222  -2.202  0.0287
            lV  5.3085   0.10244 222  51.818  <.0001
. . .
```


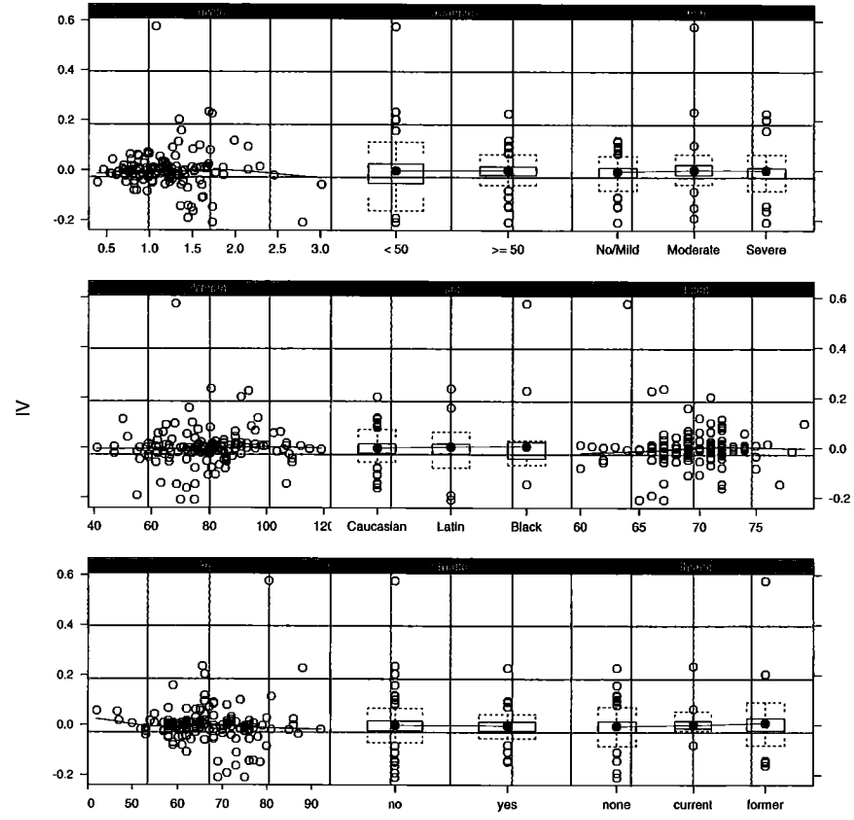

FIGURE 8.21. Estimated log-volume random effects from model `fm1Quin.nlme` versus demographic and physiological covariates in the quinidine data. A `loess` smoother is included in the scatter plots of the continuous covariates to aid in visualizing possible trends.

The estimated `lCl.glyco` fixed effect is very significant, indicating that the glycoprotein concentration should be kept in the model.

To search for further variables to include in the model, we consider the plots of the estimated `lCl.(Intercept)` random effects from the `fm2Quin.nlme` fit versus the covariates, presented in Figure 8.22.

These plots indicate that the estimated `lCl.(Intercept)` random effects increase with creatinine clearance, weight, and height, decrease with age and severity of congestive heart failure, and are smaller in Blacks than in Caucasians and Latins. The most relevant variable appears to be the creatinine clearance, which is included in the model as a binary variable taking value 0 when creatinine is < 50 and 1 when creatinine is ≥ 50.

```
> options( contrasts = c("contr.treatment", "contr.poly") )
```

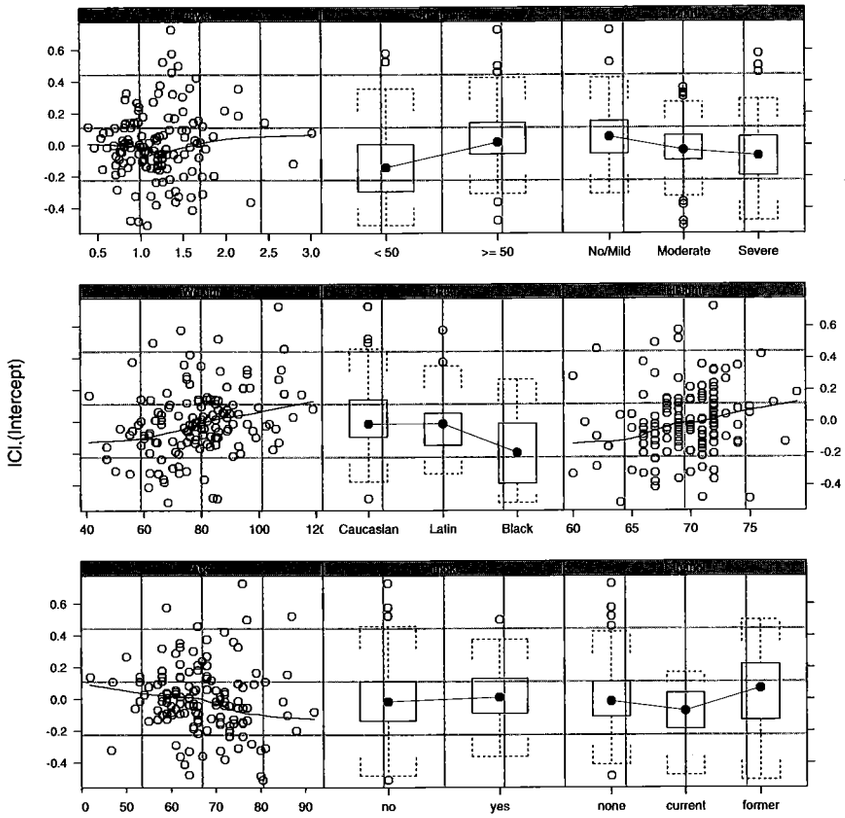

FIGURE 8.22. Estimated log-clearance random effects from model `fm2Quin.nlme` versus demographic and physiological covariates in the quinidine data. A `loess` smoother is included in the scatter plots of the continuous covariates to aid in visualizing possible trends.

```
> fm2Quin.fix <- fixef( fm2Quin.nlme )
> fm3Quin.nlme <- update( fm2Quin.nlme,
+   fixed = list(lCl ~ glyco + Creatinine, lKa + lV ~ 1),
+   start = c(fm2Quin.fix[1:2], 0.2, fm2Quin.fix[3:4]) )
> summary( fm3Quin.nlme )
. . .
Fixed effects: list(lCl ~ glyco + Creatinine, lKa + lV ~ 1)
                  Value Std.Error  DF t-value p-value
lCl.(Intercept)  3.0291   0.06387 221  47.426  <.0001
      lCl.glyco -0.4631   0.04117 221 -11.249  <.0001
 lCl.Creatinine  0.1503   0.03175 221   4.732  <.0001
            lKa -0.7458   0.29619 221  -2.518  0.0125
             lV  5.2893   0.10625 221  49.784  <.0001
. . .
```


The final model produced by this stepwise model-building approach includes an extra term for the patient's body weight to explain the clearance variation. The corresponding model for the log-clearance is expressed as

$$lCl_{ij} = (\beta_1 + b_{1i}) + \beta_4 \text{glyco}_{ij} + \beta_5 \text{Creatinine}_{ij} + \beta_6 \text{Weight}_{ij} \qquad (8.13)$$

and is fit in S with

```
> fm3Quin.fix <- fixef( fm3Quin.nlme )
> fm4Quin.nlme <- update( fm3Quin.nlme,
+   fixed = list(lCl ~ glyco + Creatinine + Weight, lKa + lV ~ 1),
+   start = c(fm3Quin.fix[1:3], 0, fm3Quin.fix[4:5]) )
> summary( fm4Quin.nlme )
. . .
     AIC    BIC  logLik
  870.94 905.94 -426.47

Random effects:
 Formula: list(lV ~ 1, lCl ~ 1)
 Level: Subject
 Structure: Diagonal
            lV lCl.(Intercept) Residual
StdDev: 0.28154         0.24128  0.63083

