Nonlinear Regression, Nonlinear Least Squares, and Nonlinear Mixed Models in R

An Appendix to *An R Companion to Applied Regression*, third edition

John Fox & Sanford Weisberg

last revision: 2018-06-02

Abstract

The *nonlinear regression model* generalizes the linear regression model by allowing for mean functions like $\mathrm{E}(y|x) = \theta_1/\{1 + \exp[-(\theta_2 + \theta_3 x)]\}$, in which the parameters, the θs in this model, enter the mean function nonlinearly. If we assume additive errors, then the parameters in models like this one are often estimated via least squares. In this appendix to Fox and Weisberg (2019) we describe how the `nls()` function in R can be used to obtain estimates, and briefly discuss some of the major issues with nonlinear least squares estimation. We also describe how to use the `nlme()` function in the **nlme** package to fit nonlinear mixed-effects models. Functions in the **car** package than can be helpful with nonlinear regression are also illustrated.

The *nonlinear regression model* is a generalization of the linear regression model in which the conditional mean of the response variable is not a linear function of the parameters. As a simple example, the data frame `USPop` in the **carData** package, which we load along with the **car** package, has decennial U. S. Census population for the United States (in millions), from 1790 through 2000. The data are shown in Figure 1 (a):[1]

```
library("car")
```

```
Loading required package: carData
```

```
brief(USPop)
```

```
22 x 2 data.frame (17 rows omitted)
   year population
    [i]        [n]
1  1790     3.9292
2  1800     5.3085
3  1810     7.2399
. . .
21 1990   248.7099
22 2000   281.4219
```

```
plot(population ~ year, data=USPop, main="(a)")
abline(lm(population ~ year, data=USPop))
```

[1] The standard R functions `plot` and general-purpose **car** functions like `brief` are discussed in Fox and Weisberg (2019). All the R code used in this appendix can be downloaded from `http://tinyurl.com/carbook`. Alternatively, if you are running R and are attached to the internet, load the **car** package and enter the command `carWeb(script="appendix-nonlinear")` to view the R command script file for the appendix in your browser.

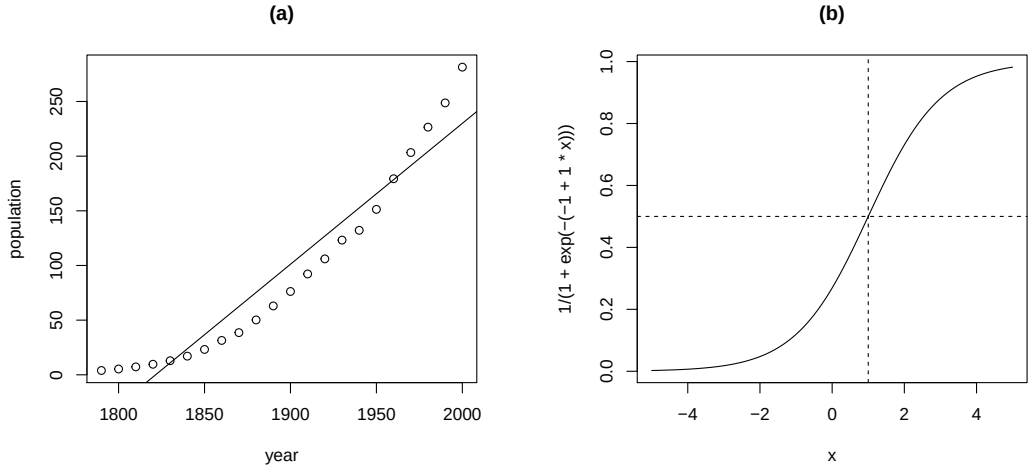

Figure 1: (a) U. S. population, from the U. S. Census, and (b) a generic logistic growth curve.

The simple linear regression least-squares line shown on the graph clearly does not match these data: The U. S. population is not growing by the same amount in each decade, as is required by the straight-line model. While in some problems transformation of predictors or the response can change a nonlinear relationship into a linear one (in these data, replacing `population` by its cube-root linearizes the plot, as can be discovered using the `powerTransform()` function in the **car** package), in many instances modeling the nonlinear pattern with a nonlinear mean function is desirable.

A common simple model for population growth is the *logistic growth model*,

$$\begin{aligned} y &= m(\mathbf{x}, \boldsymbol{\theta}) + \varepsilon \\ &= \frac{\theta_1}{1 + \exp[-(\theta_2 + \theta_3 x)]} + \varepsilon \end{aligned} \tag{1}$$

where y is the response, population size in our example, and we will take the predictor $x =$ `year`. We introduce the notation $m(\mathbf{x}, \boldsymbol{\theta})$ for the mean function, which depends on a possibly vector-valued parameter $\boldsymbol{\theta}$ and a possibly vector-valued predictor $\mathbf{x}$, here the single predictor x. More generally, we assume that the predictor vector $\mathbf{x}$ consists of fixed, known values.

As a function of x, the mean function in Equation 1 is a curve, as shown in Figure 1 (b) for the special case of $\theta_1 = 1, \theta_2 = 1, \theta_3 = 1$:

```
curve(1/(1+exp(-(-1 + 1*x))), from=-5, to=5, main="(b)")
abline(h=1/2, v=1, lty=2)
```

Changing the values of the parameters $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)'$ stretches or shrinks the axes, and changes the rate at which the curve varies from its lower value at 0 to its maximum value. If $\theta_3 > 0$, then as x gets larger the term $\exp[-(\theta_2 + \theta_3 x)]$ gets closer to 0, and so $m(x, \boldsymbol{\theta})$ will approach the value θ_1 as an *asymptote*. Assuming logistic population growth therefore imposes a limit to population size. Similarly, again if $\theta_3 > 0$, as $x \to -\infty$, the term $\exp[-(\theta_2 + \theta_3 x)]$ grows large without bound and so the mean will approach 0. Interpreting the meaning of θ_2 and θ_3 is more difficult. The logistic growth curve is symmetric about the value of x for which $m(x, \boldsymbol{\theta})$ is midway between 0 and θ_1. It is not hard to show that $m(x = -\theta_2/\theta_3, \boldsymbol{\theta}) = \theta_1/2$, and so the curve is symmetric about the midpoint $x = -\theta_2/\theta_3$. The parameter θ_3 controls how quickly the curve transitions from the lower asymptote of 0 to the upper asymptote at θ_1, and is therefore a growth-rate parameter.

It is not obvious that a curve of the shape in Figure 1 (b) can match the data shown in Figure 1 (a), but a part of the curve, from about $x = -3$ to $x = 2$, may be able to fit the data fairly well. Fitting the curve corresponds to estimating parameters to get a logistic growth function that matches the data. Determining whether extrapolation of the curve outside this range makes sense is beyond the scope of this brief report.

1 Fitting Nonlinear Regressions with the `nls()` Function

The standard `nls()` function in R is used for estimating parameters via nonlinear least squares. Following Weisberg (2014, Chap. 11), the general nonlinear regression model is[2]

$$y = \mathrm{E}(y|\mathbf{x}) + \varepsilon = m(\mathbf{x}, \boldsymbol{\theta}) + \varepsilon$$

This model posits that the mean $\mathrm{E}(y|\mathbf{x})$ depends on $\mathbf{x}$ through the *kernel mean function* $m(\mathbf{x}, \boldsymbol{\theta})$, where the predictor $\mathbf{x}$ has one or more components and the parameter vector $\boldsymbol{\theta}$ also has one or more components. In the logistic growth example in Equation 1, $\mathbf{x}$ consists of the single predictor $x = \texttt{year}$ and the parameter vector $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)'$ has three components. The model further assumes that the errors ε are independent with variance σ^2/w, where the w are known nonnegative weights, and σ^2 is a generally unknown variance to be estimated from the data.[3] In many applications $w = 1$ for all observations.

The `nls()` function can be used to estimate $\boldsymbol{\theta}$ as the values that minimize the residual sum of squares,

$$S(\boldsymbol{\theta}) = \sum w\left[y - m(\boldsymbol{\theta}, \mathbf{x})\right]^2 \tag{2}$$

We will write $\widehat{\boldsymbol{\theta}}$ for the minimizer of the residual sum of squares.

Unlike the linear least-squares problem, there is usually no closed-form formula that provides the minimizer of Equation 2. An iterative procedure is used, which in broad outline is as follows:

1. The user supplies an initial guess, say $\mathbf{t}_0$ of *starting values* for the parameters. Whether or not the algorithm can successfully find a minimizer will depend on getting starting values that are reasonably close to the solution. We discuss how this might be done for the logistic growth function below. For some special mean functions, including logistic growth, R has *self-starting* functions that can avoid this step.

2. At iteration $j \geq 1$, the current guess $\mathbf{t}_j$ is obtained by updating $\mathbf{t}_{j-1}$. If $S(\mathbf{t}_j)$ is smaller than $S(t_{\mathbf{t}-1})$ by at least a predetermined amount, then the counter j is increased by 1 and this step is repeated. If no such improvement is possible, then $\mathbf{t}_{j-1}$ is taken as the estimator $\widehat{\boldsymbol{\theta}}$ of $\boldsymbol{\theta}$.

This simple algorithm hides at least three important considerations. First, we want a method that will guarantee that at each step we either get a smaller value of S or at least S will not increase. There are many nonlinear least-squares algorithms; see, for example, Bates and Watts (1988). Many algorithms make use of the derivatives of the mean function with respect to the parameters. The default algorithm in `nls()` uses a form of Gauss-Newton iteration that employs derivatives approximated numerically unless we provide functions to compute the derivatives. Second, the sum-of-squares function S may be a perverse function with multiple minima. As a consequence, the purported least-squares estimates could be a local rather than global minimizer of S. Third, as given the algorithm can go on forever if improvements to S are small at each step. As a practical matter, therefore, there is an iteration limit that specifies the maximum number of iterations permitted, and a tolerance that defines the minimum improvement that will be considered to be greater than 0.

