E.g., suppose we have a model with expectation function $f(x, \boldsymbol{\theta}) = \theta_1/(\theta_2 + x)$ and we want $\partial f(x, \boldsymbol{\theta})/(\partial \theta_2)$ evaluated at $\boldsymbol{\theta} = (1, 1)^T$ and $x = 2$. Then the forward difference approximation is

$$\frac{\partial f(x, \boldsymbol{\theta})}{\partial \theta_2} \approx \frac{1}{h_2} \left(\frac{\theta_1}{\theta_2 + h_2 + x} - \frac{\theta_1}{\theta_2 + x} \right) \Bigg|_{\boldsymbol{\theta}=(1,1)^T, x=2}$$

where $h_2 = \sqrt{\epsilon}(1 + |\theta_2|) = \sqrt{2.2204e-016}(1+1) = 2.9802e-008$, so

$$\begin{aligned} \frac{\partial f(x, \boldsymbol{\theta})}{\partial \theta_2} &\approx \frac{1}{2.9802e-008} \left(\frac{1}{1 + 2.9802e-008 + 2} - \frac{1}{1+2} \right) \\ &= -0.1111111100763083 \end{aligned}$$

- The exact derivative is $-1/9 = -.111\bar{1}$.

Central Difference Approximations:

Some improvement in the accuracy of numerical derivatives can be obtained by using central differences rather then forward differences.

In the general set-up described above, the central difference approximation is

$$\frac{\partial g}{\partial \theta_i} \approx \frac{g(\boldsymbol{\theta} + h_i \mathbf{j}_i) - g(\boldsymbol{\theta} - h_i \mathbf{j}_i)}{2h_i}$$

- For central differences, a larger h_i value is recommended: $h_i = (\epsilon)^{1/3}(1 + |\theta_i|)$.

In the example, the central difference approximation is

$$\begin{aligned} \frac{\partial f(x, \boldsymbol{\theta})}{\partial \theta_2} &\approx \frac{1}{2h_2} \left(\frac{\theta_1}{\theta_2 + h_2 + x} - \frac{\theta_1}{\theta_2 - h_2 + x} \right) \Bigg|_{\boldsymbol{\theta}=(1,1)^T, x=2} \\ &= -0.1111111111129642 \end{aligned}$$

- Note the improved accuracy of the central difference approximation. See NumDerivExamp.R for implementation of this example.

In nls() and gnls() numerical derivatives are used by default. However, it is best to supply these functions with analytic derivatives instead. A convenient way to do this is with the deriv() function.

- The deriv() function uses symbolic mathematics (as in Maple or Mathematica) to produce the analytic derivates. This saves the user from the (very) error-prone task of taking the derivatives him/herself.

For example, consider the model fit to log(Lens) in model m2rabbit.nls. See handout Rabbit2.

- The Rabbit2 handout contains an R script rabbit2.R and its output. It also contains a SAS program, rabbit1.sas, and its output, rabbit1.lst. In rabbit2.R, we first demonstrate that a function can be coded that represents the expectation function of the model. I call this function m2rabbit.

- We can refit model m2rabbit.nls by using the m2rabbit() function. This simplifies things slightly. (Later when we consider adding covariates to nonlinear models, we'll see that this step is very handy.)

- An advantage to coding the function $f(x, \boldsymbol{\theta}) = \theta_1 - \theta_2/(\theta_3 + x)$ as m2rabbit(x,th1,th2,th3) is that we can add a derivative (gradient) attribute to m2rabbit() so that it returns not only the function's value, but also the values of the partial derivatives with respect to each parameter at each evaluation.

- This is done using the deriv() function. We can see what's going on by printing the redefined function m2rabbit and this function evaluated at some given values of Age and $\boldsymbol{\theta}$. See the R documentation on deriv() for more information.

- Now if we refit the model using the m2rabbit() function, analytical derivatives will be used because they are available as an output of this function.