Fixed effects: list(lCl~glyco + Creatinine + Weight, lKa+lV ~ 1)
                  Value Std.Error  DF t-value p-value
lCl.(Intercept)  2.7883   0.15167 220  18.384  <.0001
      lCl.glyco -0.4645   0.04100 220 -11.328  <.0001
 lCl.Creatinine  0.1373   0.03264 220   4.207  <.0001
     lCl.Weight  0.0031   0.00180 220   1.749  0.0816
            lKa -0.7974   0.29959 220  -2.662  0.0083
             lV  5.2833   0.10655 220  49.587  <.0001
. . .
```

The `lCl.Weight` coefficient is not significant at a 5% level, but it is significant at a less conservative 10% level. Given the high level of noise and the small number of observations per patient in the quinidine data, we considered a p-value of 8.2% to be small enough to justify the inclusion of `Weight` in the model.

As reported in previous analyses of the quinidine data (Davidian and Giltinan, 1995, §9.3), there is evidence that the variability in the concentration measurements increases with the quinidine concentration. We postpone the investigation of heteroscedasticity in the quinidine data until §8.3.1, when we describe the use of variance functions in `nlme`.

8.2.3 *Fitting Multilevel* `nlme` *Models*

Nonlinear mixed-effects models with nested grouping factors, called *multilevel nonlinear mixed-effects* models, are described in §7.1.2. In this section

we describe how to fit and analyze them with the nlme library. The `Wafer` example of §4.2.3 is used to illustrate the various methods available in nlme for multilevel NLME models.

As in the linear case, the only difference between a single-level and a multilevel fit in `nlme` is in the specification of the `random` argument. In the multilevel case, `random` must provide information about the nested grouping structure and the random-effects model at each level of grouping. This is generally accomplished by specifying `random` as a *named* list, with names corresponding to the grouping factors. The order of nesting is assumed to be the same as the order of the elements in the list, with the outermost grouping factor appearing first and the innermost grouping factor appearing last. Each element of the `random` list has the same structure as `random` in a single-level call: it can be a formula, a list of formulas, a `pdMat` object, or a list of `pdMat` objects. Most of the nlme extractors, such as `resid` and `ranef`, include a `level` argument indicating the desired level(s) of grouping.

Manufacturing of Analog MOS Circuits

A multilevel *linear* mixed-effects analysis of the `Wafer` data is presented in §4.2.3 to illustrate the multilevel capabilities of `lme`. The final multilevel model obtained in that section to represent the intensity of current y_{ijk} at the kth level of voltage v_k in the jth site within the ith wafer is expressed, for $i = 1, \ldots, 10$, $j = 1, \ldots, 8$, and $k = 1, \ldots, 5$, as

$$\begin{aligned} y_{ijk} &= (\beta_0 + b_{0i} + b_{0i,j}) + (\beta_1 + b_{1i} + b_{1i,j})\, v_k + (\beta_2 + b_{2i} + b_{2i,j})\, v_k^2 \\ &\quad + \beta_3 \cos(\omega v_k) + \beta_4 \sin(\omega v_k) + \epsilon_{ijk} \\ \boldsymbol{b}_i = \begin{bmatrix} b_{0i} \\ b_{1i} \\ b_{2i} \end{bmatrix} &\sim \mathcal{N}(0, \boldsymbol{\Psi}_1), \qquad \boldsymbol{b}_{i,j} = \begin{bmatrix} b_{0i,j} \\ b_{1i,j} \\ b_{2i,j} \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Psi}_2), \\ \epsilon_{ijk} &\sim \mathcal{N}(0, \sigma^2). \end{aligned} \tag{8.14}$$

where β_0, β_1, and β_2 are the fixed effects in the quadratic model, β_3 and β_4 are the fixed effects for the cosine wave of frequency ω, $\boldsymbol{b}_i$ is the *wafer-level* random effects vector, $\boldsymbol{b}_{i,j}$ is the *site within wafer-level* random-effects vector, and ϵ_{ijk} is the within-group error. The $\boldsymbol{b}_i$ are assumed to be independent for different i, the $\boldsymbol{b}_{i,j}$ are assumed to be independent for different i, j and independent of the $\boldsymbol{b}_i$, and the ϵ_{ijk} are assumed to be independent for different i, j, k and independent of the random effects. The wafer-level variance–covariance matrix $\boldsymbol{\Psi}_1$ is general positive-definite and the site-within-wafer-level matrix $\boldsymbol{\Psi}_2$ is block-diagonal, with a 1×1 block corresponding to the variance of $b_{0i,j}$ and a 2×2 block corresponding to variance–covariance matrix of $[b_{1i,j}, b_{2i,j}]^T$.

Because its model function is nonlinear in the frequency ω, model (8.14) is actually an example of a multilevel nonlinear mixed-effects model. It

is treated as a linear model in §4.2.3 by holding ω fixed at a previously estimated value. Some of the disadvantages of this approach are that no precision can be attached to the estimate of ω and we cannot test if random effects are needed to account for variation in ω, either at the wafer level, or at the site-within-wafer level. In this section we treat (8.14) as a multilevel NLME model, allowing ω to be estimated with the other parameters and testing if random effects are needed to account for its variation in the data.

We can rewrite (8.14) in the usual two-stage NLME model formulation.

$$\begin{aligned} y_{ijk} &= \phi_{1ij} + \phi_{2ij} \cos(\omega v_k) + \phi_{3ij} \sin(\omega v_k) + \epsilon_{ijk}, \\ \phi_{1ij} &= (\beta_0 + b_{0i} + b_{0i,j}) + (\beta_1 + b_{1i} + b_{1i,j})\, v_k + (\beta_2 + b_{2i} + b_{2i,j})\, v_k^2, \\ \phi_{2ij} &= \beta_3, \qquad \phi_{3ij} = \beta_4, \end{aligned} \tag{8.15}$$

where the fixed effects, $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_4)$, the random effects, $\boldsymbol{b}_i$, $\boldsymbol{b}_{i,j}$, and the within-group errors, ϵ_{ijk}, are defined as in (8.14). To illustrate some of the multilevel capabilities in `nlme`, we initially fit model (8.15) with fixed $\omega = 4.5679$ as in §4.2.3. To get results that are comparable to the `fm5Wafer` fit in §4.2.3, we need to set the estimation method in `nlme` to `REML`.

```
> fm1Wafer.nlmeR <- nlme( current ~ A + B * cos(4.5679 * voltage) +
+   C * sin(4.5679 * voltage), data = Wafer,
+   fixed = list(A ~ voltage + voltage^2, B + C ~ 1),
+   random = list(Wafer = A ~ voltage + voltage^2,
+                 Site = pdBlocked(list(A~1, A~voltage+voltage^2-1))),
+   start = fixef(fm4Wafer), method = "REML")
> fm1Wafer.nlmeR
. . .
  Fixed: list(A ~ voltage + voltage^2, B + C ~ 1)
 A.(Intercept) A.voltage A.I(voltage^2)         B        C
       -4.2554    5.6224         1.2585 -0.095557 0.10435

Random effects:
 Formula: A ~ voltage + voltage^2 | Wafer
 Structure: General positive-definite
                    StdDev   Corr
 A.(Intercept) 0.131805 A.(In) A.vltg
     A.voltage 0.354743 -0.967
A.I(voltage^2) 0.049955  0.814 -0.935

 Composite Structure: Blocked

 Block 1: A.(Intercept)
 Formula: A ~ 1 | Site %in% Wafer
        A.(Intercept)
StdDev:      0.066563
```

```
Block 2: A.voltage, A.I(voltage^2)
Formula: A ~ voltage + voltage^2 - 1 | Site %in% Wafer
Structure: General positive-definite
                    StdDev   Corr
     A.voltage 0.2674083 A.volt
A.I(voltage^2) 0.0556444 -0.973
      Residual 0.0091086
. . .
```

The only difference in the multilevel model specification in `random` between `lme` and `nlme` is the use of one-sided formulas in the former and two-sided formulas in the latter. As expected, the estimation results for `fm1Wafer.nlme` are almost identical to the ones for `fm5Wafer`.

Because only five distinct voltages are used in the MOS circuit experiment, at most five different fixed effects can be used in a mixed-effects model fitted to the `Wafer` data. This is true in general: the number of *estimable* fixed effects in a mixed-effects model cannot exceed the number of distinct design points in the data used to fit it. Therefore, in order to allow the frequency ω to be estimated from the data, we need to drop at least one fixed effect from model (8.15).

Examining the fixed effects estimates in `fm1Wafer.nlme` we see that $\widehat{\beta}_3 \simeq -\widehat{\beta}_4$. We make the assumption that $\beta_3 = -\beta_4$ in (8.15), which, using the identity $\cos(\theta) - \sin(\theta) = \cos(\theta + \pi/4)$, gives the modified model

$$y_{ijk} = \phi_{1ij} + \phi_{2ij} \cos\left(\omega_{ij} v_k + \pi/4\right) + \epsilon_{ijk}. \tag{8.16}$$

To compensate for the restriction that $\beta_3 = -\beta_4$, we include random effects for ϕ_{2ij} at the wafer and site-within-wafer levels. Preliminary analyses of the modified model indicated that random effects for ω_{ij} are needed at the wafer level only. The corresponding model for the fixed and random effects is

$$\begin{aligned}
\phi_{1ij} &= (\beta_0 + b_{0i} + b_{0i,j}) + (\beta_1 + b_{1i} + b_{1i,j})\, v_k + (\beta_2 + b_{2i} + b_{2i,j})\, v_k^2, \\
\phi_{2ij} &= \beta_3 + b_{3i} + b_{3i,j}, \\
\omega_{ij} &= \beta_4^* + b_{4i}, \\
\boldsymbol{b}_i = \begin{bmatrix} b_{0i} \\ b_{1i} \\ b_{2i} \\ b_{3i} \\ b_{4i} \end{bmatrix} &\sim \mathcal{N}(0, \boldsymbol{\Psi}_1), \qquad \boldsymbol{b}_{i,j} = \begin{bmatrix} b_{0i,j} \\ b_{1i,j} \\ b_{2i,j} \\ b_{3i,j} \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Psi}_2), \\
\epsilon_{ijk} &\sim \mathcal{N}\left(0, \sigma^2\right).
\end{aligned} \tag{8.17}$$

Because of the large number of random effects at each grouping level, to make the estimation problem more numerically stable, we make the simplifying assumption that the random effects are independent. That is, we assume that $\boldsymbol{\Psi}_1$ and $\boldsymbol{\Psi}_2$ are diagonal matrices.

A maximum likelihood fit of model (8.16), with fixed and random-effects structures given in (8.17), is obtained with

```
> fm2Wafer.nlme <- nlme( current ~ A + B * cos(w * voltage + pi/4),
+   data = Wafer, fixed = list(A ~ voltage + voltage^2, B + w ~ 1),
+   random = list(Wafer = pdDiag(list(A ~ voltage + voltage^2,
+                                     B + w ~ 1)),
+                 Site = pdDiag(list(A ~ voltage+voltage^2, B ~ 1))),
+   start = c(fixef(fm1Wafer.nlme)[-5], 4.5679) )
> fm2Wafer.nlme
Nonlinear mixed-effects model fit by maximum likelihood
  Model: current ~ A + B * cos(w * voltage + pi/4)
  Data: Wafer
  Log-likelihood: 766.44
  Fixed: list(A ~ voltage + voltage^2, B + w ~ 1)
 A.(Intercept) A.voltage A.I(voltage^2)        B       w
       -4.2653    5.6329          1.256 -0.14069 4.5937

Random effects:
 Formula: list(A ~ voltage + voltage^2, B ~ 1, w ~ 1)
 Level: Wafer
 Structure: Diagonal
        A.(Intercept) A.voltage A.I(voltage^2)         B         w
StdDev:        0.1332   0.34134       0.048243 0.0048037 0.014629

 Formula: list(A ~ voltage + voltage^2, B ~ 1)
 Level: Site %in% Wafer
 Structure: Diagonal
        A.(Intercept) A.voltage A.I(voltage^2)         B  Residual
StdDev:      0.084082   0.30872       0.067259 0.0067264 0.0008428
. . .
```

Even though models (8.15) and (8.16) are not nested, they can compared using information criterion statistics. The `anova` method can be used for that, but we must first obtain a maximum likelihood fit of model (8.15).

