Nonlinear Regression and Nonlinear Least Squares in R

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

John Fox & Sanford Weisberg

last revision: 13 December 2010

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 example, 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 (2011) 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.

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 **car** package has decennial U. S. Census population for the United States (in millions), from 1790 through 2000. The data are shown in Figure 1a:[1]

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

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 rate 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, the population size, 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. We assume the predictor $\mathbf{x}$ consists of fixed, known values.

[1] The `plot` and `abline` functions and other R functions used but not described in this appendix are discussed in Fox and Weisberg (2011). All the R code used in this appendix can be downloaded in the file `http://socserv.socsci.mcmaster.ca/jfox/Books/Companion/Appendix-Nonlinear-Regression.R`. Alternatively, if you are running R and attached to the Internet, load the **car** package and enter the command `carWeb("Appendix-Nonlinear-Regression.R")` to view the R command file for the appendix in your browser.

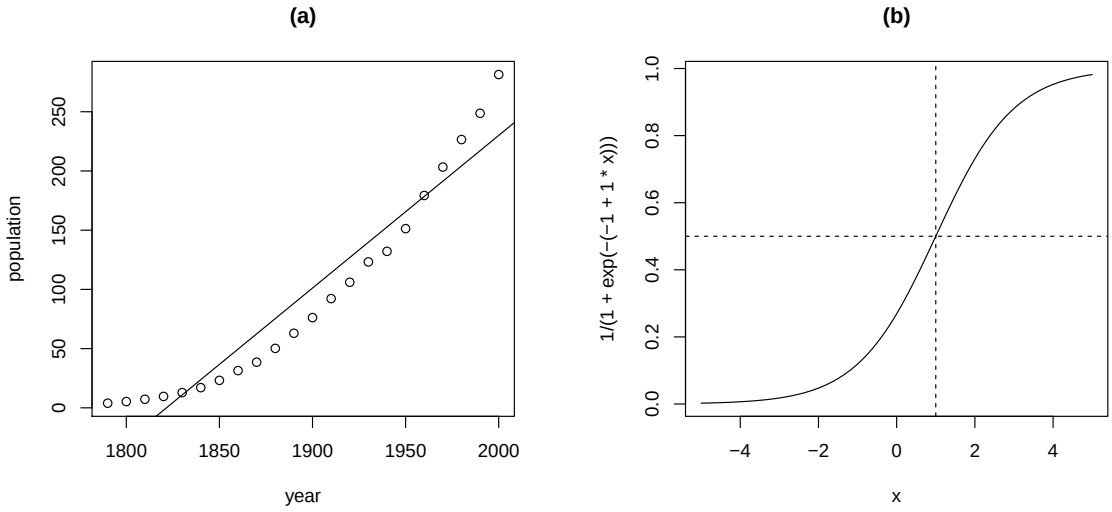

Figure 1: (a) U. S. population, from the U. S. Census, and (b) a generic logistic growth curve.

As a function of x, the mean function in Equation 1 is a curve, as shown in Figure 1b 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, lty=2)
> abline(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 1b can match the data shown in Figure 1a, 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 R function `nls` is used for estimating parameters via nonlinear least squares. Following Weisberg (2005, Chap. 11), the general nonlinear regression model is

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

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.[2] In most 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,

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

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 formula that provides the minimizer of Equation 2. Rather 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.

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 gives 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:

```
> 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.

[2] 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.

`formula` The `formula` argument is used to tell `nls` about the mean function. The formula equivalent to Equation 1 (page 1) 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 from 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 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; see Fox and Weisberg (2011, Sec. 1.1.5). We can name the parameters with any legal R name, 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. It 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. Because no starting value is given for `year`, it is by default taken by `nls` to be a predictor. The `start` argument is required unless we are using a self-starting function 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` One of the characteristics of nonlinear models is that the 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 takes a list of values that modify the criteria used in 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 this 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, weights, na.action` 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 argument `weights` is missing then all weights are set equal to 1.

2 Starting Values

Unlike in linear least squares, 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 1).[3] For the logistic growth model 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 is some value larger than any value in the data, and so a value of around $t_1 = 400$ is a reasonable start. (The estimated population in 2010 before the official Census count was released is 307 million.) 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
```

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)
```

[3] 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.

```
3061 :  400.000 -49.000   0.025
558.5 :  426.06199 -42.30786   0.02142
458 :  438.41472 -42.83690   0.02168
457.8 :  440.8903 -42.6987   0.0216
457.8 :  440.81681 -42.70805   0.02161
457.8 :  440.83445 -42.70688   0.02161
457.8 :  440.83333 -42.70698   0.02161
```

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.

As in `lm`, 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 440.83333   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: 1.48e-06
```

The column marked `Estimates` displays the least squares estimates. The estimated upper bound for the U. S. population is 440.8, or about 441 million. The column marked `Std. Error` displays the estimated standard errors of these estimates. The very large standard error for the asymptote reflects the uncertainty in the estimated asymptote when all the observed data is 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 with the `deltaMethod` function in the **car** package:

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

               Estimate    SE
-theta2/theta3     1977 7.556
```

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. The residual standard deviation is the estimate of σ,

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

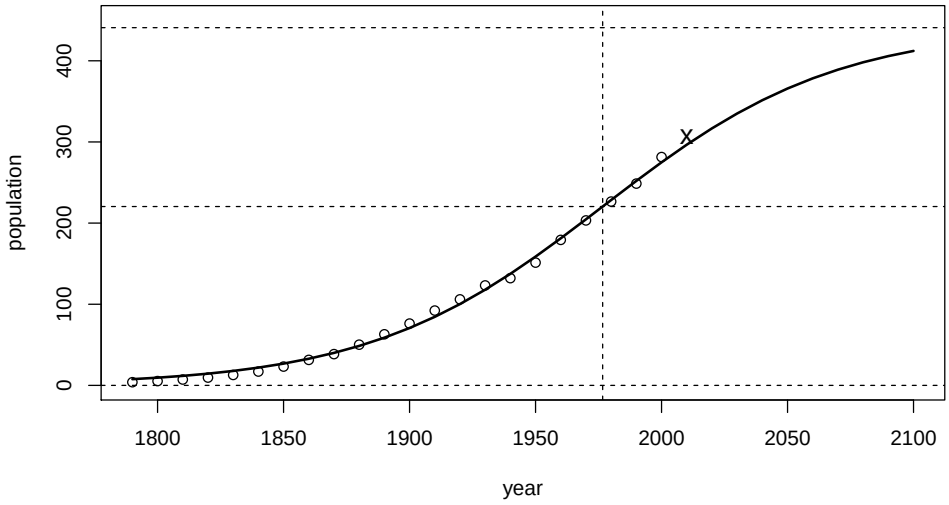

Figure 2: U. S. population, with logistic growth fit extrapolated to 2100. The circles represent observed Census population counts, while "x" represents the estimated 2010 population. The broken horizontal lines are drawn at the asymptotes and midway between the asymptotes; the broken vertical line is drawn at the year corresponding to the mid-way point.

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

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, 307, pch="x", cex=1.3)
> abline(h=0, lty=2)
> abline(h=coef(pop.mod)[1], lty=2)
> abline(h=.5*coef(pop.mod)[1], lty=2)
> abline(v= -coef(pop.mod)[2]/coef(pop.mod)[3], lty=2)
```

We have made this plot more complicated than needed 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 estimated population in 2010 (before the 2010 Census was released), 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:

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

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

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).

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: 3.82e-06
```

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
1/theta3    46.28 2.157
```

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.

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

Nonlinear regression model
  model:  population ~ SSlogis(decade, nu1, nu2, nu3)
   data:  USPop
   nu1    nu2    nu3
440.83  18.66   4.63
 residual sum-of-squares: 458

Number of iterations to convergence: 0
Achieved convergence tolerance: 3.81e-06
```

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.

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 function `bootCase` in the **car** package will compute a case bootstrap that can be used for inference about nonlinear least squares estimates.[4] For example, the following command computes $B = 999$ bootstrap samples and retain the values of the coefficient estimates for each bootstrap sample:

```
> set.seed(12345) # for repeatability
> out4 <- bootCase(pop.ss, B=999)
> data.frame(summary(pop.ss)$coef[, 1:2],
+     bootMean=apply(out4, 2, mean), bootSD=apply(out4, 2, sd))

     Estimate Std..Error bootMean bootSD
phi1   440.83     35.000   420.12 55.451
phi2  1976.63      7.556  1971.62 12.671
phi3    46.28      2.157    45.06  2.942
```

The differences between the estimates and the bootstrap means suggest substantial bias in the estimates. The nominal standard errors are also considerably smaller than the standard errors based on the bootstrap. Histograms of the bootstrap samples, not shown, reveal that the marginal distribution of each parameter estimate is moderately to severely skewed, suggesting that normal inference is a poor idea.

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

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

a mean function is largely arbitrary, as only characteristics of the curve, such as an asymptote, are known in advance. Ratkowsky (1990) provides a very helpful catalog of nonlinear mean functions that are 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.

6 Nonlinear Models Fit with a Factor

A common problem with nonlinear models is that we would like to fit a model with 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:

```
> Data <- data.frame(rbind(data.frame(country="US", USPop[,1:2]),
+                          data.frame(country="Canada", CanPop)))
> some(Data)
```