- In rabbit1.sas the same model is fit using SAS' PROC NLIN. As of version 8, PROC NLIN uses analytic derivatives by default. It used to be the case that derivatives had to be coded into PROC NLIN using der.parm statements.

- The automatic calculation of analytic derivatives with symbolic math is a very helpful recent development. Mistakes in the calculation of analytic derivatives by hand are very common, and the elimination of this step by the software makes the fitting of nonlinear models much more reliable. When using software that cannot provide analytic derivatives, it is recommended that a symbolic math program such as Maple be used to compute derivatives when possible.

- As you can see, PROC NLIN and the nls() function produce identical results.

3. Starting Values:

For the G-N and other commonly used algorithms for fitting nonlinear models, starting values of the parameters are needed. Bad starting values can lead to slow or no convergence, or possibly convergence to the wrong solution.

There are several methods for selecting starting values and often some combination of these methods is necessary for a given problem:

1. Analysis of the Expectation Function.

   Starting values can often be obtained by considering the behavior of $f(\mathbf{x}, \boldsymbol{\theta})$ as the x-values approach some limiting value (e.g., 0, $\pm\infty$). This information is often used with the observed y-values.

   E.g., if one of the parameters has an interpretation as the asymptote in a growth model, then we can take the maximum y-value as an initial guess of that parameter.

   E.g., consider the simple logistic model

   $$f(x, \boldsymbol{\theta}) = \frac{\theta_1}{1 + \exp\{(\theta_2 - x)/\theta_3\}}.$$

   As $x \to \infty$, the denominator $\to 1$ which implies that θ_1 is the limiting value for large x. If this is an upper asymptote as in a growth model, then take $\hat{\theta}_1^0 = y_{\max}$ where $y_{\max}$=the largest response.

   In general, it may be possible to solve

   $$y_i = f(\mathbf{x}_i, \boldsymbol{\theta})$$

   for one or more elements of $\boldsymbol{\theta}$ for certain well chosen pairs $(y_i, \mathbf{x}_i)$.

Another example: in the asymptotic regression model,

$$f(x, \boldsymbol{\theta}) = \theta_1 + \theta_2 e^{-\theta_3 x},$$

letting $x \to \infty$ we obtain the asymptote θ_1, and letting $x = 0$ gives the initial value $\theta_1 + \theta_2$. Therefore, we might take $\hat{\theta}_1^0 = y_{\max}$ and $\hat{\theta}_2^0 = y_0 - y_{\max}$ where y_0 is the (observed or "expected") y-value at $x = 0$.

2. Analysis of Derivatives of the Expectation Function.

   Sometimes the derivatives of $f(\mathbf{x}, \boldsymbol{\theta})$ with respect to $\mathbf{x}$ are simpler than f itself and can be solved for one or more components of $\boldsymbol{\theta}$.

   Suppose $\mathbf{x} = x$ consists of just one variable and let $x^{(1)}, \ldots, x^{(n)}$ represent the ordered (e.g., smallest to largest) x-values and $y^{(1)}, \ldots, y^{(n)}$ the corresponding y-values.

   Then if we can solve

$$\underbrace{\frac{y^{(i)} - y^{(i-1)}}{x^{(i)} - x^{(i-1)}}}_{\text{"empirical derivative"}} = \left.\frac{\partial f(x, \boldsymbol{\theta})}{\partial x}\right|_{x=x^{(i)}}$$

   for θ_j then take this solution as the initial value for θ_j.

   For example: In the Michaelis-Menten model, $f(x, \boldsymbol{\theta}) = \frac{\theta_1 x}{\theta_2 + x}$, we have

$$\left.\frac{\partial f(x, \boldsymbol{\theta})}{\partial x}\right|_{x=0} = \left.\frac{\theta_1 \theta_2}{(\theta_2 + x)^2}\right|_{x=0} = \frac{\theta_1}{\theta_2}$$

   In addition, θ_1 has an interpretation as an upper asymptote since $\lim_{x \to \infty} f(x, \boldsymbol{\theta}) = \theta_1$. Therefore, we take $\hat{\theta}_1^0 = y_{\max}$ and solve

$$\frac{y^{(1)} - y^{(0)}}{x^{(1)} - x^{(0)}} = \frac{y_{\max}}{\hat{\theta}_2^0}$$

   for $\hat{\theta}_2^0$ to get an an initial value for θ_2.