```
> fm1Wafer.nlme <- update( fm1Wafer.nlmeR, method = "ML" )
> anova( fm1Wafer.nlme, fm2Wafer.nlme, test = F )
              Model df      AIC      BIC logLik
fm1Wafer.nlme     1 16 -1503.9 -1440.1 767.96
fm2Wafer.nlme     2 15 -1502.9 -1443.0 766.44
```

The more conservative BIC favors the model with fewer parameters, `fm2Wafer.nlme`, while the more liberal AIC favors the model with larger log-likelihood, `fm1Wafer.nlme`.

The `intervals` method is used to obtain confidence intervals on the fixed effects and the variance components.

```
> intervals( fm2Wafer.nlme )
Approximate 95% confidence intervals
```

```
Fixed effects:
                    lower      est.     upper
A.(Intercept) -4.35017 -4.26525 -4.18033
    A.voltage  5.40994  5.63292  5.85589
A.I(voltage^2)  1.22254  1.25601  1.28948
            B -0.14403 -0.14069 -0.13735
            w  4.58451  4.59366  4.60281

Random Effects:
 Level: Wafer
                       lower      est.     upper
sd(A.(Intercept)) 0.0693442 0.1331979 0.255849
    sd(A.voltage) 0.1715436 0.3413426 0.679214
sd(A.I(voltage^2)) 0.0222685 0.0482433 0.104516
            sd(B) 0.0022125 0.0048037 0.010430
            sd(w) 0.0078146 0.0146290 0.027385
 Level: Site
                       lower      est.     upper
sd(A.(Intercept)) 0.0664952 0.0840825 0.1063214
    sd(A.voltage) 0.2442003 0.3087244 0.3902973
sd(A.I(voltage^2)) 0.0532046 0.0672589 0.0850257
            sd(B) 0.0053128 0.0067264 0.0085162

Within-group standard error:
     lower      est.     upper
0.00066588 0.0008428 0.0010667
```

The fixed effects and the within-group standard error are estimated with more relative precision than the random-effects variance components. In the random-effects variance components, the *site-within-wafer* standard deviations are estimated with greater precision than the *wafer* standard deviations.

The plot of the within-group residuals versus voltage by wafer, displayed in Figure 8.23, and produced with

```
> plot( fm2Wafer.nlme, resid(.) ~ voltage | Wafer,
+       panel = function(x, y, ...) {
+                 panel.grid()
+                 panel.xyplot(x, y)
+                 panel.loess(x, y, lty = 2)
+                 panel.abline(0, 0)
+               } )                                    # Figure 8.23
```

does not reveal any periodic patterns as observed, for example, in Figure 4.27, indicating that the inclusion of a random effect for ω accounted successfully for variations in frequency among wafers.

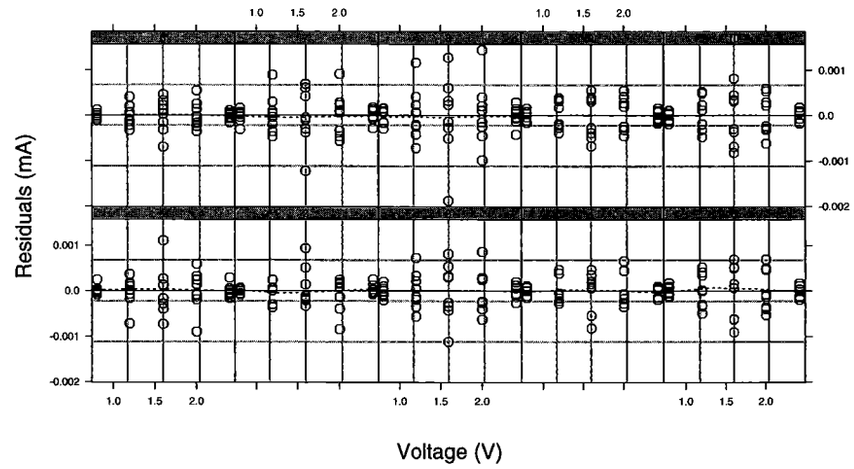

FIGURE 8.23. Scatter plots of within-group residuals versus voltage by wafer for the `fm2Wafer.nlme` fit. A `loess` smoother has been added to each panel to enhance the visualization of the residual pattern.

The normal plots of the within-group residuals and of the estimated site-within-wafer random effects, not shown here, do not indicate any violations of the NLME model assumptions.

8.3 Extending the Basic `nlme` Model

The extended nonlinear mixed-effects model with heteroscedastic, correlated within-group errors was introduced in §7.4.1. In this section, we describe the use of the `nlme` function for fitting such models. The use of variance functions for modeling heteroscedasticity in NLME models are discussed and illustrated in §8.3.1. Correlation structures for modeling within-group error dependence in NLME models are illustrated in §8.3.2. The `gnls` function to fit the extended nonlinear regression model presented in §7.5.1 is described and illustrated in §8.3.3, together with its associated class and methods.

8.3.1 Variance Functions in `nlme`

Variance functions are specified for an `nlme` model in the same way as for an `lme` model: through the `weights` argument. Any of the varFunc classes described in §5.2.1, and listed in Table 5.1, can also be used with `nlme`. The same diagnostic plots discussed in §5.2.1 for identifying within-group heteroscedasticity and assessing the adequacy of a variance function for lme objects, can also be used with nlme objects. We revisit the theophylline

example of §8.2.1 and the quinidine example of §8.2.2 to illustrate the use of variance functions in nlme.

Theophylline Kinetics

The plot of the standardized residuals versus the fitted values for the fitted object fm3Theo.nlme, displayed in Figure 8.13, suggests that the within-group variance increases with the concentration of theophylline.

The definition of the first-order open-compartment model (8.2) implies that the fitted value for the concentration at time $t = 0$ is $\widehat{c}_0 = 0$. Therefore, a *power* variance function, a natural candidate for this type of heteroscedastic pattern, cannot be used in this example, as the corresponding weights are undefined at $t = 0$. (Davidian and Giltinan (1995, §5.5, p. 145) argue that the observations at $t = 0$ do not add any information for the model and should be omitted from the data. We retain them here for illustration.) The *constant plus power* variance function, described in §5.2 and represented in the nlme library by the varConstPower class, accommodates the problem with $\widehat{c}_0 = 0$ by adding a constant to the power of the fitted value. In the theophylline example, the varConstPower variance function is expressed as $g(\widehat{c}_{ij}, \boldsymbol{\delta}) = \delta_1 + \widehat{c}_{ij}^{\,\delta_2}$. We incorporate it in the nlme fit using

```
> fm4Theo.nlme <- update( fm3Theo.nlme,
+    weights = varConstPower(power = 0.1) )
> fm4Theo.nlme
Nonlinear mixed-effects model fit by maximum likelihood
  Model: conc ~ SSfol(Dose, Time, lKe, lKa, lCl)
  Data: Theoph
  Log-likelihood: -167.68
  Fixed: list(lKe ~ 1, lKa ~ 1, lCl ~ 1)
     lKe     lKa     lCl
 -2.4538 0.43348 -3.2275

Random effects:
 Formula: list(lKa ~ 1, lCl ~ 1)
 Level: Subject
 Structure: Diagonal
           lKa     lCl Residual
StdDev: 0.6387 0.16979   0.3155

Variance function:
 Structure: Constant plus power of variance covariate
 Formula:  ~ fitted(.)
 Parameter estimates:
   const   power
 0.71966 0.31408
```

An initial value for the power parameter (δ_2) is specified in the call to the varConstPower constructor to avoid convergence problems associated with the default value power=0, for this example.

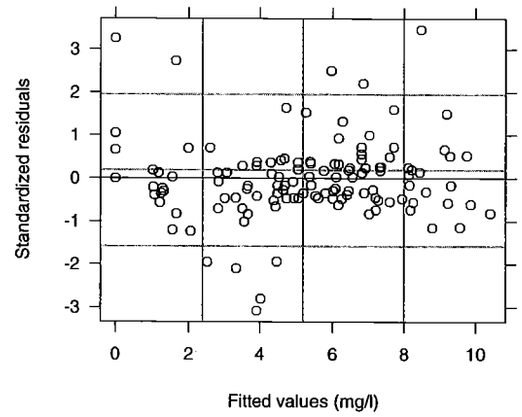

FIGURE 8.24. Scatter plot of standardized residuals versus fitted values for fm4Theo.nlme.

The anova method is used to assess the statistical significance of the variance function.

```
> anova( fm3Theo.nlme, fm4Theo.nlme )
             Model df    AIC    BIC  logLik   Test L.Ratio p-value
fm3Theo.nlme     1  6 366.04 383.34 -177.02
fm4Theo.nlme     2  8 351.35 374.41 -167.68 1 vs 2  18.694   1e-04
```

The small p-value for the likelihood ratio test indicates that the incorporation of the variance function in the model produced a significant increase in the log-likelihood. Both the AIC and the BIC also favor the fm4Theo.nlme fit. The plot of the standardized residuals versus the fitted values, shown in Figure 8.24, confirms the adequacy of the varConstPower variance function.