The call to `nls()` is similar to the call to `lm()` for linear models. Here are the arguments:

[2] Also see Fox (2016, Sec. 17.4)

[3] The assumption of independent errors is often untenable for time-series data such as the U. S. population data, where errors may be substantially autocorrelated. See the appendix on time-series regression.

```
args(nls)

function (formula, data = parent.frame(), start, control = nls.control(),
    algorithm = c("default", "plinear", "port"), trace = FALSE,
    subset, weights, na.action, model = FALSE, lower = -Inf,
    upper = Inf, ...)
NULL
```

We discuss each of these arguments in turn:

`formula` The `formula` argument is used to tell `nls()` about the mean function. The formula equivalent to Equation 1 (page 2) is

```
population ~ theta1/(1 + exp(-(theta2 + theta3*year)))
```

As in `lm()`, the left side of the formula specifies the response variable, and is followed by the tilde (`~`) as a separator that is commonly read as "is regressed on" or "is modeled by." The right side of the formula for nonlinear models is very different from `lm()` models. In the simplest form for `nls()`, the right side is a mathematical expression consisting of constants, like the number `1`; predictors, in this case just `year`; named parameters like `theta1`, `theta2` and `theta3`; and mathematical functions and operators like `exp()` for exponentiation, `/` for division and `+` for addition. Factors, interactions, and in general the Wilkinson-Rogers notation employed for linear models, are not used with `nls()`. Parentheses are used with the usual precedence rules for mathematics, but square brackets "`[ ]`" and curly braces "`{ }`" cannot be used. If values of the parameters and predictors were specified, then the right side of the formula would evaluate to a number. We can name the parameters with any legal R names, such as `theta1`, `alpha`, `t1`, or `Asymptote`.

The `formula` for `nls()` can take several other forms beyond the sample form described here. We will use a few other forms in later sections of this appendix, but even so we won't describe this argument in full generality.

`start` The argument `start` is a list that tells `nls()` which of the named quantities on the right side of the formula are parameters, and thus implicitly which are predictors. The argument also provides starting values for the parameter estimates. For the example, `start=list(theta1=440, theta2= -4, theta3=0.2)` names the `theta`s as the parameters and also specifies starting values for them. Since `year` has no starting value, it is taken by `nls()` to be a predictor. The `start` argument is required unless a self-starting function is used in the `formula` argument.

`algorithm = "default"` The `"default"` algorithm used in `nls()` is a Gauss-Newton algorithm. Other possible values are `"plinear"` for the Golub-Pereyra algorithm for partially linear models and `"port"` for a algorithm that should be selected if there are constraints on the parameters (see the next argument). The help page for `nls()` gives references.

`lower = -Inf`, `upper = Inf` Parameters of the model might be constrained to lie in a certain region. In the logistic population-growth model, for example, we must have $\theta_3 > 0$, as population size is increasing, and we must also have $\theta_1 > 0$. In some problems we would like to be certain that the algorithm never considers values for $\boldsymbol{\theta}$ outside the feasible range. The arguments `lower` and `upper` are vectors of lower and upper bounds, replicated to be as long as `start`. If unspecified, all parameters are assumed to be unconstrained. *Bounds can only be used with the* `"port"` *algorithm. They are ignored, with a warning, if given for other algorithms.*

`control = nls.control()` This argument is set to a call to the `nls.control()` function, which may be used to modify characteristics of the computing algorithm:

```
args(nls.control)
```

```
function (maxiter = 50, tol = 1e-05, minFactor = 1/1024, printEval = FALSE,
    warnOnly = FALSE)
NULL
```

Users will generally be concerned with the `control` argument only if convergence problems are encountered. For example, to change the maximum number of iterations to 40 and the convergence tolerance to 10^{-6}, set `control=nls.control(maxiter=40, tol=1e-6)`.

`trace = FALSE` If `TRUE`, print the value of the residual sum of squares and the parameter estimates at each iteration. The default is `FALSE`.

`data`, `subset`, `na.action`, `weights` These arguments are the same as for `lm()`, with `data`, `subset`, and `na.action` specifying the data to which the model is to be fit, and `weights` giving the weights w to be used for the least-squares fit. If the `weights` argument is missing then all weights are set equal to 1.

2 Starting Values

Most nonlinear least-squares algorithms require specification of *starting values* for the parameters, which are θ_1, θ_2 and θ_3 for the logistic growth model of Equation 1 (page 2).[4] There are many ways to determine starting values in nonlinear regression, but they generally require consideration of the mathematical form of the model. For the logistic growth model, for example, we can write

$$\begin{aligned} y &\approx \frac{\theta_1}{1+\exp[-(\theta_2+\theta_3 x)]} \\ y/\theta_1 &\approx \frac{1}{1+\exp[-(\theta_2+\theta_3 x)]} \\ \log\left[\frac{y/\theta_1}{1-y/\theta_1}\right] &\approx \theta_2+\theta_3 x \end{aligned}$$

The first of these three equations is the original logistic growth function. In the second equation, we divide through by the asymptote θ_1, so the left side is now a positive number between 0 and 1. We then apply the logit transformation, as in binomial regression with a logit link, to get a linear model. Consequently, if we have a starting value t_1 for θ_1 we can compute starting values for the other θs by the OLS linear regression of the logit of y/t_1 on x.

A starting value of the asymptote θ_1 should be some value larger than any value in the data, and so a value of around $t_1 = 400$ is a reasonable choice. (The Census count for 2010, which we deliberately omitted from the data, was 308,745,538.) Using `lm()` and the `logit()` function from the **car** package, we compute starting values for the other two parameters:

```
lm(logit(population/400) ~ year, USPop)
```

```

Call:
lm(formula = logit(population/400) ~ year, data = USPop)

Coefficients:
(Intercept)         year
   -49.2499       0.0251
```

[4] Generalized linear models use a similar iterative estimation method, but finding starting values is usually not important because a set of default starting values is almost always adequate.

Thus, starting values for the other θs are $t_2 = -49$ and $t_3 = 0.025$.

```
pop.mod <- nls(population ~ theta1/(1 + exp(-(theta2 + theta3*year))),
     start=list(theta1 = 400, theta2 = -49, theta3 = 0.025),
     data=USPop, trace=TRUE)
```

```
3060.8 :  400.000 -49.000   0.025
558.54 :  426.061991 -42.307856   0.021421
457.97 :  438.414699 -42.836902   0.021677
457.81 :  440.890336 -42.698662   0.021602
457.81 :  440.816807 -42.708050   0.021606
457.81 :  440.834471 -42.706883   0.021606
457.81 :  440.833344 -42.706977   0.021606
```

By setting `trace=TRUE`, we can see that S evaluated at the starting values is 3061. The first iteration reduces this to 558.5, the next iteration to 458, and the remaining iterations result in only very small changes. We get convergence in 6 iterations.

We use the `summary()` function to print a report for the fitted model:

```
summary(pop.mod)
```

```
Formula: population ~ theta1/(1 + exp(-(theta2 + theta3 * year)))

Parameters:
        Estimate Std. Error t value Pr(>|t|)
theta1 440.83334   35.00014    12.6  1.1e-10
theta2 -42.70698    1.83914   -23.2  2.1e-15
theta3   0.02161    0.00101    21.4  8.9e-15

Residual standard error: 4.91 on 19 degrees of freedom

Number of iterations to convergence: 6
Achieved convergence tolerance: 1.24e-06
```

The column marked `Estimate` displays the least squares estimates of the parameters. The estimated upper bound for the U. S. population is thus 440.8, or about 441 million. The column marked `Std. Error` displays the standard errors of thes estimated regression coefficients. The very large standard error for the $\widehat{\theta}_1$ reflects the uncertainty in the estimated asymptote when all the observed data are much smaller than the asymptote. The estimated year in which the population is half the asymptote is $-\widehat{\theta}_3/\widehat{\theta}_2 = 1976.6$. The standard error of this estimate can be computed by the `deltaMethod()` function in the **car** package:

```
deltaMethod(pop.mod, "-theta2/theta3")
```

```
               Estimate     SE  2.5 % 97.5 %
-theta2/theta3   1976.6 7.5558 1961.8 1991.4
```

and so the standard error is about 7.6 years.

The column `t value` in the `summary()` output shows the ratio of each parameter estimate to its standard error. In sufficiently large samples, this ratio will generally have a normal distribution, but interpreting "sufficiently large" is difficult with nonlinear models. *Even if the errors ε are normally distributed, the estimates may be far from normally distributed in small samples.* The p-values shown

are based on asymptotic normality, and test the null hypotheses that each θ is equal to 0. In this example, and frequently in nonlinear models, these hypotheses are not really of interest; for example, it isn't sensible to suppose that the asymptotic population of the U. S. is 0. The residual standard deviation is the estimate of σ,

$$\widehat{\sigma} = \sqrt{S(\widehat{\boldsymbol{\theta}})/(n-k)}$$

where k is the number of parameters in the mean function, 3 in this example.

The `Confint()` function in the **car** package can be used to get confidence intervals for each of the parameter estimates. These intervals are based on the profile log-likelihood, and are generally much more accurate than are intervals based on asymptotic normality:

```
Confint(pop.mod)
```

```
Waiting for profiling to be done...
```

```
         Estimate       2.5%      97.5%
theta1 440.833344 381.493655 536.938593
theta2 -42.706977 -46.586310 -39.110835
theta3   0.021606   0.019613   0.023719
```

Many familiar generic functions, such as `residuals()`, have methods for the nonlinear-model objects produced by `nls()`. For example, the `predict()` function makes it simple to plot the fit of the model as in Figure 2:

```
plot(population ~ year, USPop, xlim=c(1790, 2100), ylim=c(0, 450))
with(USPop, lines(seq(1790, 2100, by=10),
      predict(pop.mod, data.frame(year=seq(1790, 2100, by=10))), lwd=2))
points(2010, 308.745538, pch=16, cex=1.3)
abline(h=0, lty=2)
abline(h=coef(pop.mod)[1], lty=2)
abline(h=0.5*coef(pop.mod)[1], lty=2)
abline(v= -coef(pop.mod)[2]/coef(pop.mod)[3], lty=2)
```

We made this plot more complicated than necessary simply to display the fit of the logistic model to the data, in order to show the shape of the fitted curve projected into the future, to include the Census population in 2010, and to add lines for the asymptotes and for the point half way between the asymptotes. While Figure 2 confirms that the logistic growth mean function generally matches the data, the residual plot in Figure 3 suggests that there are systematic features that are missed, reflecting differences in growth rates, perhaps due to factors such as changes in immigration policies:

```
with(USPop, plot(year, residuals(pop.mod), type='b'))
abline(h=0, lty=2)
```

3 Self-Starting Models

Bates and Watts (1988, Sec. 3.2) describe many techniques for finding starting values for fitting nonlinear models. For the logistic growth model described in this paper, for example, finding starting values amounts to (1) guessing the parameter θ_1 as a value larger than any observed in the data; and (2) substituting this value into the mean function, rearranging terms, and then getting other starting values by OLS simple linear regression. This algorithm can of course be written as an R function to get starting values automatically, and this is the basis of the *self-starting nonlinear models* described by Pinheiro and Bates (2000, Sec. 8.1.2).