3. Transformation of the Expectation Function.

   If the expectation function is transformably linear, then we can fit a linear model to the transformed model (don't need to worry about error term to get roughly accurate parameter estimates) to get initial values.

   E.g., If the expectation function for response R is $f(\mathbf{x}, \boldsymbol{\theta}) = \theta_1 x_1^{\theta_2} x_2^{\theta_3}$ then the log transformation yields

$$\log\{f(\mathbf{x}, \boldsymbol{\theta})\} = \log(\theta_1) + \theta_2 \log(x_1) + \theta_3 \log(x_2).$$

   Therefore, by fitting the linear regression model:

$$\log(R_i) = \beta_1 + \beta_2 \log(x_{1i}) + \beta_3 \log(x_{2i}), \quad i = 1, \ldots, n$$

   we can obtain starting values:

$$\hat{\theta}_1^0 = e^{\hat{\beta}_1}, \quad \hat{\theta}_2^0 = \hat{\beta}_2, \quad \hat{\theta}_3^0 = \hat{\beta}_3.$$

4. Conditional Linearity.

   In many model functions, one or more of the parameters are conditionally linear. In that case, once starting values for the nonlinear parameters have been obtained, then linear regression conditional on the nonlinear parameters' starting values can be used to obtain starting values for the linear parameters.

   For example: consider the asymptotic regression model $y = \theta_1 + \theta_2 e^{-\theta_3 x}$. If a starting value $\hat{\theta}_3^0$ is available, then a simple linear regression of y on $z = e^{-\hat{\theta}_3^0 x}$ will yield starting values for θ_1 and θ_2.

5. Reducing Dimensions.

   Usually, more than one of the above approaches are necessary and are combined in succession. After an initial value of each successive parameter has been obtained, the problem of obtaining starting values for all parameters is simplified because it has been reduced in dimension.

   E.g., in the last example, once $\hat{\theta}_3^0$ has been obtained it becomes easier to obtain $\hat{\theta}_1^0$ and $\hat{\theta}_2^0$ because the problem is conditionally linear.

   Another example is provided by the method of **exponential peeling**. This method is useful when the expectation function is a sum of exponentials.