```
> plot( fm4Theo.nlme )                                    # Figure 8.24
```

Clinical Study of Quinidine

Figure 8.25 presents the scatter plot of the standardized residuals versus the fitted values, corresponding to the final model for the quinidine data in §8.2.2, represented by the fitted object fm4Quin.nlme.

```
> ## xlim used to hide an unusually high fitted value and enhance
> ## visualization of the heteroscedastic pattern
> plot( fm4Quin.nlme, xlim = c(0, 6.2) )                  # Figure 8.25
```

As reported in previous analyses of the quinidine data (Davidian and Giltinan, 1995, §9.3), and indicated in Figure 8.25, the variance of the within-group errors appears to increase with the quinidine concentration. The fitted values do not get sufficiently close to zero to cause problems in

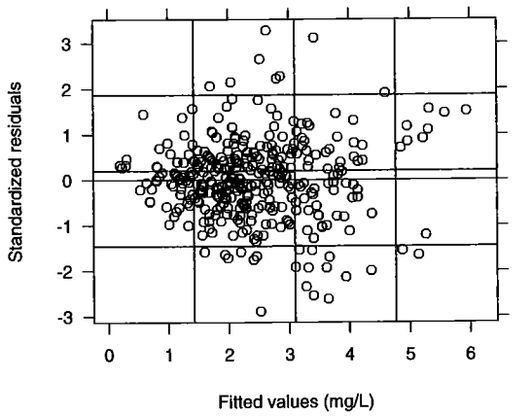

FIGURE 8.25. Scatter plot of standardized residuals versus fitted values for `fm4Quin.nlme`.

the calculation of weights for the power variance function and we choose the varPower class to model the within-group heteroscedasticity.

```
> fm5Quin.nlme <- update( fm4Quin.nlme, weights = varPower() )
> summary( fm5Quin.nlme )
. . .
Random effects:
 Formula: list(lV ~ 1, lCl ~ 1)
 Level: Subject
 Structure: Diagonal
            lV lCl.(Intercept) Residual
StdDev: 0.32475         0.25689  0.25548

Variance function:
 Structure: Power of variance covariate
 Formula:  ~ fitted(.)
 Parameter estimates:
   power
 0.96616
Fixed effects: list(lCl ~ glyco + Creatinine + Weight, lKa + lV ~ 1)
                  Value Std.Error  DF t-value p-value
lCl.(Intercept)  2.7076   0.15262 220  17.741  <.0001
      lCl.glyco -0.4110   0.04487 220  -9.161  <.0001
 lCl.Creatinine  0.1292   0.03494 220   3.696  0.0003
     lCl.Weight  0.0033   0.00179 220   1.828  0.0689
            lKa -0.4269   0.25518 220  -1.673  0.0958
             lV  5.3700   0.08398 220  63.941  <.0001
. . .
```


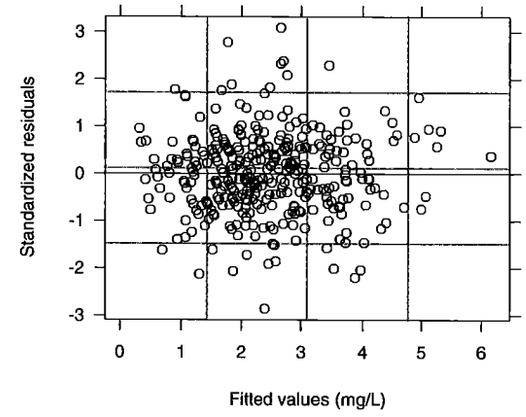

FIGURE 8.26. Scatter plot of standardized residuals versus fitted values for `fm5Quin.nlme`.

```
> anova( fm4Quin.nlme, fm5Quin.nlme )
             Model df    AIC    BIC  logLik   Test L.Ratio p-value
fm4Quin.nlme     1  9 870.94 905.94 -426.47
fm5Quin.nlme     2 10 812.13 851.01 -396.06 1 vs 2  60.813  <.0001
```

The incorporation of the power variance function in the NLME model for the quinidine data produced a significant increase in the log-likelihood, as evidenced by the small p-value for the likelihood ratio statistics. The plot of the standardized residuals versus fitted values, displayed in Figure 8.26, gives further evidence of the adequacy of the variance function model.

```
> plot( fm5Quin.nlme, xlim = c(0, 6.2) )                # Figure 8.26
```

8.3.2 Correlation Structures in `nlme`

Just as in `lme` and `gls`, correlation structures are specified in `nlme` through the `correlation` argument. All of the corStruct classes described in §5.3.3, and listed in Table 5.3, can be used with `nlme`. As in the linear case, we investigate the need for within-group correlation structures in the NLME model by looking at plots of the empirical autocorrelation function (ACF) and the sample semivariogram, and assess the adequacy of a particular corStruct class by examining plots of the normalized residuals. We revisit the ovary example of §5.3.4 to illustrate the use of correlation structures with `nlme`.

Counts of Ovarian Follicles

The ovary example was analyzed in §5.3.4 using an LME model, by assuming that the number of ovarian follicles was a periodic function of time with

known frequency equal to 1. A more general model formulation assumes that the frequency is an unknown parameter to be estimated from the data, with a possible random effect associated with it. Because the frequency enters the model nonlinearly, this becomes an NLME model, represented as

$$y_{ij} = \phi_{0i} + \phi_{1i} \sin(2\pi\phi_{2i} t_{ij}) + \phi_{3i} \cos(2\pi\phi_{2i} t_{ij}) + \epsilon_{ij}, \tag{8.18}$$

where y_{ij} represents the number of follicles observed for mare i at time t_{ij}, ϕ_{0i}, ϕ_{1i}, and ϕ_{3i} represent the intercept and the terms defining the amplitude and the phase of the cosine wave for mare i, ϕ_{2i} is the frequency of cosine wave for mare i, and ϵ_{ij} is the within-group error. We initially assume that the within-group errors are independently distributed as $\mathcal{N}(0, \sigma^2)$.

The final LME model used to fit the `Ovary` data in §5.3.4 used independent random effects for the ϕ_{0i} and ϕ_{1i} coefficients in model (8.18). We use this as a starting point for the random-effects model, incorporating an extra independent random effect for the frequency ϕ_{2i}. The fixed- and random-effects models corresponding to (8.18) are then expressed as

$$\phi_{0i} = \beta_0 + b_{0i}, \quad \phi_{1i} = \beta_1 + b_{1i}, \quad \phi_{2i} = \beta_2 + b_{2i}, \quad \phi_{3i} = \beta_3,$$
$$\boldsymbol{b}_i = \begin{bmatrix} b_{0i} \\ b_{1i} \\ b_{2i} \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Psi}), \tag{8.19}$$

where $\boldsymbol{\Psi}$ is diagonal. The random effects, $\boldsymbol{b}_i$, are assumed to be independent for different i and to be independent of the within-group errors.

We fit model (8.18), with fixed- and random-effects structures given by (8.19), using

```
> fm1Ovar.nlme <- nlme(follicles ~ A + B * sin(2 * pi * w * Time) +
+   C * cos(2 * pi * w *Time), data = Ovary,
+   fixed = A + B + C + w ~ 1, random = pdDiag(A + B + w ~ 1),
+   start = c(fixef(fm5Ovar.lme), 1))
> fm1Ovar.nlme
Nonlinear mixed-effects model fit by maximum likelihood
  Model: follicles ~ A + B * sin(2 * pi * w * Time) +
          C * cos(2 * pi * w * Time)
  Data: Ovary
  Log-likelihood: -803.83
  Fixed: A + B + C + w ~ 1
      A      B       C       w
 12.184 -3.376 -1.6812 0.93605

Random effects:
 Formula: list(A ~ 1, B ~ 1, w ~ 1)
 Level: Mare
 Structure: Diagonal
             A        B         w Residual
StdDev: 2.9051 2.0061 0.073598   2.9387
. . .
```


The estimated fixed effects for `fm5Ovar.lme` are used as initial values for β_0, β_1, and β_3.

As mentioned in §5.3.4, the observations in the `Ovary` data were collected at equally spaced calendar times, and then converted to an ovulation cycle scale. Therefore, the empirical ACF can be used to investigate the correlation at different lags. The `ACF` method can also be used with `nlme` objects.

```
> ACF( fm1Ovar.nlme )
   lag        ACF
1    0  1.0000000
2    1  0.3110027
3    2  0.0887701
4    3 -0.0668554
5    4 -0.0314934
6    5 -0.0810381
7    6 -0.0010647
8    7  0.0216463
9    8  0.0137578
10   9  0.0097497
11  10 -0.0377027
12  11 -0.0741284
13  12 -0.1504872
14  13 -0.1616297
15  14 -0.2395797
```

Because they are based on fewer residual pairs, empirical autocorrelations at larger lags are less reliable. We can control the number of lags calculated in `ACF` using the `maxLag` argument. We use it in the plot of empirical ACF, displayed in Figure 8.27 and obtained with

```
> plot( ACF(fm1Ovar.nlme,  maxLag = 10),
+       alpha = 0.05 )                                  # Figure 8.27
```

Figure 8.27 shows that only the lag-1 autocorrelation is significant at the 5% level, but the lag-2 autocorrelation, which is approximately equal to the square of the lag-1 autocorrelation, is nearly significant. This suggests two different candidate correlation structures for modeling the within-group error covariance structure: $AR(1)$ and $MA(2)$. The two correlation models are not nested, but can be compared using using the information criteria provided by the `anova` method, AIC and BIC. The empirical lag-1 autocorrelation is used as a starting value for the `corAR1` coefficient.