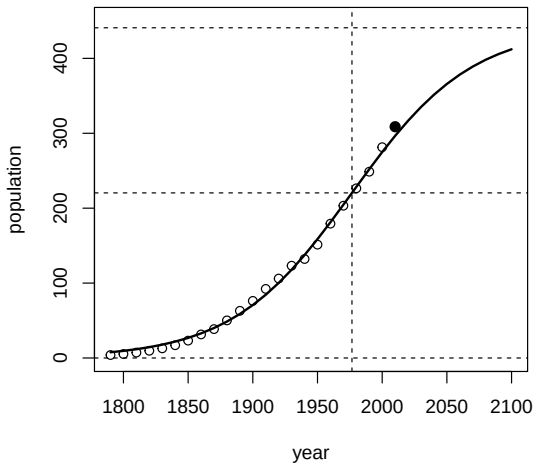

Figure 2: U. S. population in millions, with logistic growth fit extrapolated to 2100. The open circles represent Census population counts through 2000, while the filled circle represents the 2010 census population. The broken horizontal lines are drawn at the asymptotes and midway between the asymptotes; the broken vertical line is drawn at the year corresponding to the mid-way point.

Figure 3: Residuals from the logistic growth model fit to the U. S. population data.

The self-starting logistic growth model in R is based on a different, but equivalent, parametrization of the logistic function. We will start again with the logistic growth model, with mean function

$$m(x, \boldsymbol{\theta}) = \frac{\theta_1}{1 + \exp[-(\theta_2 + \theta_3 x)]}$$

We have seen that the ratio $-\theta_2/\theta_3$ is an interesting function of the θs, and so we might reparametrize this mean function as:

$$m(x, \boldsymbol{\phi}) = \frac{\phi_1}{1 + \exp[-(x - \phi_2)/\phi_3]}$$

As the reader can verify, we have defined $\phi_1 = \theta_1$ $\phi_2 = -\theta_2/\theta_3$, and $\phi_3 = 1/\theta_3$. In the ϕ-parametrization, ϕ_1 is the upper asymptote, ϕ_2 is the value of x at which the response is half its asymptotic value, and ϕ_3 is a rate parameter. Because the ϕ-parameters are a one-to-one nonlinear transformation of the θ-parameters, the two models provide the same fit to the data.

Fitting a nonlinear model with the self-starting logistic growth function in R is quite easy:

```
pop.ss <- nls(population ~ SSlogis(year, phi1, phi2, phi3), data=USPop)
summary(pop.ss)
```

```
Formula: population ~ SSlogis(year, phi1, phi2, phi3)

Parameters:
     Estimate Std. Error t value Pr(>|t|)
phi1   440.83      35.00    12.6  1.1e-10
phi2  1976.63       7.56   261.6  < 2e-16
phi3    46.28       2.16    21.4  8.9e-15

Residual standard error: 4.91 on 19 degrees of freedom

Number of iterations to convergence: 0
Achieved convergence tolerance: 6.82e-07
```

The right side of the formula is now the name of a function that has the responsibility for computing the mean function and for finding starting values. The estimate of the asymptote parameter ϕ_1 and its standard error are identical to the estimate and standard error for $\widehat{\theta_1}$ in the θ-parametrization. Similarly, the estimate for ϕ_2 and its standard error are identical to the estimate and standard error for $-\theta_2/\theta_3$ obtained from the delta method (page 6). Finally, applying the delta method again,

```
deltaMethod(pop.mod, "1/theta3")
```

```
         Estimate     SE  2.5 % 97.5 %
1/theta3   46.284 2.1574 42.055 50.512
```

we see that $\widehat{\phi_3} = 1/\widehat{\theta_3}$ with the same standard error from the delta method as from the ϕ-parametrized model.

Table 1 lists the self-starting nonlinear functions that are available in the standard R system. Pinheiro and Bates (2000, Sec. 8.1.2) discuss how users can make their own self-starting functions.

4 Parametrization

4.1 Linear Reparametrization

In some problems it may be useful to transform a predictor linearly to make the resulting parameters more meaningful. In the U. S. population data, we might consider replacing the predictor `year` by `decade` = (`year` − 1790)/10.

Function	Equation, $m(x, \phi) =$
SSasymp	Asymptotic regression
	$\phi_1 + (\phi_2 - \phi_1)\exp[-\exp(\phi_3)x]$
`SSasympOff()`	Asymptotic regression with an offset
	$\phi_1\{1 - \exp[-\exp(\phi_2) \times (x - \phi_3)]\}$
`SSasympOrig()`	Asymptotic regression through the origin
	$\phi_1\{1 - \exp[-\exp(\phi_2)x]\}$
`SSbiexp()`	Biexponential model
	$\phi_1 \exp[-\exp(\phi_2)x] + \phi_3 \exp[-\exp(\phi_4)x]$
`SSfol()`	First-order compartment model
	$\frac{D\exp(\phi_1+\phi_2)}{\exp(\phi_3)[\exp(\phi_2)-\exp(\phi_1)]}\{\exp[-\exp(\phi_1)x] - \exp([-\exp(\phi_2)x]\}$
`SSfpl()`	Four-parameter logistic growth model
	$\phi_1 + \frac{\phi_2-\phi_2}{1+\exp[(\phi_3-x)/\phi_4]}$
`SSgompertz()`	Gompertz model
	$\phi_1 \exp(\phi_2 x^{\phi_3})$
`SSlogis()`	Logistic model
	$\phi_1/(1 + \exp[(\phi_2 - x)/\phi_3])$
`SSmicmen()`	Michaelis-Menten model
	$\phi_1 x/(\phi_2 + x)$
`SSweibull()`	Weibull model
	$\phi_1 + (\phi_2 - \phi_1)\exp[-\exp(\phi_3)x^{\phi_4}]$

Table 1: The self-starting functions available in R, from Pinheiro and Bates (2000, Appendix C). Tools are also available for users to write their own self-starting functions.

```
USPop$decade <- (USPop$year - 1790)/10
pop.ss.rescaled <- nls(population ~ SSlogis(decade, nu1, nu2, nu3), data=USPop)
compareCoefs(pop.ss, pop.ss.rescaled)

Calls:
1: nls(formula = population ~ SSlogis(year, phi1, phi2, phi3),
   data = USPop, algorithm = "default", control = list(maxiter =
   50, tol = 1e-05, minFactor = 0.0009765625, printEval = FALSE,
   warnOnly = FALSE), trace = FALSE)
2: nls(formula = population ~ SSlogis(decade, nu1, nu2, nu3),
  data = USPop, algorithm = "default", control = list(maxiter =
  50, tol = 1e-05, minFactor = 0.0009765625, printEval = FALSE,
  warnOnly = FALSE), trace = FALSE)

     Model 1 Model 2
phi1     441
SE        35

phi2 1976.63
SE      7.56

phi3   46.28
SE      2.16

nu1              441
SE                35
```

```
nu2           18.663
SE             0.756

nu3            4.628
SE             0.216
```

```
sigmaHat(pop.ss.rescaled)
```

```
[1] 4.9087
```

The asymptote parameter estimate is the same in this model. The time at half-asymptote is now measured in decades, so $18.66 \times 10 + 1790 = 1977$ as before. The rate per decade is 1/10 times the rate per year. Other summaries, like the estimated value of σ, are identical in the two fits.

Scaling like this can often be helpful to avoid computational problems, or to get parameters that correspond to units of interest. Apart from computational issues, linear transformation of predictors has no effect on the fitted model.

4.2 Nonlinear Reparametrization

Nonlinear transformations of predictors can also be used to give different representations of the same fitted model. Sometimes the choice of parametrization can make a difference in computations, with one parametrization working while another one fails.

Nonlinear transformation brings up an interesting issue. For example, in the logistic growth model used in this appendix, suppose that in the θ parametrization the estimates are approximately normally distributed. Then, because the ϕs are nonlinear transformations of the θs, the estimates of the ϕs are necessarily *not* normally distributed. This observation highlights the problem with using asymptotic normality for inference in nonlinear models. Whether or not approximate normality is appropriate in the parametrization that was actually employed depends on the parametrization, on the sample size, and on the values of the data. Determining if normal inference is appropriate is a difficult issue; see Bates and Watts (1988, Chap. 6) for a graphical approach to this problem, and the documentation for the `nls.profile()` function in R.

The `Boot` function in the **car** package computes a case-resampling bootstrap that can be used for inference about nonlinear least squares estimates.[5] For example, the following command computes $R = 999$ bootstrap samples and retains the values of the coefficient estimates for each bootstrap sample:

```
set.seed(12345) # for repeatability
out4 <- Boot(pop.ss)
```

```
Loading required namespace: boot
```

```
summary(out4)
```

```
Number of bootstrap replications R = 999
     original bootBias bootSE bootMed
phi1    440.8   -18.33  48.86   431.2
phi2   1976.6    -4.35  10.87  1974.4
phi3     46.3    -1.06   2.55    45.6
```

[5] For information on bootstrapping regression models in R, see Section 4.3.7 of the text and the appendix on bootstrapping.

```
hist(out4)

Confint(out4)

Bootstrap bca confidence intervals

     Estimate    2.5 %   97.5 %
phi1  440.833  353.380  545.613
phi2 1976.634 1957.428 1997.657
phi3   46.284   42.208   52.215
```

The differences between the original estimates and the bootstrap means, recorded in the column named `bootBias`, suggest substantial bias in the estimates. The nominal standard errors reported in summary output for the model on page 9 are also considerably smaller than the standard errors based on the bootstrap. Histograms and density estimates of the bootstrap samples, produced by the `hist()` function and shown in Figure 4, reveal that the marginal distribution of each parameter estimate is moderately to severely skewed, suggesting that inference based on asymptotic normality is a poor idea. By default, the `Confint()` function reports BC_a biased-corrected bootstrap confidence intervals, which also appear in the graphs produced by `hist()`: See `?Confint.boot` and `?hist.boot`.