   Consider the biexponential model

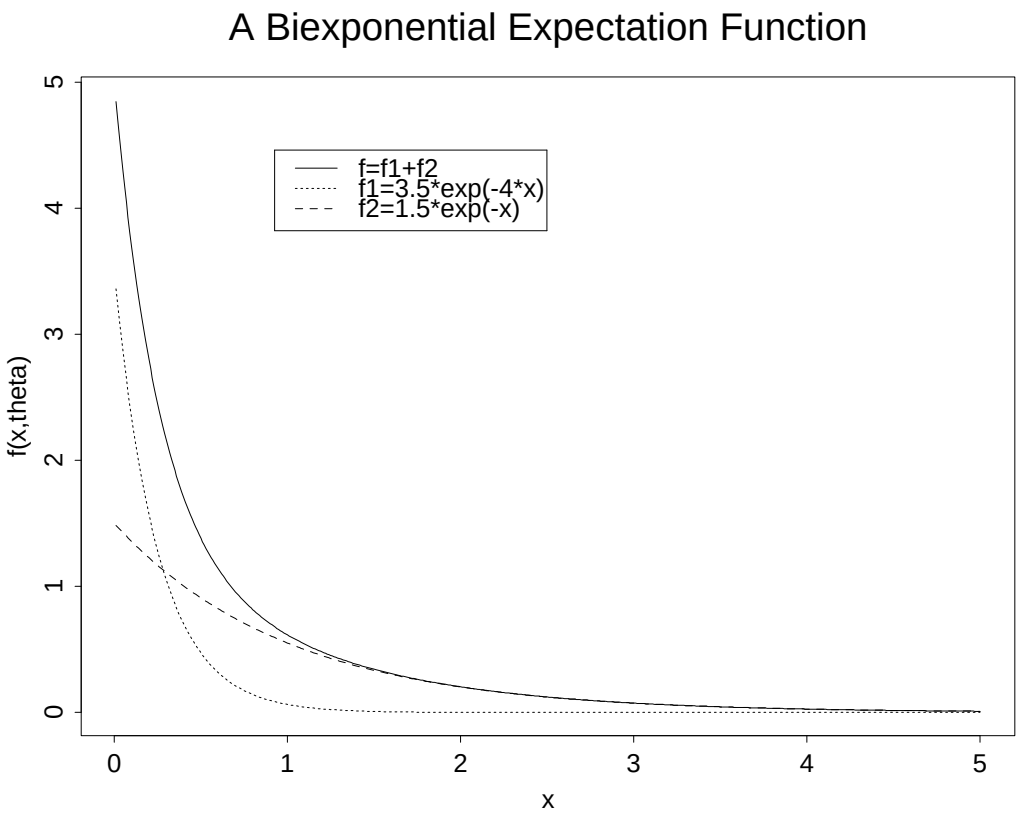
$$f(x, \boldsymbol{\theta}) = \theta_1 e^{-\theta_2 x} + \theta_3 e^{-\theta_4 x}$$

   with $\theta_1, \theta_2, \theta_3, \theta_4$ assumed positive.

   – Notice that in such a model the two pairs of parameters (θ_1, θ_2) and (θ_3, θ_4) are **exchangeable**, meaning that the values of the pairs can be exchanged without changing the value of f.

   – This means that the parameters of the model are not **identifiable**. To get around this we create an identifiable parameterization by requiring $\theta_2 > \theta_4$.

The following plot gives an example of a biexponential where $\boldsymbol{\theta} = (3.5, 4, 1.5, 1)^T$:



Because $\theta_2 > \theta_4$ the function behaves like a simple exponential $\theta_3 e^{-\theta_4 x}$ for large x, and like $\theta_1 e^{-\theta_2 x} + \theta_3$ for small x.

This suggests splitting the data into a portion corresponding to the largest x values and the remaining portion corresponding to the smallest x values. Then fit obtain starting values by fitting the components as follows:

i. First fit the simple linear regression

$$\log(y_i) = \alpha_0 + \alpha_1 x_i,$$

for the portion of the data with the largest values of x and set $\hat{\theta}_3^0 = e^{\hat{\alpha}_0}$ and $\hat{\theta}_4^0 = -\hat{\alpha}_1$.

ii. Then calculate the residuals, $r_i = y_i - \hat{\theta}_3^0 \exp(-\hat{\theta}_4^0 x_i)$ for the portion of the data with the smallest x values. Fit the simple linear regression model

$$\log(|r_i|) = \beta_0 + \beta_1 x_i$$

and set $\hat{\theta}_1^0 = e^{\hat{\beta}_0}$ and $\hat{\theta}_2^0 = -\hat{\beta}_1$.

Example 1 — Rabbit Age and Eye Weight:

Again consider the model

$$y_i = \theta_1 - \frac{\theta_2}{\theta_3 + x_i} + e_i$$

where

$$y = \log(\text{eye lens weight})$$
$$x = \text{age in days}$$

From a plot of y versus x we saw that y reaches an asymptote as x gets large.

As $x \to \infty$, $f(x, \boldsymbol{\theta}) \to \theta_1$ so we can take

$$\hat{\theta}_1^0 = y_{\max} = \log(246.70) = 5.51$$

As $x \to 0$, $f(x, \boldsymbol{\theta}) \to \theta_1 - \theta_2/\theta_3$. The minimum x value in the data was $x = 15$ for which there were three observations with y values $\log(21.66)$, $\log(22.75)$, and $\log(22.30)$. We take their average, 3.10, to represent y when x is small.