```
> fm2Ovar.nlme <- update( fm1Ovar.nlme, corr = corAR1(0.311) )
> fm3Ovar.nlme <- update( fm1Ovar.nlme, corr = corARMA(p=0, q=2) )
> anova( fm2Ovar.nlme, fm3Ovar.nlme, test = F )
             Model df    AIC    BIC  logLik
fm2Ovar.nlme     1  9 1568.3 1601.9 -775.15
fm3Ovar.nlme     2 10 1572.1 1609.4 -776.07
```

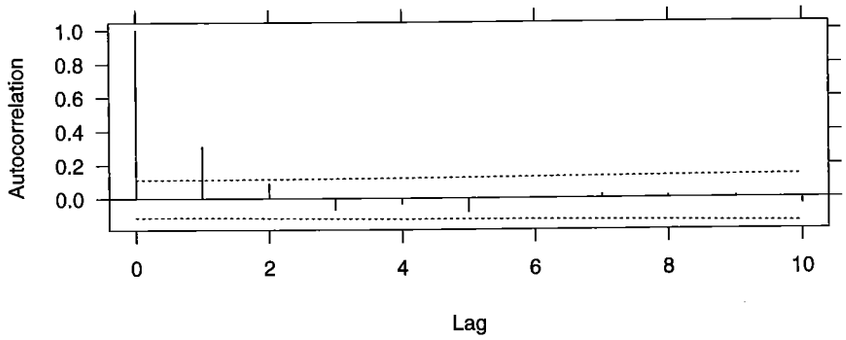

FIGURE 8.27. Empirical autocorrelation function corresponding to the standardized residuals of the `fm1Ovar.nlme` fitted object.

The $AR(1)$ model uses one fewer parameter than the $MA(2)$ model to give a larger log-likelihood and hence is the preferred model by both AIC and BIC.

The approximate 95% confidence intervals for the variance components in `fm2Ovar.nlme`, obtained with

```
> intervals( fm2Ovar.nlme )
. . .
 Random Effects:
  Level: Mare
           lower      est.      upper
sd(A) 1.5465e+00 3.3083316 7.0772e+00
sd(B) 3.4902e-01 1.4257894 5.8245e+00
sd(w) 2.4457e-89 0.0020967 1.7974e+83
. . .
```

indicate that there is no precision in the estimate of the standard deviation for the frequency ϕ_{2i} and little precision in the estimate of the standard deviation for ϕ_{1i}. The incorporation of the within-group autocorrelation structure into the NLME model seems to have reduced the need for random effects in the model. This is not uncommon: the random-effects model and the within-group correlation model compete with each other, in the sense that fewer random effects may be needed when within-group correlation structures are present, and viceversa. We test if the two variance components can be dropped from the model using `anova`.

```
> fm4Ovar.nlme <- update( fm2Ovar.nlme, random = A ~ 1)
> anova( fm2Ovar.nlme, fm4Ovar.nlme )
             Model df    AIC    BIC  logLik   Test L.Ratio p-value
fm2Ovar.nlme     1  9 1568.3 1601.9 -775.15
fm4Ovar.nlme     2  7 1565.2 1591.4 -775.62 1 vs 2 0.94001   0.625
```

The high p-value for the likelihood ratio test suggests that the two models give essentially equivalent fits so the simpler model is preferred.

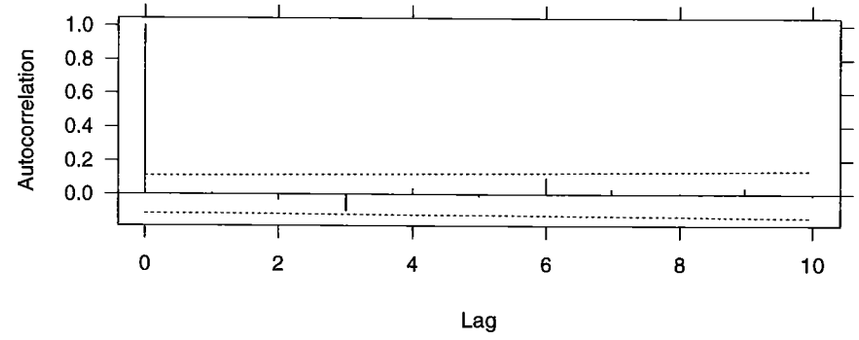

FIGURE 8.28. Empirical autocorrelation function corresponding to the normalized residuals of the `fm1Ovar.nlme` fitted object.

An alternative, "intermediate" model between the $AR(1)$ and $MA(2)$ correlation structures is the $ARMA(1,1)$ model. This structure has an exponentially decaying ACF for lags ≥ 2, but allows greater flexibility in the lag-1 autocorrelation. Because the $AR(1)$ model is nested within the $ARMA(1,1)$ model, they can be compared via a likelihood ratio test.

```
> fm5Ovar.nlme <- update( fm4Ovar.nlme, corr = corARMA(p=1, q=1))
> anova( fm4Ovar.nlme, fm5Ovar.nlme )
             Model df    AIC    BIC  logLik   Test L.Ratio p-value
fm4Ovar.nlme     1  7 1565.2 1591.4 -775.62
fm5Ovar.nlme     2  8 1562.1 1592.0 -773.07 1 vs 2  5.1134  0.0237
```

The $ARMA(1,1)$ gives a significantly better representation of the within-group correlation, as indicated by the small p-value for the likelihood ratio test.

The plot of the empirical ACF of the normalized residuals, displayed in Figure 8.28, attests the the adequacy of the $ARMA(1,1)$ model for the `Ovary` data. No significant autocorrelations are detected, indicating that the normalized residuals behave like uncorrelated noise, as expected under the appropriate correlation model.

```
> plot( ACF(fm5Ovar.nlme,  maxLag = 10, resType = "n"),
+       alpha = 0.05 )                                  # Figure 8.28
```

It is illustrative, at this point, to compare the `nlme` fit represented by `fm5Ovar.nlme` to the `lme` fit corresponding to `fm5Ovar.lme` of §5.3.4. To have comparable log-likelihoods, we need to first obtain a maximum likelihood version of `fm5Ovar.lme`.

```
> fm5Ovar.lmeML <- update( fm5Ovar.lme, method = "ML" )
> intervals( fm5Ovar.lmeML )
. . .
 Random Effects:
```

```
  Level: Mare
                         lower      est.    upper
      sd((Intercept)) 1.2933904 2.4565297 4.665674
sd(sin(2 * pi * Time)) 0.1164408 0.8521297 6.236003
. . .
```

The wide confidence interval for the standard deviation of the random effect corresponding to `sin(2*pi*Time)` indicates that the fit is not very sensitive to the value of this coefficient and perhaps it could be eliminated from the model. We test this assumption using the likelihood ratio test.

```
> fm6Ovar.lmeML <- update( fm5Ovar.lmeML, random = ~1 )
> anova( fm5Ovar.lmeML, fm6Ovar.lmeML )
              Model df    AIC    BIC  logLik   Test L.Ratio
fm5Ovar.lmeML     1  8 1562.7 1592.6 -773.37
fm6Ovar.lmeML     2  7 1561.0 1587.1 -773.51 1 vs 2 0.28057
              p-value
fm5Ovar.lmeML
fm6Ovar.lmeML  0.5963
```

The large p-value for the test indicates that the two models are essentially equivalent so the simpler model with a single random intercept is preferred.

The LME model represented by `fm6Ovar.lmeML` is nested within the NLME model represented by `fm5Ovar.nlme`, corresponding to the case of $\beta_2 = 1$. Hence, we can test the assumption that the frequency of the ovulation cycle is equal to 1 using the likelihood ratio test.

```
> anova( fm6Ovar.lmeML, fm5Ovar.nlme )
              Model df    AIC    BIC  logLik   Test L.Ratio
fm6Ovar.lmeML     1  7 1561.0 1587.1 -773.51
 fm5Ovar.nlme     2  8 1562.1 1592.0 -773.07 1 vs 2 0.87881
              p-value
fm6Ovar.lmeML
 fm5Ovar.nlme  0.3485
```

There is no significant evidence that $\beta_2 \neq 1$. This conclusion is also supported by the approximate confidence interval for β_2, which contains 1.

```
> intervals( fm5Ovar.nlme, which = "fixed" )
Approximate 95% confidence intervals

 Fixed effects:
     lower      est.     upper
A 10.37917  12.15566  13.932151
B -3.91347  -2.87191  -1.830353
C -3.07594  -1.56879  -0.061645
w  0.81565   0.93111   1.046568
```


8.3.3 *Fitting Extended Nonlinear Regression Models with gnls*

The general formulation of the extended nonlinear regression model, as well as the estimation methods used to fit it, have been described in §7.5.1 and §7.5.2. In this section, we present and illustrate the capabilities available in the nlme library for fitting and analyzing such models.

The `gnls` function fits the extended nonlinear regression model (7.34) using maximum likelihood. It can be viewed either as a version of `nlme` without the argument `random`, or as a version of `nls` with the arguments `weights` and `correlation`. Several arguments are available in `gls`, but typical calls are of the form