5 Nonlinear Mean Functions

Some problems will dictate the form of the nonlinear mean function that is of interest, possibly through theoretical considerations, differential equations, and the like. In other cases, the choice of a mean function is largely arbitrary, as only qualitative characteristics of the curve, such as the existence of an asymptote, are known in advance. Ratkowsky (1990) provides a very helpful catalog of nonlinear mean functions that are commonly used in practice, a catalog that both points out the variety of available functions and some of the ambiguity in selecting a particular function to study.

For example, consider the following three-parameter *asymptotic regression* models:

$$\begin{aligned}
m_1(x, \boldsymbol{\theta}) &= \alpha - \beta\gamma^x \\
m_2(x, \boldsymbol{\theta}) &= \alpha - \beta \exp(-\kappa x) \\
m_3(x, \boldsymbol{\theta}) &= \alpha\{1 - \exp[-\kappa(x - \zeta)]\} \\
m_4(x, \boldsymbol{\theta}) &= \alpha + (\mu - \alpha)\gamma^x \\
m_5(x, \boldsymbol{\theta}) &= \alpha + (\mu - \alpha) \exp(-\kappa x) \\
m_6(x, \boldsymbol{\theta}) &= \alpha - \exp[-(\delta + \kappa x)] \\
m_7(x, \boldsymbol{\theta}) &= \theta_1 + \beta[1 - \exp(-\kappa x)]
\end{aligned}$$

All of these equations describe exactly the same function! In each equation, α corresponds to the asymptote as $x \to \infty$, $\beta = \alpha - \mu$ is the range of y from its minimum to its maximum, μ is the mean value of y when $x = 0$, $\delta = \log(\beta)$ is the log of the range, ζ is the value of x when the mean response is 0, and κ and γ are rate parameters. Many of these functions have names in particular areas of study; for example, $m_3(x, \boldsymbol{\theta})$ is called the *von Bertalanffy function* and is commonly used in the fisheries literature to model fish growth. Ratkowsky (1990, Sec. 4.3) points out that none of these parametrizations dominates the others with regard to computational and asymptotic properties. Indeed, other parametrizations of this same function may be used in some circumstances. For example, Weisberg et al. (2010) reparametrize $m_3(x, \boldsymbol{\theta})$ as

$$m_{30}(x, \boldsymbol{\theta}) = \alpha\{1 - \exp[-(\log(2)/\kappa_0)(x - \zeta)]\}$$

If x is the age of an organism, then in this parametrization the transformed rate parameter κ_0 is the age at which the size y of the organism is expected to be one-half its asymptotic size.

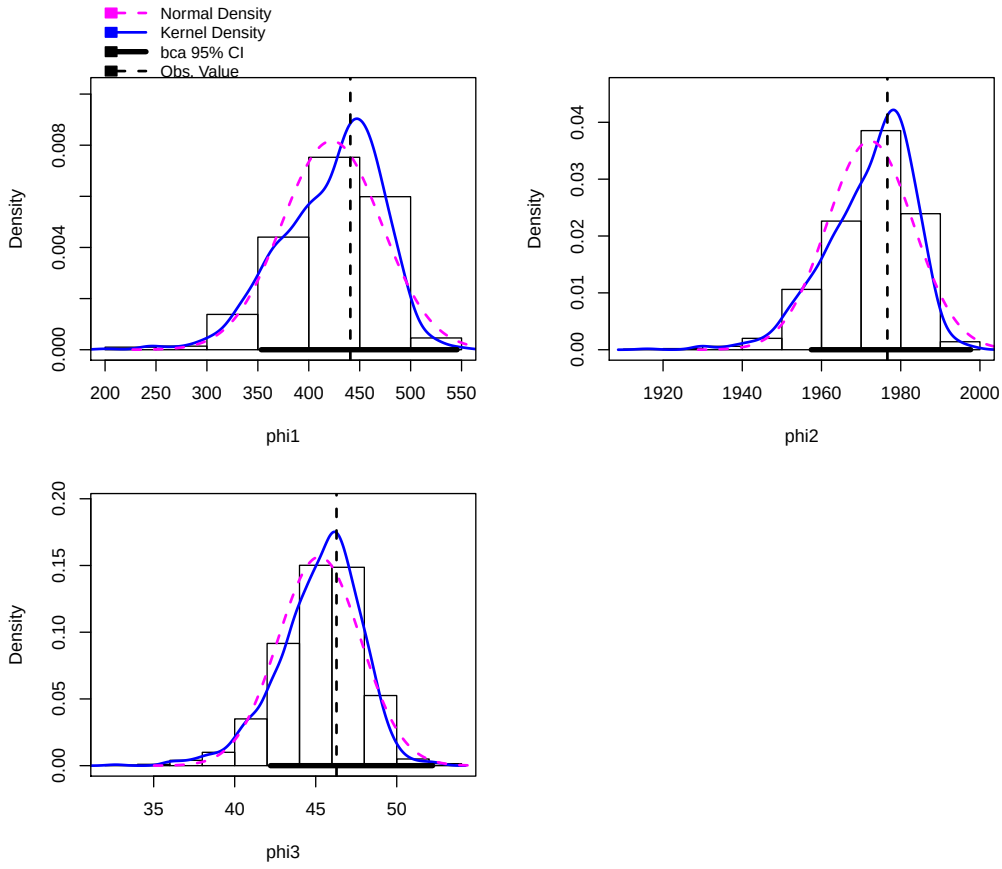

Figure 4: Distributions of the bootstrapped regression coefficients for the logistic growth model fit to the U. S. population data.

6 Nonlinear Models Fit with a Factor

A common problem with nonlinear models is fitting the same mean function to each of several groups of data. For example, the data frame `CanPop` in the **car** package has Canadian population data in the same format at the U. S. data. We combine the two data frames into one, and then draw a graph of the data for both countries:

```
brief(CanPop)
```

```
16 x 2 data.frame (11 rows omitted)
   year population
    [n]        [n]
1  1851      2.436
2  1861      3.230
3  1871      3.689
. . .
15 1991     26.429
16 2001     30.007
```

```
popData <- data.frame(rbind(data.frame(country="US", USPop[,1:2]),
                            data.frame(country="Canada", CanPop)))
brief(popData)
```

```
38 x 3 data.frame (33 rows omitted)
    country year population
        [f]  [n]        [n]
1    US     1790     3.9292
2    US     1800     5.3085
3    US     1810     7.2399
. . .
151  Canada 1991    26.4290
161  Canada 2001    30.0070
```

The combined data frame has a new variable called `country` with values `US` or `Canada`. The `brief()` function from the **car** package displays a few of the rows of `popData`.

```
scatterplot(population ~ year|country, data=popData, box=FALSE,
   reg=FALSE)
```

The `scatterplot()` function in the **car** package allows for automatic differentiation of the points by groups. Setting both `box` and `reg` to `FALSE` suppresses the irrelevant boxplots and simple-regression lines, respectively. The lines shown on the plot are nonparametric smooths, which we could also have suppressed by adding `smooth=FALSE` to the function call.

We can fit a logistic growth mean function separately to each country. The `nlsList()` function in the **nlme** package (Pinheiro and Bates, 2000) makes this easy. By default `nlsList()` assumes the same variance for the errors in all groups, but this is not appropriate in this example because the population counts for the larger U. S. are more variable than the counts for Canada. We use the argument `pool=FALSE` to force separate estimates of error variance for the two countries:

```
library("nlme")
m.list <- nlsList(population ~ SSlogis(year, phi1, phi2, phi3)|country,
    pool=FALSE, data=popData)
summary(m.list)
```

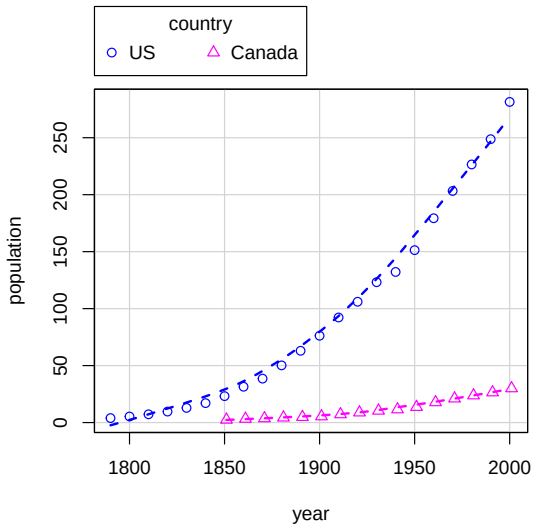

Figure 5: U.S. and Canadian population at decennial censuses; the curves were computed by nonparametric regression.

```
Call:
  Model: population ~ SSlogis(year, phi1, phi2, phi3) | country
   Data: popData

Coefficients:
   phi1
        Estimate Std. Error t value   Pr(>|t|)
US       440.833      35.00 12.5952 1.1390e-10
Canada    71.446      14.15  5.0492 2.2277e-04
   phi2
        Estimate Std. Error t value   Pr(>|t|)
US       1976.6     7.5558  261.61 2.9420e-35
Canada   2015.7    16.4747  122.35 2.7301e-21
   phi3
        Estimate Std. Error t value   Pr(>|t|)
US       46.284     2.1574  21.453 8.8670e-15
Canada   47.748     3.0601  15.604 8.4773e-10
```

```
(sds <- sapply(m.list, sigmaHat))
```

```
     US  Canada
4.90867 0.56713
```

The asymptotes are quite different for the two countries, and the year for achieving half the asymptote is about 40 years later in Canada than in the U.S., but the rate parameters are similar in the two countries. The residual SD estimates are very different as well.