```
    country year population
2        US 1800      5.308
6        US 1840     17.063
10       US 1880     50.189
16       US 1940    132.165
20       US 1980    226.542
21       US 1990    248.710
71   Canada 1911      7.207
```

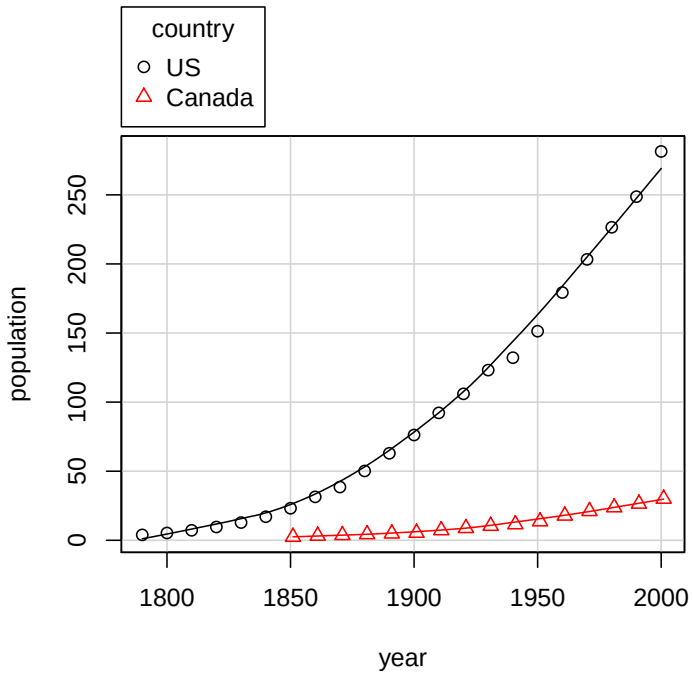

Figure 4: U. S. and Canadian population at decennial censuses; the curves were computed by nonparametric regression.

```
91    Canada 1931      10.377
111   Canada 1951      13.648
151   Canada 1991      26.429
```

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

```
> scatterplot(population ~ year|country, data=Data, 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` suppress 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 function `nlsList` 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 larger U. S. is more variable than Canada. We use the argument `pool=FALSE` to force separate estimates of variance in each country:

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

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

Coefficients:
   phi1
        Estimate Std. Error t value  Pr(>|t|)
US       440.83      35.00  12.595 1.139e-10
Canada    71.45      14.15   5.049 2.228e-04
   phi2
        Estimate Std. Error t value  Pr(>|t|)
US         1977      7.556   261.6 2.942e-35
Canada     2016     16.475   122.3 2.730e-21
   phi3
        Estimate Std. Error t value  Pr(>|t|)
US        46.28      2.157   21.45 8.867e-15
Canada    47.75      3.060   15.60 8.477e-10
```

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

    US Canada
4.9087 0.5671
```

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 the rate 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
 440.83 1976.63   46.28

$Canada
   phi1    phi2    phi3
  71.45 2015.66   47.75

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

$US
        phi1   phi2   phi3
phi1 1225.02 262.86 69.128
phi2  262.86  57.09 15.229
phi3   69.13  15.23  4.655
```

```
$Canada
       phi1   phi2   phi3
phi1 200.22 232.48 40.835
phi2 232.48 271.42 48.461
phi3  40.83  48.46  9.364
```

We then use `unlist` to combine the estimates into a single vector, and also combine the two variance matrices into a single block-diagonal matrix:

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

    US.phi1     US.phi2     US.phi3 Canada.phi1 Canada.phi2 Canada.phi3
     440.83     1976.63       46.28       71.45     2015.66       47.75

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

        phi1   phi2   phi3
phi1 1225.02 262.86 69.128   0.00   0.00  0.000
phi2  262.86  57.09 15.229   0.00   0.00  0.000
phi3   69.13  15.23  4.655   0.00   0.00  0.000
phi1    0.00   0.00  0.000 200.22 232.48 40.835
phi2    0.00   0.00  0.000 232.48 271.42 48.461
phi3    0.00   0.00  0.000  40.83  48.46  9.364

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

                      Estimate    SE
US.phi3 - Canada.phi3   -1.464 3.744

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

                      Estimate    SE
US.phi2 - Canada.phi2   -39.03 18.12
```

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

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

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 variances are different for the two countries, we can do this only approximately, by setting weights as follows:

```
> w <- ifelse(Data$country=="Canada", (sds[1]/sds[2])^2, 1)
```

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.. Then one parametrization of the largest model we contemplate is

```
> Data$can <- ifelse(Data$country=="Canada", 1, 0)
> 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=Data, weights=w, start=list(phi11=b[1, 1], phi12=b[1, 2],
+        phi21=b[1, 2], phi22=b[2, 2], phi31=b[1, 3], phi32=b[2, 3]))
```

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=Data, 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    0.7
```

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

```
> confint(m2)

         2.5%   97.5%
phi11  396.01  526.07
phi12   55.33   89.35
phi21 1966.60 1993.38
phi22 1994.65 2033.30
phi3    43.48   50.45
```

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 groups, an appropriate model may be a *nonlinear mixed model*. These are described in Pinheiro and Bates (2000, Part II), and can be fit using the `nlme` function in the **nlme** package.

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 function `model` 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 Complementary Reading and References

Nonlinear regression and nonlinear least squares are discussed in Weisberg (2005, Chap. 11) and Fox (2008, Chap. 15). Bates and Watts (1988) present a comprehensive 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).

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. (2008). *Applied Regression Analysis and Generalized Linear Models*. Sage, Thousand Oaks, CA, second edition.

Fox, J. and Weisberg, S. (2011). *An R Companion to Applied Regression*. Sage, Thousand Oaks, CA, second 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.

Weisberg, S. (2005). *Applied Linear Regression*. John Wiley & Sons, Hoboken, NJ, third 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.