```
gnls(model, data, params, start, correlation)   # correl. errors
gnls(model, data, params, start, weights)     # heterosc. errors
gnls(model, data, params, start, correlation, weights)  # both
```

The first argument, `model`, is a two-sided nonlinear formula specifying the model for the expected value of the response. It uses the same syntax as the `model` argument to `nlme`. `Correlation` and `weights` are used as in `lme`, `gls`, and `nlme` to define, respectively, the correlation model and the variance function model for the error term. `Data` specifies a data frame in which the variables named in `model`, `correlation`, and `weights` can be evaluated. The parameters in the model are specified via the `params` argument, which can be either a two-sided linear formula, or a list of two-sided linear formulas. The syntax for `params` is identical to that of the `fixed` argument to `nlme`. Starting values for the model parameters are specified in `start`, which uses the same syntax as the argument with the same name to `nlme`. Starting values need not be given when the model function defined in `model` is a self-starting model and the the right-hand side of the parameter formulas in `param` do not include any covariates.

The fitted object returned by `gnls` is of class gnls, for which several methods are available to display, plot, update, and further explore the estimation results. Table 8.5 lists the most important gnls methods. The syntax of the gnls methods is identical to the syntax of the gls methods, described in §5.4. In fact, with the exception of `coef`, `formula`, `logLik`, `predict`, and `update`, all methods listed in Table 8.5 are common to both classes.

The use of the `gnls` function and its associated methods is described and illustrated through the re-analysis of the hemodialyzer example introduced in §5.2.2.

High-Flux Hemodialyzer Ultrafiltration Rates

The hemodialyzer ultrafiltration rates data were analyzed in §5.2.2 and §5.4 using an empirical polynomial model suggested by Littell et al. (1996). The model originally proposed for these data by Vonesh and Carter (1992) is

TABLE 8.5. Main gnls methods.

ACF	empirical autocorrelation function of residuals
anova	likelihood ratio or Wald-type tests
augPred	predictions augmented with observed values
coef	estimated coefficients for expected response model
fitted	fitted values
intervals	confidence intervals on model parameters
logLik	log-likelihood at convergence
plot	diagnostic Trellis plots
predict	predicted values
print	brief information about the fit
qqnorm	normal probability plots
resid	residuals
summary	more detailed information about the fit
update	update the gnls fit
Variogram	semivariogram of residuals

an asymptotic regression model with an offset, identical to the one used for the CO_2 uptake data in §8.2.2. The model for the expected ultrafiltration rate y at transmembrane pressure x is written as

$$\mathrm{E}[y] = \phi_1 \{1 - \exp[-\exp(\phi_2)(x - \phi_3)]\}. \tag{8.20}$$

The parameters in model (8.20) have a physiological interpretation: ϕ_1 is the maximum ultrafiltration rate that can be attained, ϕ_2 is the logarithm of the hydraulic permeability transport rate, and ϕ_3 is the transmembrane pressure required to offset the oncotic pressure.

Vonesh and Carter (1992) suggest using different parameters in (8.20) for each blood flow rate level. We use the self-starting function SSasympOff and the nlsList function to investigate which parameters in the asymptotic regression model (8.20) depend on the blood flow rate.

```
> fm1Dial.lis <-
+   nlsList( rate ~ SSasympOff(pressure, Asym, lrc, c0) | QB,
+            data = Dialyzer )
> fm1Dial.lis
. . .
Coefficients:
      Asym     lrc      c0
200 44.988 0.76493 0.22425
300 62.217 0.25282 0.22484
```

The coefficient estimates suggest that Asym (ϕ_1) and lrc ($\log \phi_2$) depend on the blood flow rate level, but c0 (ϕ_3) does not. The plot of the individual

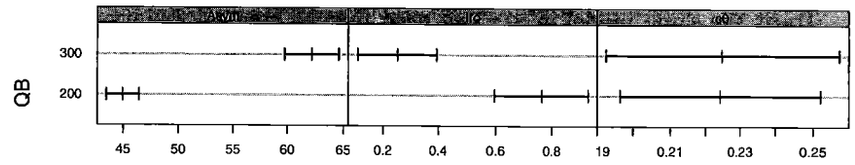

FIGURE 8.29. Ninety-five percent confidence intervals on the asymptotic regression model parameters for each level of blood flow rate (QB) in the dialyzer data.

confidence intervals in Figure 8.29 confirms that only Asym and lrc vary with blood flow level.

```
> plot( intervals(fm1Dial.lis) )                    # Figure 8.29
```

The ultrafiltration rate y_{ij} at the jth transmembrane pressure x_{ij} for the ith subject is represented by the nonlinear model

$$y_{ij} = (\phi_1 + \gamma_1 Q_i)\{1 - \exp[-\exp(\phi_2 + \gamma_2 Q i)(x_{ij} - \phi_3)]\} + \epsilon_{ij}, \tag{8.21}$$

where Q_i is a binary variable taking values -1 for 200 dl/min hemodialyzers and 1 for 300 dl/min hemodialyzers; ϕ_1, ϕ_2, and ϕ_3 are, respectively, the asymptotic ultrafiltration rate, the log-transport rate, and the transmembrane pressure offset averaged over the levels of Q; γ_i is the blood flow effect associated with the coefficient ϕ_i; and ϵ_{ij} is the error term, initially assumed to be independently distributed $\mathcal{N}(0, \sigma^2)$ random variables.

The nonlinear model (8.21) can be fitted with nls, but it is easier to express the dependency of the asymptote and the log-rate on the blood flow rate using gnls. The average of the fm1Dial.lis coefficients are used as the initial estimates for ϕ_1, ϕ_2, and ϕ_3, while the half differences between the first two coefficients are used as initial estimates for γ_1 and γ_2.

```
> fm1Dial.gnls <- gnls( rate ~ SSasympOff(pressure, Asym, lrc, c0),
+   data = Dialyzer, params = list(Asym + lrc ~ QB, c0 ~ 1),
+   start = c(53.6, 8.6, 0.51, -0.26, 0.225) )
> fm1Dial.gnls
Generalized nonlinear least squares fit
  Model: rate ~ SSasympOff(pressure, Asym, lrc, c0)
  Data: Dialyzer
  Log-likelihood: -382.65

Coefficients:
 Asym.(Intercept) Asym.QB lrc.(Intercept)   lrc.QB      c0
           53.606    8.62         0.50874 -0.25684 0.22449

Degrees of freedom: 140 total; 135 residual
Residual standard error: 3.7902
```

To fit the same model in nls, we need first to create a binary variable representing Q in (8.21) and then include all coefficients explicitly in the model formula.

```
> Dialyzer$QBcontr <- 2 * (Dialyzer$QB == 300) - 1
> fm1Dial.nls <-
+   nls( rate ~ SSasympOff(pressure, Asym.Int + Asym.QB * QBcontr,
+   lrc.Int + lrc.QB * QBcontr, c0), data = Dialyzer,
+   start = c(Asym.Int = 53.6, Asym.QB = 8.6, lrc.Int = 0.51,
+   lrc.QB = -0.26, c0 = 0.225) )
> summary( fm1Dial.nls )

Formula: rate ~ SSasympOff(pressure, Asym.Int + Asym.QB * QBcontr,
          lrc.Int + lrc.QB * QBcontr, c0)

Parameters:
           Value Std. Error  t value
Asym.Int  53.60660   0.705409  75.9937
 Asym.QB   8.61999   0.679240  12.6906
 lrc.Int   0.50872   0.055233   9.2105
  lrc.QB  -0.25683   0.045021  -5.7047
      c0   0.22448   0.010623  21.1318

Residual standard error: 3.79022 on 135 degrees of freedom
> logLik( fm1Dial.nls )
[1] -382.65
```

As expected, the results are nearly identical.

The plot method is the primary tool for assessing the quality of a gnls fit. It uses the same syntax as the other plot methods in the nlme library. For example, the plot of the residuals versus the transmembrane pressure, shown in Figure 8.30 and obtained with

```
> plot(fm1Dial.gnls, resid(.) ~ pressure, abline = 0)  # Figure 8.30
```

indicates that the error variability increases with the transmembrane pressure. This heteroscedastic pattern is also observed in the linear model fits of the hemodialyzer data, presented in §5.2.2 and §5.4.

As in the previous analyses of the hemodialyzer data presented in §5.2.2 and §5.4, the power variance function, represented in nlme by the varPower class, is used to model the heteroscedasticity in the ultrafiltration rates.

```
> fm2Dial.gnls <- update( fm1Dial.gnls,
+                         weights = varPower(form = ~ pressure) )
> anova( fm1Dial.gnls, fm2Dial.gnls)
             Model df    AIC    BIC  logLik   Test L.Ratio p-value
fm1Dial.gnls     1  6 777.29 794.94 -382.65
fm2Dial.gnls     2  7 748.47 769.07 -367.24 1 vs 2  30.815  <.0001
```

As expected, the likelihood ratio test strongly rejects the assumption of homoscedasticity. The plot of the standard residuals for fm2Dial.gnls

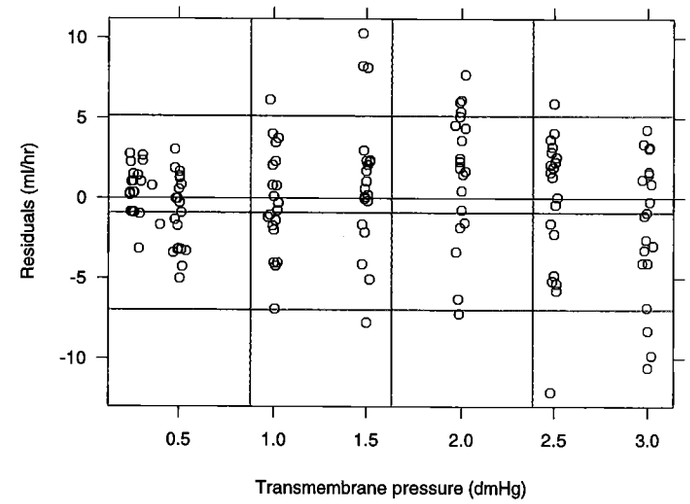

FIGURE 8.30. Plot of residuals versus transmembrane pressure for the homoscedastic fitted object fm1Dial.gls.

versus pressure, shown in Figure 8.31, indicates that the power variance function successfully models the heteroscedasticity in the data.

The hemodialyzer ultrafiltration rates measurements made sequentially on the same subject are correlated. Random effects can be used in an NLME model to account for the within-group correlation, but we choose here to model the within-subject dependence directly by incorporating a correlation structure for the error term in the gnls model. Because the measurements are equally spaced in time, the empirical autocorrelation function can be used to investigate the within-subject correlation. The ACF method is used to obtain the empirical ACF, with the time covariate and the grouping factor specified via the form argument.