With a little effort, we can use the `deltaMethod()` function to compute the standard error of the difference between corresponding parameters for the two countries. The object `m.list` created above is a list of `nls()` objects. We can use `lapply()` to get the estimates and their estimated variance-covariance matrices:

```
(betas <- lapply(m.list, coef))
```

```
$US
    phi1     phi2     phi3
 440.833 1976.634   46.284

$Canada
    phi1     phi2     phi3
  71.446 2015.663   47.748
```

```
(vars  <- lapply(m.list, vcov))
```

```
$US
         phi1    phi2    phi3
phi1 1225.012 262.854 69.1283
phi2  262.854  57.090 15.2287
phi3   69.128  15.229  4.6546

$Canada
        phi1    phi2   phi3
phi1 200.225 232.479 40.835
phi2 232.479 271.416 48.461
phi3  40.835  48.461  9.364
```

We then use `unlist` to combine the estimates into a single vector, and also combine the two covariance matrices into a single block-diagonal matrix; the block-diagonal structure makes sense because we have independent estimates of the parameters for the two countries:

```
(betas <- unlist(betas))
```

```
    US.phi1     US.phi2     US.phi3 Canada.phi1 Canada.phi2
    440.833    1976.634      46.284      71.446    2015.663
Canada.phi3
     47.748
```

```
zero <- matrix(0, nrow=3, ncol=3)
(var <- rbind( cbind(vars[[1]], zero), cbind(zero, vars[[2]])))
```

```
         phi1    phi2    phi3
phi1 1225.012 262.854 69.1283   0.000   0.000  0.000
phi2  262.854  57.090 15.2287   0.000   0.000  0.000
phi3   69.128  15.229  4.6546   0.000   0.000  0.000
phi1    0.000   0.000  0.0000 200.225 232.479 40.835
phi2    0.000   0.000  0.0000 232.479 271.416 48.461
phi3    0.000   0.000  0.0000  40.835  48.461  9.364
```

```
deltaMethod(betas, "US.phi3 - Canada.phi3", vcov=var)
```

```
                      Estimate     SE   2.5 % 97.5 %
US.phi3 - Canada.phi3  -1.4645 3.7441 -8.8028 5.8739
```

```
deltaMethod(betas, "US.phi2 - Canada.phi2", vcov=var)
```

```
                      Estimate     SE   2.5 %  97.5 %
US.phi2 - Canada.phi2  -39.029 18.125 -74.553 -3.5051
```

The rate parameters differ by about half a standard error of the difference, while the times at half-asymptotic population differ by about two standard errors, with the U. S. earlier.

If we number the countries as $k = 1, 2$ for the U. S. and Canada respectively, then the model fit by `nlsList()` is

$$y_k(x) = \frac{\phi_{1k}}{1 + \exp[-(x - \phi_{2k})/\phi_{3k}]} + \varepsilon_k$$

Interesting hypotheses could consist of testing $\phi_{21} = \phi_{22}$, $\phi_{31} = \phi_{32}$, or both of these. We now proceed to test these relevant hypotheses in R using likelihood-ratio tests. Because the error variances are different for the two countries, we can do this only approximately, by setting weights as follows:

```
w <- ifelse(popData$country == "Canada", (sds[1]/sds[2])^2, 1)
head(w)  # first few for U.S.
```

```
[1] 1 1 1 1 1 1
```

```
tail(w)  # last few for Canada
```

```
[1] 74.914 74.914 74.914 74.914 74.914 74.914
```

This procedure adjusts the residual sum of squares for the combined data to give the same standard errors we obtained using `nlsList()`. To get exact tests we would need to weight according to the unknown population variance ratio, rather than the sample variance ratio.

We let `can` be a dummy variable that has the value 1 for Canada and 0 for the U. S.,

```
popData$can <- ifelse(popData$country == "Canada", 1, 0)
```

The largest model we contemplate has separate parameters for each country. We fit this model first defining the formula, then getting starting values from `m.list`, and finally computing the fit:

```
form1 <- population ~ (1 - can)*(phi11/(1 + exp(-(year - phi21)/phi31))) +
                      can*(phi12/(1 + exp(-(year - phi22)/phi32)))
b <- coef(m.list)
m1 <- nls(form1, data=popData, weights=w,
          start=list(phi11=b[1, 1], phi12=b[1, 2], # U.S.
       phi21=b[1, 2], phi22=b[2, 2], phi31=b[1, 3], phi32=b[2, 3])) # Canada
```

A model with the same rate parameter for the two countries, $\phi_{31} = \phi_{32} = \phi_3$, is

```
form2 <- population ~ (1 - can)*(phi11/(1 + exp(-(year - phi21)/phi3))) +
                      can*(phi12/(1 + exp(-(year - phi22)/phi3)))
m2 <- nls(form2, data=popData, weights=w, start=list(phi11=b[1, 1], phi12=b[1, 2],
       phi21=b[1, 2], phi22=b[2, 2], phi3=b[1, 3]))
```

The `anova` function can be used to get the (approximate, because of weighting) likelihood-ratio test of equal rate parameters:

```
anova(m2, m1)
```

```
Analysis of Variance Table

Model 1: population ~ (1 - can) * (phi11/(1 + exp(-(year - phi21)/phi3)))
    + can * (phi12/(1 + exp(-(year - phi22)/phi3)))
Model 2: population ~ (1 - can) * (phi11/(1 + exp(-(year - phi21)/phi31)))
    + can * (phi12/(1 + exp(-(year - phi22)/phi32)))
  Res.Df Res.Sum Sq Df Sum Sq F value Pr(>F)
1     33        775
2     32        771  1    3.8    0.16   0.69
```

The p-value of nearly 0.7 suggests no evidence against a common rate. Confidence intervals (also approximate because of the approximate weights) for all parameters in the common-rate model are given by

```
Confint(m2)

Waiting for profiling to be done...

      Estimate     2.5%    97.5%
phi11  448.423  396.009  526.073
phi12   67.491   55.329   89.348
phi21 1978.290 1966.601 1993.380
phi22 2010.821 1994.653 2033.297
phi3    46.772   43.478   50.448
```

There is considerable imprecision in the estimates of the asymptotes, $\widehat{\phi}_{11}$ and $\widehat{\phi}_{12}$.

In problems with many groups that can be viewed as sampled from a population of groups, an appropriate model may be a *nonlinear mixed model*, as we explain in Section 8.

7 Analytic Derivatives

If the errors are assumed to be normally distributed, then the likelihood for the nonlinear regression model is

$$L(\boldsymbol{\theta}, \sigma^2) = -\frac{1}{\sqrt{2\pi}} \left(\prod \left[\frac{w}{\sigma}\right]\right) \exp\left[-\frac{1}{2\sigma^2} S(\boldsymbol{\theta})\right]$$

where the product is over the n observations, and $S(\boldsymbol{\theta})$ is the residual sum of squares (Equation 2 on page 3) evaluated at $\boldsymbol{\theta}$. Because $S(\boldsymbol{\theta}) \geq 0$, the likelihood is maximized by making $S(\boldsymbol{\theta})$ as small as possible, leading to the least squares estimates.

Suppose we let $r(\boldsymbol{\theta}) = \sqrt{w}(y - m(\mathbf{x}, \boldsymbol{\theta}))$ be the (Pearson) residual at $\mathbf{x}$ for a given value of $\boldsymbol{\theta}$. Differentiating $S(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ gives

$$\frac{\partial S(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -2 \sum \left[r(\boldsymbol{\theta}) \frac{\partial m\left(\boldsymbol{\theta}, \mathbf{x}\right)}{\partial \boldsymbol{\theta}} \right]$$

Setting the partial derivatives to $\mathbf{0}$ produces estimating equations for the regression coefficients. Because these equations are in general nonlinear, they require solution by numerical optimization. As in a linear model, it is usual to estimate the error variance by dividing the residual sum of squares for the model by the number of observations less the number of parameters (in preference to the ML estimator, which divides by n).

Coefficient variances may be estimated from a linearized version of the model. Let

$$F_{ij} = \frac{\partial m\left(\boldsymbol{\theta}, \mathbf{x}_i\right)}{\partial \boldsymbol{\theta}_j}$$

where i indexes the observation number and j the element of $\boldsymbol{\theta}$. Then the estimated asymptotic covariance matrix of the regression coefficients is

$$\widehat{\mathcal{V}}(\widehat{\boldsymbol{\theta}}) = \widehat{\sigma}^2 (\mathbf{F}'\mathbf{F})^{-1}$$

where $\widehat{\sigma}^2$ is the estimated error variance, and $\mathbf{F} = \{F_{ij}\}$ is evaluated at the least squares estimates. The square roots of the diagonal elements of this matrix are the standard errors returned by the `summary()` function.

The process of maximizing the likelihood involves calculating the *gradient* matrix $\mathbf{F}$. By default, `nls()` computes the gradient numerically using a finite-difference approximation, but it is also possible to provide a formula for the gradient directly to `nls()`. This is done by writing a function of the parameters and predictors that returns the fitted values of y, with the gradient as an attribute.

For the logistic growth model in the θ-parametrization we used at the beginning of this appendix, the partial derivatives of $m(x, \boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ are

$$\begin{aligned}
\frac{\partial m(x,\boldsymbol{\theta})}{\partial \theta_1} &= [1 + \exp(-(\theta_2 + \theta_3 x))]^{-1} \\
\frac{\partial m(x,\boldsymbol{\theta})}{\partial \theta_2} &= \theta_1 [1 + \exp(-(\theta_2 + \theta_3 x))]^{-2} \exp(-(\theta_2 + \theta_3 x)) \\
\frac{\partial m(x,\boldsymbol{\theta})}{\partial \theta_3} &= \theta_1 [1 + \exp(\theta_2 + \theta_3 x)]^{-2} \exp[-(\theta_2 + \theta_3 x)x]
\end{aligned}$$

In practice, the derivatives are evaluated at the current estimates of the θs.

To use an analytical gradient, the quantity on the right side of the `formula` must have an *attribute* called `gradient`. Here is how this is done for the logistic growth model:

```
model <- function(theta1, theta2, theta3, year){
     yhat <- theta1/(1 + exp(-(theta2 + theta3*year)))
     term <- exp(-(theta2 + theta3*year))
     gradient <- cbind((1 + term)^-1, # in proper order
         theta1*(1 + term)^-2 * term,
         theta1*(1 + term)^-2 * term * year)
     attr(yhat, "gradient") <- gradient
     yhat
     }
```

The `model()` function takes as its arguments all the parameters (`theta1`, `theta2`, `theta3`) and the predictors (in this case, just `year`). In the body of the function, `yhat` is set equal to the logistic growth function, and `gradient` is the gradient computed according to the formulas just given. We use the `attr()` function to assign `yhat` the gradient as an attribute. Our function then returns `yhat`. The `model()` function is used with `nls()` as follows:

```
(nls(population ~ model(theta1, theta2, theta3, year),
     data=USPop, start=list(theta1=400, theta2=-49, theta3=0.025)))

Nonlinear regression model
  model: population ~ model(theta1, theta2, theta3, year)
   data: USPop
  theta1   theta2   theta3
440.8334 -42.7070   0.0216
 residual sum-of-squares: 458

Number of iterations to convergence: 6
Achieved convergence tolerance: 1.31e-06
```

In many—perhaps most—cases, little is gained by this procedure, because the increase in computational efficiency is more than offset by the additional mathematical and programming effort required. It might be possible, however, to have one's cake and eat it too, by using the `deriv()` function in R to compute a formula for the gradient and to build the requisite function for the right side of the model. For the example:

```
(model2 <- deriv(~ theta1/(1 + exp(-(theta2 + theta3*year))), # rhs of model
     c("theta1", "theta2", "theta3"), # parameter names
     function(theta1, theta2, theta3, year){} # arguments for result
     ))

function (theta1, theta2, theta3, year)
{
    .expr4 <- exp(-(theta2 + theta3 * year))
    .expr5 <- 1 + .expr4
    .expr9 <- .expr5^2
    .value <- theta1/.expr5
    .grad <- array(0, c(length(.value), 3L), list(NULL, c("theta1",
        "theta2", "theta3")))
    .grad[, "theta1"] <- 1/.expr5
    .grad[, "theta2"] <- theta1 * .expr4/.expr9
    .grad[, "theta3"] <- theta1 * (.expr4 * year)/.expr9
    attr(.value, "gradient") <- .grad
    .value
}

(nls(population ~ model2(theta1, theta2, theta3, year),
     data=USPop, start=list(theta1=400, theta2=-49, theta3=0.025)))

Nonlinear regression model
  model: population ~ model2(theta1, theta2, theta3, year)
   data: USPop
  theta1   theta2   theta3
440.8334 -42.7070   0.0216
 residual sum-of-squares: 458

Number of iterations to convergence: 6
Achieved convergence tolerance: 1.31e-06
```

The first argument to `deriv()` gives the right side of the model as a one-sided formula; the second argument specifies the names of the parameters, with respect to which derivatives are to be found; and the third argument is a function with an empty body, specifying the arguments for the function that is returned by `deriv`.

8 Fitting Nonlinear Mixed-Effects Models

One extension of the nonlinear regression model to include random effects, due to Pinheiro and Bates (2000) and available in the `nlme()` function in the **nlme** package, is as follows (but with different notation than in the original source):[6]

$$\begin{aligned} \mathbf{y}_i &= f(\boldsymbol{\theta}_i, \mathbf{X}_i) + \boldsymbol{\varepsilon}_i \\ \boldsymbol{\theta}_i &= \mathbf{A}_i\boldsymbol{\beta} + \mathbf{B}_i\boldsymbol{\delta}_i \end{aligned} \tag{3}$$

where

- $\mathbf{y}_i$ is the $n_i \times 1$ response vector for the n_i observations in the ith of m groups.

[6]The material in this section closely follows Fox (2016, Sec. 24.2), some of it verbatim or nearly so.

- $\mathbf{X}_i$ is a $n_i \times s$ matrix of predictors (some of which may be categorical) for observations in group i.
- $\boldsymbol{\varepsilon}_i \sim \mathbf{N}_{n_i}(\mathbf{0}, \sigma_\varepsilon^2 \boldsymbol{\Lambda}_i)$ is a $n_i \times 1$ vector of multivariately normally distributed errors for observations in group i; the matrix $\boldsymbol{\Lambda}_i$, which is $n_i \times n_i$, is typically parametrized in terms of a much smaller number of parameters, and $\boldsymbol{\Lambda}_i = \mathbf{I}_{n_i}$ if the observations are independently sampled within groups.
- $\boldsymbol{\theta}_i$ is a $n_i \times 1$ *composite coefficient vector* for the observations in group i, incorporating both fixed and random effects.
- $\boldsymbol{\beta}$ is the $p \times 1$ vector of fixed-effect parameters.
- $\boldsymbol{\delta}_i \sim \mathbf{N}_q(\mathbf{0}, \boldsymbol{\Psi})$ is the $q \times 1$ vector of random-effect coefficients for group i.
- $\mathbf{A}_i$ and $\mathbf{B}_i$ are, respectively, $n_i \times p$ and $n_i \times q$ matrices of known constants for combining the fixed and random effects in group i. These will often be "incidence matrices" of zeroes and ones but may also include group-level explanatory variables, treated as conditionally fixed (as in the standard linear model).

We draw data and a model for an example from a study by Wong et al. (2001) of recovery of IQ following coma.[7] The data pertain to 200 patients who sustained traumatic brain injuries resulting in comas of varying duration. After awakening from their comas, patients were periodically administered a standard IQ test. We will model recovery of *performance IQ* (*PIQ*) post-coma; the data set includes a measure of *verbal IQ*, which was also examined in the original source.

We take a preliminary look at the data, which reside in the `Wong` data frame in **carData** package:

```
ord <- with(Wong, order(id, days))
Wong <- Wong[ord, ]
brief(Wong, rows=c(10, 10))
```

```
331 x 7 data.frame (311 rows omitted)
      id days duration    sex    age piq viq
     [i]  [n]      [i]    [f]    [n] [i] [i]
286  405  986        0 Male   21.470  66 116
287  626  870       55 Male   19.754  80  85
303  651 1491       21 Male   22.007  71  94
325  651 3412       21 Male   22.007  68  92
265  781  714       15 Male   29.870  85  85
167 1048   85       94 Male   20.115  63  82
266 1048  576       94 Male   20.115  91  96
288 1075  907       42 Female 27.277  63  64
320 1075 2259       42 Female 27.277  78  79
168 1085  159       11 Male   30.710 103  97
. . .
93  7084   54        2 Male   36.572  87  93
42  7120   39        0 Male   69.706  84  86
116 7173   84        4 Male   24.980  72  75
206 7173  210        4 Male   24.980  79  78
117 7221   98        0 Male   63.504  74  79
94  7271   55        0 Male   41.766 100  95
```

[7] We am grateful to Georges Monette for making the data and associated materials available to us. The analysis reported here is very similar to that in the original source.

```
43  7309   31        0 Female 50.667  85  95
44  7321   23        0 Male   26.004  84  83
95  7371   55        1 Male   56.786  80  88
45  7548   31        0 Male   24.367 108 106
```

```
nrow(Wong)
```

```
[1] 331
```

```
patients <- unique(with(Wong, id))
length(patients)
```

```
[1] 200
```

```
table(xtabs(~id, data=Wong))
```

```
  1   2   3   4   5
107  61  27   4   1
```

```
with(Wong, sum(days > 1000))
```

```
[1] 40
```

```
plot(piq ~ days, xlab="Days Post Coma",
     ylab="PIQ", xlim=c(0, 1000),
     data=Wong, subset = days <= 1000)
for (patient in patients){
    with(Wong, lines(days[id==patient], piq[id == patient], col="gray"))
}
with(Wong, lines(lowess(days, piq), lwd=2, col="darkblue"))
```

The variable `id` records patient number; `days`, the days post-recovery at which the IQ measurments were taken; `duration`, the length in days of the coma; and `piq`, the measured performance IQ of the patient. The other variables, `sex`, `age`, and `viq` won't enter our analysis and are self-explanatory. We sort the data by patient `id` and `days` within patients. Sorting isn't necessary for fitting a mixed-effects model to the data, but helps us to draw graphs of the data.[8]

About half of the patients in the study (107) completed a single IQ test, but the remainder were measured on two to five irregularly timed occasions, raising the possibility of tracing the trajectory of IQ recovery post-coma. A mixed-effects model is very useful here because it allows us to pool the information in the small number of observations available per patient to estimate the typical within-subject trajectory of recovery along with variation in this trajectory. The data are graphed in Figure 6, which shows the individual patients' `piq` trajectories post-coma along with an average trajectory computed by the `lowess()` nonparametric-regression function. This graph is a scatterplot of `piq` versus number of `days` post-coma at the time of measurement, with the observations for each patient connected by lines. Forty of the 331 measurements were taken after 1000 days post-coma, and these are omitted from the graph to allow us to discern more clearly the general pattern of the data.

After examining the data, Wong et al. (2001) posited an *asymptotic growth model* for IQ recovery:

$$
\begin{aligned}
y_{ij} &= \theta_{1i} + \theta_{2i} e^{-\theta_{3i} x_{1ij}} + \varepsilon_{ij} \\
\theta_{1i} &= \beta_1 + \beta_2 \sqrt{x_{2i}} + \delta_{1i} \\
\theta_{2i} &= \beta_3 + \beta_4 \sqrt{x_{2i}} + \delta_{2i} \\
\theta_{3i} &= \beta_5 + \delta_{3i}
\end{aligned}
\tag{4}
$$

where the variables and parameters of the model have the following interpretations:

[8] Actually, the `Wong.txt` data file is effectively presorted within patients, and so this step isn't strictly necessary.

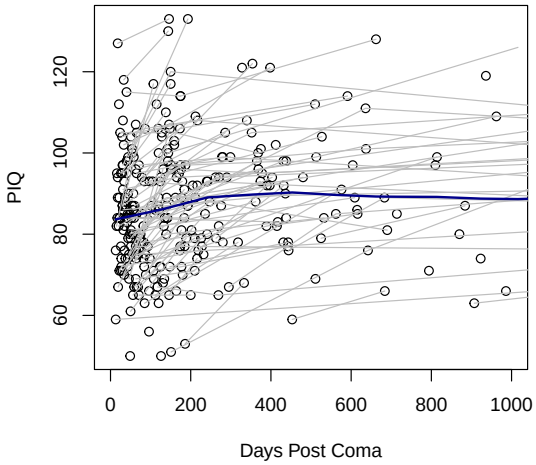

Figure 6: Individual trajectories of `piq` post coma (circles and light gray lines) and average `piq` as a function of `days` of measurement post-coma (solid dark-blue line), as determined by nonparametric regression.