```
> ACF( fm2Dial.gnls, form = ~ 1 | Subject )
  lag      ACF
1   0  1.00000
2   1  0.71567
3   2  0.50454
4   3  0.29481
5   4  0.20975
6   5  0.13857
7   6 -0.00202
```

The empirical ACF values confirm the within-group correlation and indicates that the correlation decreases with lag. As usual, it is more informative to look at a plot of the empirical ACF, displayed in Figure 8.32 and obtained with

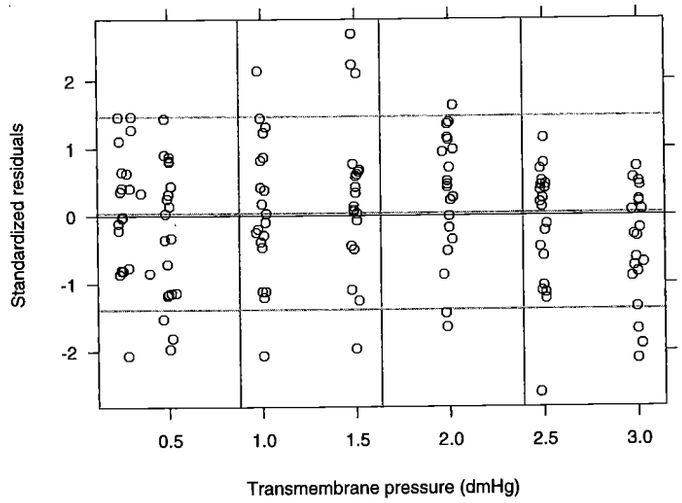

FIGURE 8.31. Plot of standardized residuals versus transmembrane pressure for the heteroscedastic fitted object `fm2Dial.gnls`.

```
> plot( ACF( fm2Dial.gnls, form = ~ 1 | Subject),
+       alpha = 0.05 )                                  # Figure 8.32
```

The autocorrelation pattern in Figure 8.32 suggests that an $AR(1)$ model, represented in nlme by the corAR1 class, may be appropriate to describe the within-group correlation.

```
> fm3Dial.gnls <-
+  update(fm2Dial.gnls, corr = corAR1(0.716, form = ~ 1 | Subject))
> fm3Dial.gnls
. . .
Coefficients:
 Asym.(Intercept) Asym.QB lrc.(Intercept)   lrc.QB      c0
          55.111  8.1999         0.37193 -0.16974 0.21478

Correlation Structure: AR(1)
 Formula: ~ 1 | Subject
 Parameter estimate(s):
    Phi
 0.7444
Variance function:
 Structure: Power of variance covariate
 Formula:  ~ pressure
 Parameter estimates:
  power
 0.5723
Degrees of freedom: 140 total; 135 residual
Residual standard error: 3.1844
```


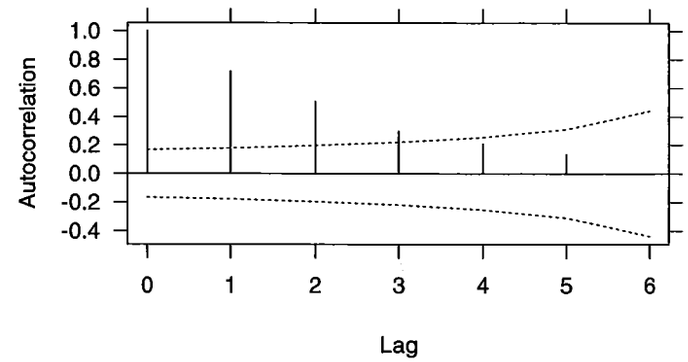

FIGURE 8.32. Empirical autocorrelation function corresponding to the standardized residuals of the `fm2Dial.gnls` fitted object.

The lag-1 empirical autocorrelation is used as initial value for the `corAR1`.

The variability in the estimates is assessed with the `intervals` method.

```
> intervals( fm3Dial.gnls )
. . .
 Correlation structure:
      lower   est.   upper
Phi 0.55913 0.7444 0.85886
. . .
```

The confidence interval on the autocorrelation parameter is bounded away from zero, suggesting that the $AR(1)$ model provides a significantly better fit. We confirm this with the likelihood ratio test.

```
> anova( fm2Dial.gnls, fm3Dial.gnls )
             Model df    AIC    BIC  logLik   Test L.Ratio p-value
fm2Dial.gnls     1  7 748.47 769.07 -367.24
fm3Dial.gnls     2  8 661.04 684.58 -322.52 1 vs 2  89.433  <.0001
```

The plot of the empirical ACF for the normalized residuals corresponding to `fm3Dial.gnls`, displayed in Figure 8.33, does not show any significant correlations, indicating that the $AR(1)$ adequately represents the within-subject dependence in the `gnls` model for the hemodialyzer data.

The plot of the standardized residuals versus the transmembrane pressure in Figure 8.34 suggests a certain lack-of-fit for the asymptotic regression model (8.21): the residuals for the highest two transmembrane pressures are predominantly negative. This is consistent with the plot of the hemodialyzer data, shown in Figure 5.1, and also with Figure 3 in Vonesh and Carter (1992), which indicate that, for many subjects, the ultrafiltration rates decrease for the highest two transmembrane pressures. The asymptotic regression model is monotonically increasing in the transmembrane pressure and cannot properly accommodate the nonmonotonic behavior of the ultrafiltration rates.

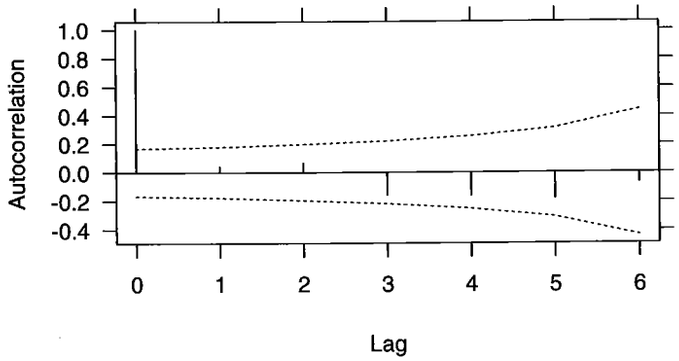

FIGURE 8.33. Empirical autocorrelation function for the normalized residuals corresponding to the `fm3Dial.gnls` fitted object.

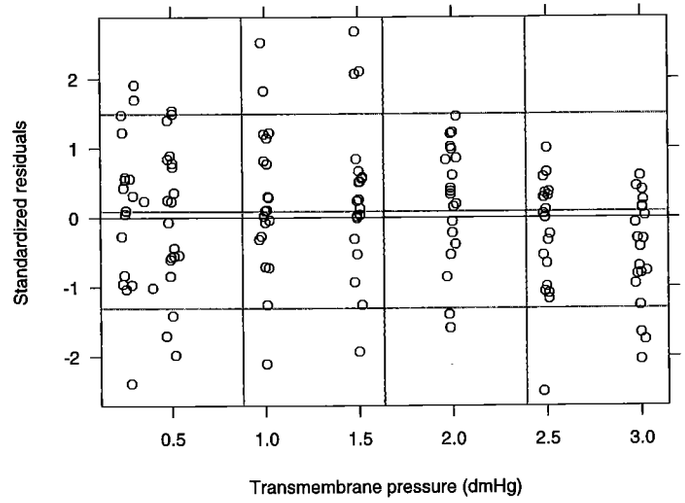

FIGURE 8.34. Plot of standardized residuals versus transmembrane pressure for the `fm3Dial.gnls` fitted object.

The gnls model corresponding to `fm3Dial.gnls` may be compared to the best models obtained for the `Dialyzer` data in §5.2.2 and §5.4, corresponding, respectively, to the objects `fm2Dial.lme` and `fm3Dial.gls`. To have comparable fits, we need to first obtain ML versions of `fm2Dial.lme` and `fm3Dial.gls`.

```
> fm2Dial.lmeML <- update( fm2Dial.lme, method = "ML")
> fm3Dial.glsML <- update( fm3Dial.gls, method = "ML")
```

As the models are not nested, only the information criterion statistics can be compared.