- y_{ij} is the `piq` of the ith patient measured on the jth occasion, $j = 1, \ldots, n_i$; as mentioned, $n_i = 1$ for about half the patients.
- x_{1ij} is the time post-coma (in `days`) for the ith patient at the jth occasion.
- x_{2i} is the `duration` of the coma (in days) for the ith patient.
- θ_{1i} is the eventual, recovered level of `piq` for patient i, specified to depend linearly on the square-root of the `duration` of the coma, with fixed-effect parameters β_1 and β_2, and a random-effect component δ_{1i}. Were patients to recover `piq` fully, the average value of θ_{1i} would be 100, assuming that coma patients are representative of the general population in their pre-coma average level of IQ. Thus, the fixed-effect intercept β_1 is interpretable as the expected eventual level of `piq` for a patient in a coma of zero days `duration`.
- θ_{2i} is the negative of the amount of `piq` eventually regained by patient i, beginning at the point of recovery from coma. Like θ_{1i}, the coefficient θ_{2i} has a fixed-effect component depending linearly on the square-root `duration` of coma, with parameters β_3 and β_4, and a random-effect component, δ_{2i}.
- θ_{3i} is the recovery rate for patient i, treated as a fixed effect, β_5, plus a random effect, δ_{3i}, with $(\log 2)/\theta_{3i}$ representing the time required to recover half the difference between final and (expected) initial post-coma `piq` (the *half-recovery time*), that is, $-\theta_{2i}/2$.
- ε_{ij} is the error for patient i on occasion j.

There are, therefore, seven variance-covariance components in this model: $\text{Var}(\varepsilon_{ij}) = \sigma^2_\varepsilon$; three variances for the δs, $\text{Var}(\delta_{ki}) = \psi^2_k$, $k = 1, 2, 3$; and three covariances among the δs, $\text{Cov}(\delta_{ki}, \delta_{k'i}) = \psi_{kk'}$, $k < k' = 1, 2, 3$. Although the data are longitudinal, there are too few observations per patient to entertain a model with serially correlated errors, and indeed, we'll find that the model as specified is too elaborate for the data.

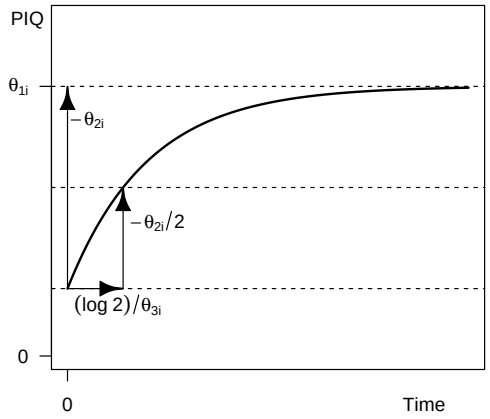

Figure 7: Interpretation of the parameters for patient i in the asymptotic growth model.

Figure 7, produced by the following R code, shows the interpretation of the patient-specific parameters in the asymptotic growth model:

```
library(sfsmisc) # for p.arrows()
plot(c(0, 10), c(0, 5), axes=FALSE, xlab="", ylab="", type="n")
axis(1, at=0)
mtext("Time", side=1, line=1, adj=0.9)
text(-1, 5, labels="PIQ", xpd=TRUE)
axis(2, at=c(0, 4), labels=c(expression(0), expression(theta["1i"])), las=1)
box()
curve(4 - 3*exp(-0.5*x), 0, 10, lwd=2, add=TRUE)
abline(h=c(1, 2.5, 4), lty=2)
p.arrows(0, 1, 0, 4, fill="black")
text(0.5, 3.5, labels=expression(-theta["2i"]))
p.arrows(log(2)/0.5, 1, log(2)/0.5, 2.5, fill="black")
text(2.25, 2.0, labels=expression(-theta["2i"]/2))
p.arrows(0, 1, log(2)/0.5, 1, fill="black")
text(1.25, 0.75, labels=expression((log~2)/theta["3i"]))
```

Figure 6 helps both to determine whether the posited model seems reasonable for the data, and to provide rough guesses for the fixed-effects parameters. As in nonlinear least squares, initial guesses of the fixed-effects parameters provide a starting point for the iterative process of maximizing the likelihood in the nonlinear mixed model. The lowess line on the graph combines the observations from all patients, and is therefore difficult to interpret, but, on the other hand, there are too few observations for each patient to establish clear individual trajectories. Nevertheless, the asymptotic growth model is roughly consistent with the general pattern of the data, and the patients for whom there are multiple observations do tend to improve over time.

Figure 8 shows the relationship between the first post-coma `piq` measurement for each patient and the `duration` of his or her coma on the square-root scale. The graph is restricted to patients whose coma lasted 100 days or less. This graph, along with Figure 6, is helpful in selecting starting

values for the model parameters.

```
Wong.first <- aggregate(Wong[, c("piq", "duration", "days")],
    by=list(id=Wong$id), function(x) x[1])
Wong.first <- subset(Wong.first, duration <= 100)
brief(Wong.first)
```

```
197 x 4 data.frame (192 rows omitted)
      id piq duration days
     [i] [i]      [i]  [n]
1    405  66        0  986
2    626  80       55  870
3    651  71       21 1491
. . .
199 7371  80        1   55
200 7548 108        0   31
```

```
plot(piq ~ sqrt(duration), data=Wong.first,
     xlab="Days in Coma (square-root scale)", ylab="Initial PIQ",
     axes=FALSE, frame=TRUE)
axis(2)
axis(1, at=sqrt(c(0, 5, 10, seq(20, 100, by=20))),
    labels=c(0, 5, 10, seq(20, 100, by=20)))
(mod.ag <- lm(piq ~ sqrt(duration), data=Wong.first))
```

```

Call:
lm(formula = piq ~ sqrt(duration), data = Wong.first)

Coefficients:
   (Intercept)  sqrt(duration)
         88.49           -1.93
```

```
abline(mod.ag, lwd=2, lty=2)
with(Wong.first,
    lines(lowess(sqrt(duration), piq), lwd=2, col="darkblue"))
```

We use the `aggregate()` function to create a data set that contains the first `piq` score for each patient, along with the patient's `duration` of coma and the `days` post-coma at the first IQ measurement, and then limit the data set to durations of 100 days or less. The first measurement for each patient was *never* taken immediately upon recovery from coma (i.e., at `days` = 0), complicating interpretation, but our purpose is to get roughly reasonable start values for the parameters.

- Figure 6 leads us to expect that the average eventual level of recovered `piq` will be less than 100, but Figure 8 also suggests that the average initial and hence eventual level for those who spent fewer days in a coma should be somewhat higher. We therefore use the start value $\beta_1 = 100$.
- The slope of the least-squares line in Figure 8, relating initial `piq` to the square-root of `duration` of coma, is -1.9, and thus we take $\beta_2 = -2$.
- The parameter β_3 represents the negative of the expected eventual gain in `piq` for a patient who spent zero days in a coma. On the basis of the graphs, we guess that such patients start on average at a `piq` of 90 and eventually recover to an average of 100, suggesting the start value $\beta_3 = -10$.

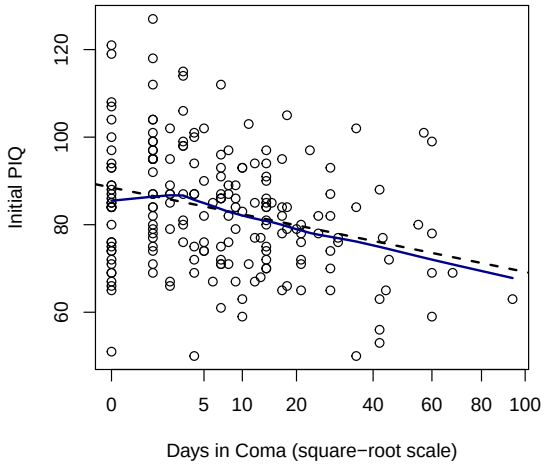

Figure 8: Scatterplot of initial `piq` for each patient versus `duration` of coma. The black broken line is the least-squares line, the solid dark blue line is a nonparametric-regression smooth.

- The parameter β_4 represents the change in expected eventual `piq` gain with a one-unit increase in the `duration` of the coma on the square-root scale. Our examination of the data does not provide a basis for guessing the value of this parameter, and so we take $\beta_4 = 0$.
- Recall that the time to half-recovery is $(\log 2)/\beta_5$. From Figure 6, it seems reasonable to guess that the half-recovery time is around 100 days. Thus, $\beta_5 = (\log 2)/100 = 0.007$.

With the start values in hand, we call `nlme()` to fit the model:[9]

```
piq.mod.1 <- nlme(piq ~ theta1 + theta2*exp(-theta3*days), data=Wong,
    fixed=list(
        theta1 ~ 1 + sqrt(duration),
        theta2 ~ 1 + sqrt(duration),
        theta3 ~ 1),
    random=list(id = list(theta1 ~ 1, theta2 ~ 1, theta3~1)),
    start=list(fixed=c(100, -2, -10, 0, 0.007)),
    control=nlmeControl(msMaxIter=500))
```

```
Warning in nlme.formula(piq ~ theta1 + theta2 * exp(-theta3 * days), data = Wong:
Iteration 3, LME step: nlminb() did not converge (code = 1).
PORT message: function evaluation limit reached without convergence (9)

Warning in nlme.formula(piq ~ theta1 + theta2 * exp(-theta3 * days), data = Wong:
Singular precision matrix in level -1, block 1
```

The specification of the model follows the general nonlinear mixed model in Equation 3 (page 20). Unlike for linear mixed models fit by `lme()`, the structure of the model is specified hierarchically.

[9]The default method in `nlme()` for fitting nonlinear mixed models is maximum likelihood, obtained explicitly by the argument `method="ML"`. Alternatively, we can fit by `method="REML"`, but that proves even more unstable for this problem. We invite the reader to refit the final model by REML, say `piq.mod.3r <- update(piq.mod.3, method="REML")`, and compare the results.

The first (`formula`) argument is expressed in terms of patient-specific coefficients and is similar to the formula for a nonlinear regression model fit by `nls()` (see Section **??**). The `fixed` argument specifies relationships between the subject-specific coefficients and subject-level characteristics, here `duration`. The `random` argument specifies the random-effect structure of the model, which is here just a random error associated with each subject-specific coefficient, allowing these coefficients to vary by subject.

There are numerical problems in fitting this model, and so we consider simpler random-effects structures by successively removing δ_3 and δ_2 from the model:

```
piq.mod.2 <- nlme(piq ~ theta1 + theta2*exp(-theta3*days), data=Wong,
    fixed=list(
        theta1 ~ 1 + sqrt(duration),
        theta2 ~ 1 + sqrt(duration),
        theta3 ~ 1),
    random=list(id = list(theta1 ~ 1, theta2 ~ 1)),
    start=list(fixed=c(100, -2, -10, 0, 0.007)))
```

```
Warning in nlme.formula(piq ~ theta1 + theta2 * exp(-theta3 * days), data = Wong,:
Iteration 1, LME step: nlminb() did not converge (code = 1).
Do increase 'msMaxIter'!
```

```
Warning in nlme.formula(piq ~ theta1 + theta2 * exp(-theta3 * days), data = Wong,:
Iteration 2, LME step: nlminb() did not converge (code = 1).
Do increase 'msMaxIter'!
```

```
piq.mod.3 <- nlme(piq ~ theta1 + theta2*exp(-theta3*days), data=Wong,
    fixed=list(
      theta1 ~ 1 + sqrt(duration),
      theta2 ~ 1 + sqrt(duration),
      theta3 ~ 1),
    random=list(id = list(theta1 ~ 1)),
    start=list(fixed=c(100, -2, -10, 0, 0.007)))
```

Only the simplest model, `piq.mod.3`, fits without difficulty. Likelihood-ratio tests, AIC, and BIC suggest that there's no reason to prefer the more complex models:

```
anova(piq.mod.1, piq.mod.2, piq.mod.3)
```

```
          Model df    AIC    BIC  logLik   Test  L.Ratio p-value
piq.mod.1     1 12 2599.4 2645.0 -1287.7
piq.mod.2     2  9 2593.4 2627.6 -1287.7 1 vs 2 0.002278  1.0000
piq.mod.3     3  7 2589.5 2616.2 -1287.8 2 vs 3 0.185001  0.9116
```

The LR tests compare each model to the one immediately above it. Moreover, as is frequently the case, the fixed-effect estimates are quite similar for the three random-effect specifications, even with convergence problems apparent in fitting the models:

```
compareCoefs(piq.mod.1, piq.mod.2, piq.mod.3)
```

```
Calls:
1: nlme.formula(model = piq ~ theta1 + theta2 * exp(-theta3 *
  days), data = Wong, fixed = list(theta1 ~ 1 + sqrt(duration),
  theta2 ~ 1 + sqrt(duration), theta3 ~ 1), random = list(id =
```

```
  list(theta1 ~ 1, theta2 ~ 1, theta3 ~ 1)), start = list(fixed
  = c(100, -2, -10, 0, 0.007)), control = nlmeControl(msMaxIter
  = 500))
2: nlme.formula(model = piq ~ theta1 + theta2 * exp(-theta3 *
  days), data = Wong, fixed = list(theta1 ~ 1 + sqrt(duration),
  theta2 ~ 1 + sqrt(duration), theta3 ~ 1), random = list(id =
  list(theta1 ~ 1, theta2 ~ 1)), start = list(fixed = c(100, -2,
   -10, 0, 0.007)))
3: nlme.formula(model = piq ~ theta1 + theta2 * exp(-theta3 *
  days), data = Wong, fixed = list(theta1 ~ 1 + sqrt(duration),
  theta2 ~ 1 + sqrt(duration), theta3 ~ 1), random = list(id =
  list(theta1 ~ 1)), start = list(fixed = c(100, -2, -10, 0,
  0.007)))

                      Model 1 Model 2 Model 3
theta1.(Intercept)      97.09   97.09   98.15
SE                       2.02    2.02    2.02

theta1.sqrt(duration)  -1.245  -1.245  -1.236
SE                      0.477   0.477   0.462

theta2.(Intercept)     -11.14  -11.15  -12.14
SE                       3.18    3.18    3.01

theta2.sqrt(duration)  -3.249  -3.248  -2.860
SE                      1.069   1.069   0.968

theta3                0.00825 0.00825 0.00621
SE                    0.00164 0.00164 0.00125
```

summary(piq.mod.3)

```
Nonlinear mixed-effects model fit by maximum likelihood
  Model: piq ~ theta1 + theta2 * exp(-theta3 * days)
 Data: Wong
     AIC    BIC  logLik
  2589.5 2616.2 -1287.8

Random effects:
 Formula: theta1 ~ 1 | id
        theta1.(Intercept) Residual
StdDev:             12.908   6.6948

Fixed effects: list(theta1 ~ 1 + sqrt(duration), theta2 ~ 1 + sqrt(duration),
                    theta3 ~ 1)
                       Value Std.Error  DF t-value p-value
theta1.(Intercept)     98.152   2.03824 127  48.156  0.0000
theta1.sqrt(duration)  -1.236   0.46558 127  -2.656  0.0089
theta2.(Intercept)    -12.136   3.03679 127  -3.996  0.0001
theta2.sqrt(duration)  -2.860   0.97565 127  -2.932  0.0040
theta3                  0.006   0.00126 127   4.912  0.0000
```

```
 Correlation:
                      t1.(I) th1.() t2.(I) th2.()
theta1.sqrt(duration) -0.712
theta2.(Intercept)    -0.584  0.444
theta2.sqrt(duration)  0.455 -0.443 -0.805
theta3                -0.352  0.009  0.128 -0.336

Standardized Within-Group Residuals:
      Min        Q1       Med        Q3       Max
-3.067982 -0.375418  0.013861  0.375336  2.460507

Number of Observations: 331
Number of Groups: 200
```

Focusing then on model `piq.mod.3`, the average final level of recovered `piq` following a coma of zero days is $\widehat{\beta}_1 = 98.15$, and this level declines with the square-root `duration` of the coma at the rate $\widehat{\beta}_2 = -1.236$ points per square-root day (e.g., from day 1 to day 4 or from day 4 to day 9). On average, patients who spend zero days in coma recover $-\widehat{\beta}_3 = 12.14$ `piq` points, and the size of the recovery grows with the square-root `duration` of the coma, $-\widehat{\beta}_4 = 2.860$. The estimated half-recovery time is $(\log 2)/\widehat{\beta}_5 = (\log 2)/0.00621 = 112$ days. We can use the delta method to get a standard error and confidence limits (which turn out to be wide) for this estimate:

```
deltaMethod(piq.mod.3, "log(2)/beta5", parameterNames=paste0("beta", 1:5))
```

```
             Estimate     SE  2.5 % 97.5 %
log(2)/beta5   111.67 22.564 67.446 155.89
```

Figure 9 is a fixed-effect display for the model, similar to one that appears in Wong et al. (2001), showing estimated average `piq` as a function of `duration` of coma and `days` post coma:

```
newdata <- expand.grid(duration=c(1, 10, 20, 50, 100, 200),
                       days=seq(0, 1000, 20))
newdata$piq <- predict(piq.mod.3, newdata, level=0)
plot(piq ~ days, type="n", xlab="Days Post Coma", ylab="Average PIQ",
    ylim=c(20, 100), xlim=c(-100, 1000), data=newdata, axes=FALSE, frame=TRUE)
axis(2) # left
axis(4) # right
axis(1, at=seq(0, 1000, by=100)) # bottom
grid(lty=2, col="gray")
for (dur in c(1, 10, 20, 50, 100, 200)){
    with(newdata, {
        lines(spline(seq(0, 1000, 20), piq[duration == dur]), lwd=2)
        text(-25, piq[duration == dur][1], labels=dur, adj=0.9)
    })
}
text(-100, 95, labels="Duration\nof Coma", adj=0)
```

9 Complementary Reading and References

Nonlinear regression and nonlinear least squares are discussed in Weisberg (2014, Chap. 11) and Fox (2016, Chap. 17), and in Ritz and Streibig (2008). Bates and Watts (1988) present a comprehensive

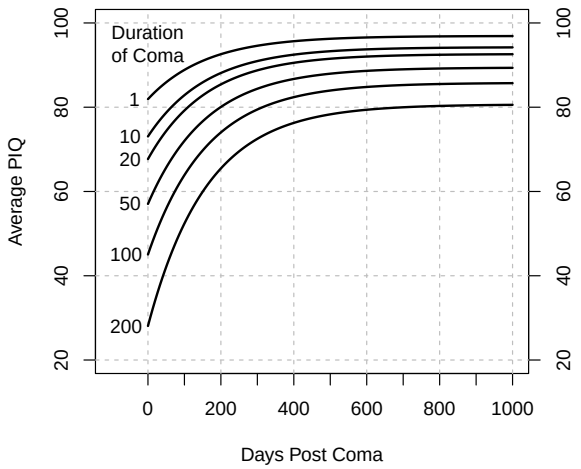

Figure 9: Estimated average `piq` as a function of `days` post coma and `duration` of coma.

treatment of the subject; a brief introduction is provided by Gallant (1975). The use of the `nls()` function in S is described in detail by Bates and Chambers (1992); much of this material is also relevant to R. Also see Pinheiro and Bates (2000, Sec. 8.1). Nonlinear mixed-effects models in S (and R) are discussed in Pinheiro and Bates (2000, Chaps. 6–8).

References

Bates, D. and Watts, D. (1988). *Nonlinear Regression Analysis and Its Applications*. Wiley, New York.

Bates, D. M. and Chambers, J. M. (1992). Nonlinear models. In Chambers, J. M. and Hastie, T. J., editors, *Statistical Models in S*, pages 421–454. Wadsworth, Pacific Grove, CA.

Fox, J. (2016). *Applied Regression Analysis and Generalized Linear Models*. Sage, Thousand Oaks CA, third edition.

Fox, J. and Weisberg, S. (2019). *An R Companion to Applied Regression*. Sage, Thousand Oaks, CA, third edition.

Gallant, A. R. (1975). Nonlinear regression. *The American Statistician*, 29:73–81.

Pinheiro, J. C. and Bates, D. M. (2000). *Mixed-Effects Models in S and S-PLUS*. Springer, New York.

Ratkowsky, D. A. (1990). *Handbook of Nonlinear Regression Models*. Marcel Dekker, New York.

Ritz, C. and Streibig, J. (2008). *Nonlinear regression with R*. Springer Verlag.

Weisberg, S. (2014). *Applied Linear Regression*. Wiley, Hoboken NJ, fourth edition.

Weisberg, S., Spangler, G., and Richmond, L. S. (2010). Mixed effects models for fish growth. *Can. J. Fish. Aquat. Sci.*, 67(2):269–277.

Wong, P. P., Monette, G., and Weiner, N. I. (2001). Mathematical models of cognitive recovery. *Brain Injury*, 15:519–530.