```
> anova(  fm2Dial.lmeML, fm3Dial.glsML, fm3Dial.gnls, test = F )
             Model df    AIC    BIC  logLik
fm2Dial.lmeML     1 18 651.75 704.70 -307.87
fm3Dial.glsML     2 13 647.56 685.80 -310.78
 fm3Dial.gnls     3  8 661.04 684.58 -322.52
```

The more conservative BIC favors the `fm3Dial.gnls` model because of the fewer number of parameters it uses; the more liberal AIC favors `fm3Dial.glsML` because of its larger log-likelihood value. In practice, the choice of the "best model" should take into account other factors besides the information criteria, such as the interpretability of the parameters.

8.4 Chapter Summary

This chapter describes the nonlinear modeling capabilities available in the nlme library. A brief review of the nonlinear least-squares function `nls` in S is presented and self-starting models for automatically producing starting values for the coefficients in a nonlinear model are introduced and illustrated. The `nlsList` function for fitting separate nonlinear regression models to data partitioned according to the levels of a grouping factor is described and its use for model building of nonlinear mixed-effects models illustrated.

Nonlinear mixed-effects models are fitted with the `nlme` function. Data from several real-life applications are used to illustrate the various capabilities available in `nlme` for fitting and analyzing single and multilevel NLME models. Variance functions and correlation structures to model the within-group variance–covariance structure are used with `nlme` in the exact same way as with `lme`, the linear mixed-effects modeling function. Several examples are used to illustrate the use of varFunc and corStruct classes with `nlme`.

A new modeling function, `gnls`, for fitting the extended nonlinear model with heteroscedastic, correlated errors is introduced. The `gnls` function can be regarded as an extended version of `nls` which allows the use of varFunc and corStruct objects to model the error variance–covariance structure, or

as a simplified version of `nlme`, without random effects. The hemodialyzer example is used to illustrate the use of `gnls` and its associated methods.

Exercises

1. Plots of the `DNase` data in §3.3.2 and in Appendix A.7 suggest a sigmoidal relationship between the `density` and `log(conc)`. Because there are no data points at sufficiently high DNase concentrations to define the upper part of the curve, the sigmoidal relationship is only suggested and is not definite.

   A common model for this type of assay data is the four-parameter logistic model available as `SSfpl` (Appendix C.6).

   (a) Create separate fits of the `SSfpl` model to each run using `nlsList`. Note that the display formula, `density ~conc | Run`, defines the primary covariate as `conc` but the model should be fit as a function of `log(conc)`. You will either need to give explicit arguments to the `SSfpl` function or define a new groupedData object with a formula based on `log(conc)`.

   (b) Examine the plot of the residuals versus the fitted values from this `nlsList` fit. Does the plot indicate nonconstant variance?

   (c) Fit an NLME model to these data using random effects for each model parameter and a general positive-definite Ψ. Perform the usual checks on the model using diagnostic plots, summaries, and intervals. Can the confidence intervals on the variance–covariance parameters be evaluated?

   (d) Davidian and Giltinan (1995, §5.2.4) conclude from a similar analysis that the variance of the ϵ_{ij} increases with increasing optical density. They estimate a variance function that corresponds to the `varPower` variance function. Update the previous fit by adding `weights = varPower()`. Does this updated model converge? If not, change the definition of the random effects so that Ψ is diagonal.

   (e) Compare the model fit with the `varPower` variance function to one fit without it. Note that if you have fit the model with the variance function by imposing a diagonal Ψ, you should refit the model without the variance function but with a diagonal Ψ before comparing. Does the addition of the `varPower` variance function make a significant contribution to the model?

   (f) Examine the confidence intervals on your best-fitting model to this point. Are there parameters that can be modeled as purely fixed effects? (In our analysis the standard deviation of the random effect for the `xmid` parameter turned out to be negligible.) If you determine that some random effects are no longer needed, remove them and refit the model.

   (g) If you have modified the model to reduce the number of random effects, refit with a general positive-definite Ψ rather than a diagonal Ψ. Does this fit converge? If so, compare the fits with diagonal Ψ and general positive-definite Ψ.

   (h) Write a report on the analysis, including data plots and diagnostic plots where appropriate.

2. As shown in Figure 3.4 (p. 107), the relationship between `deltaBP` and `log(dose)` in the phenybiguanide data `PBG` is roughly sigmoidal. There is a strong indication that the effect of the `Treatment` is to shift the curve to the right.

   (a) Fit separate four-parameter logistic models (`SSfpl`, Appendix C.6) to the data from each `Treatment` within each `Rabbit` using `nlsList`. Recall from §3.2.1 that the primary covariate in the display formula for `PBG` is `dose` but we want the model to be fit as a function of `log(dose)`. You will either need to specify the model explicitly or to re-define the display formula for the data. Also note that the grouping formula should be `~Rabbit/Treatment`, but the grouping in the display formula is `~Rabbit`.

   (b) Plot the confidence intervals on the coefficients. Which parameters appear to be constant across all `Rabbit/Treatment` combinations? Does there appear to be a systematic shift in the `xmid` parameter according to `Treatment`?

   (c) Fit a two-level NLME model with a fixed effect for `Treatment` on the `xmid` parameter and with random effects for `Rabbit` on the `B` parameter and for `Treatment` within `Rabbit` on the `B` and `xmid` parameters. Begin with a diagonal Ψ_2 matrix. If that model fit converges, update to a general positive-definite Ψ_2 matrix and compare the two fitted models with `anova`. Which model is preferred?

   (d) Summarize your preferred model. Is the fixed effect for `Treatment` on `xmid` significant? Also check using the output from `intervals`.

   (e) Plot the augmented predictions from your preferred model. Remember that if the display formula for the data has `dose` as the primary covariate then you will need to use `scales = list(x = list(log = 2))` to get the symmetric shape of the logistic curve. Adjust the `layout` argument for the plot so the panels are aligned by `Rabbit`. Do these plots indicate deficiencies in the model?

(f) Examine residual plots for other possible deficiencies in the model. Also check plots of the random effects versus covariates to see if important fixed effects have been omitted.

(g) Is it necessary to use a four-parameter logistic curve? Experiment with fitting the three-parameter logistic model (`SSlogis`, Appendix C.7) to see if a comparable fit can be obtained. Check residual plots from your fitted models and comment.

3. The `Glucose2` data described in Appendix A.10 consist of blood glucose levels measured 14 times over a period of 5 hours on 7 volunteers who took alcohol at time 0. The same experiment was repeated on a second occasion with the same subjects but with a dietary additive used for all subjects. These data are analyzed in Hand and Crowder (1996, Example 8.4, pp. 118–120), where the following empirical model relating the expected `glucose` level to `Time` is proposed.

$$\texttt{glucose} = \phi_1 + \phi_2 \texttt{Time}^3 \exp\left(-\phi_3 \texttt{Time}\right)$$

Note that there are two levels of grouping in the data: `Subject` and `Date` within `Subject`.

(a) Plot the data at the `Subject` display level (use `plot(Glucose2, display = 1)`. Do there appear to be systematic differences between the two dates on which the experiments were conducted (which could be associated with the dietary supplement)?

(b) There is no self-starting function representing the model for the glucose level included in the nlme library. Use `nlsList` with starting values `start = c(phi1=5, phi2=-1, phi3=1)` (derived from Hand and Crowder (1996)) to fit separate models for each `Subject` and for each `Date` within `Subject`. Plot the individual confidence intervals for each of the two `nlsList` fits. Verify that `phi1` and `phi2` seem to vary significantly for both levels of grouping, but `phi3` does not. (There is an unusual estimate of `phi3` for `Subject` 6, `Date` 1 but all other confidence intervals overlap.)

(c) Fit a two-level NLME model with random effects for `phi1` and `phi2`, using as starting values for the fixed effects the estimates from either of the `nlsList` fits (`start = fixef(object)`, with `object` replaced with the name of the `nlsList` object). Examine the confidence intervals on the variance–covariance components; what can you say about the precision of the estimated correlation coefficients?

(d) Refit the NLME model using diagonal Ψ_q matrices for both grouping levels and compare the new fit to the previous one using `anova`. Investigate if there are random effects that can be dropped from the model using `intervals`. If so, refit the model with fewer random effects and compare it to the previous fit using `anova`. Plot the residuals versus `Time` and comment on the apparent adequacy of the empirical model.

(e) Plot the semivariogram of the standardized residuals corresponding to the final NLME fit obtained in the last item. Use `Time` as the covariate to define the distances in the semivariogram and consider only distances$\leq$ 22 (that is, use `plot(Variogram(object, form = ~Time, maxDist = 22))`). Notice the increase in the semivariogram for smaller distances, which suggests that there the within-group errors may be correlated.

(f) Hand and Crowder (1996) use a continuous AR1 model for the within-group errors, with `Time` as the covariate. Update the NLME fit adding a continuous AR1 correlation structure on `Time` (`corr = corCAR1(form = Time)`. Compare the fits using `anova`. Examine the plot of the semivariogram of the *normalized* residuals for the NLME with `corCAR1` correlation structure and comment on the adequacy of the within-group correlation model.

(g) The NLME model used by Hand and Crowder (1996) includes random effects for `phi1` and `phi2` only at the `Subject` level and uses a `corCAR1` correlation structure on `Time` for the `Date` within `Subject` errors, with errors from different `Dates` within the same `Subject` assumed to be independent. Fit such model using

```
random = B1 + B2 ~ 1 | Subject,
corr = corCAR1(form = ~Time | Subject/Date)
```

and compare it to the multilevel NLME model obtained in the previous item. Which one do you think gives a better fit?

(h) The main objective of the experiment was to determine if there were significant differences between the blood glucose level profiles over time associated with the use of the dietary additive (which is totally confounded with `Dose` in this case). Investigate the significance of the dietary additive by refitting the NLME model (with `corCAR1` correlation) incorporating a "`Date` effect" for each fixed effect (use `fixed = phi1 + phi2 + phi3 Date`). You will need to give new starting values for the fixed effects (use the previous fixed effects estimates for the `Intercept` terms and 0 for the `Date` terms). Use `summary` to assess the significance of the dietary additive effect; does it seem to make any difference on the blood glucose levels? Are your conclusions consistent with the plot of the data?

4. An NLME analysis of the `Theoph` data is presented in §8.2.1. The final `nlme` fit obtained in that section, `fm3Theo.nlme`, includes random effects for the `lKa` and `lCl` coefficients and a diagonal Ψ.

(a) Use the `gnls` function described in §8.3.3 to fit the `SSfol` model to the theophylline concentrations with no random effects. Compare this fit to `fm3Theo.nlme` using `anova`. Obtain the boxplots of the residuals by `Subject` (`plot(object, Subject~resid(.))`) and comment on the observed pattern.

(b) Print and plot the ACF of the standardized residuals for the `gnls` fit (use `form = ~1 | Subject` to specify the grouping structure). The decrease in the ACF with lag suggests that an AR1 model may be adequate.

(c) Update the `gnls` fit incorporating an AR1 correlation structure, using the lag-1 autocorrelation from the `ACF` output as an initial estimate for the correlation parameter (`corr = corAR1(0.725, form = ~1 | Subject)`). Compare this fit to the previous `gnls` fit using `anova`. Is there significant evidence of autocorrelation? Examine the plot of the ACF of the normalized residuals. Does the AR1 model seem adequate?

(d) Compare the `gnls` fit with AR1 correlation structure to the `fm3Theo.nlme` fit of §8.2.1 using `anova` with the argument `test` set to `FALSE` (why?). Which model seems better?