Now solve

$$\hat{\theta}_1^0 - \frac{\theta_2}{\theta_3} = 3.10$$

to get

$$\frac{\theta_2}{\theta_3} = 5.51 - 3.10 = 2.41$$

Since f is approximately linear near $x = 0$ we can examine

$$\frac{\partial f(x, \boldsymbol{\theta})}{\partial x} = \frac{\theta_2}{(\theta_3 + x)^2} \to \frac{\theta_2}{\theta_3^2} \quad \text{as } x \to 0.$$

The second smallest x value was $x = 18$ with a corresponding y value of $y = \log(31.25)$. Therefore, we can solve

$$\underbrace{\frac{\log(31.25) - 3.10}{18 - 15}}_{\text{empirical derivative}} = \frac{\theta_2}{\theta_3^2} \quad \Rightarrow \frac{\theta_2}{\theta_3^2} = .113$$

We now have the two equations in two unknowns:

$$\frac{\theta_2}{\theta_3} = 2.41$$
$$\frac{\theta_2}{\theta_3^2} = .113$$

which can be solved to get $\hat{\theta}_3^0 = 21.3$ and $\hat{\theta}_2^0 = 51.4$.

- So, our starting values are given by $\hat{\boldsymbol{\theta}}^0 = (5.51, 51.4, 21.4)^T$.

Example — Pressure vs. Temperature in Saturated Steam:

Data:

Temperature	Pressure
0	4.14
10	8.52
20	16.31
30	32.18
40	64.62
50	98.76
60	151.13
70	224.74
80	341.35
85	423.36
90	522.78
95	674.32
100	782.04
105	920.01

Let y =pressure, and x =temperature. Then the model we consider is

$$y_i = \theta_1 \exp\{\theta_2 x_i/(\theta_3 + x_i)\} + e_i, \quad i = 1, \ldots, n = 14$$

where $e_1, \ldots, e_n \overset{iid}{\sim} N(0, \sigma^2)$.

Since $f(0, \boldsymbol{\theta}) = \theta_1$ we take $\hat{\theta}_1^0 = 4.14$.

Now note that

$$\begin{aligned} \frac{y_i}{\theta_1} &\approx \exp\{\theta_2 x_i/(\theta_3 + x_i)\} \\ \Rightarrow \qquad \log(y_i/\theta_1) &\approx \frac{\theta_2 x_i}{\theta_3 + x_i} = \frac{\theta_2}{1 + \theta_3/x_i} \\ \Rightarrow \qquad \frac{1}{\log(y_i/\theta_1)} &\approx \frac{1}{\theta_2} + \frac{\theta_3}{\theta_2}\frac{1}{x_i} \end{aligned}$$

This suggests that we obtain starting values for θ_2, θ_3 by performing a simple linear regression of $1/\log(y_i/\theta_1)$ on $1/x_i$ for cases 2–14 (we need to exclude $x = 0$).

- See handout Pressure1.

This simple linear regression yields

$$\frac{1}{\hat{\theta}_2^0} = .0553 \qquad \Rightarrow \qquad \hat{\theta}_2^0 = 1/.0553 = 18.07,$$

and

$$\frac{\hat{\theta}_3^0}{\hat{\theta}_2^0} = 13.28 \qquad \Rightarrow \qquad \hat{\theta}_3^0 = 13.28\hat{\theta}_2^0 = 240.03$$

so that $\hat{\boldsymbol{\theta}}^0 = (4.14, 18.1, 240.1)^T$.

- A very convenient feature in the nlme library is the ability to define *self-starting models*.

- We've already seen that it can be convenient to code the expectation function as an R function, and use the deriv() function so that this function returns not only the value of the expectation function, but also the values of its partial derivatives. E.g., m2rabbit() returned a gradient component as well as the function value.

- If we can code the process used to obtain starting values, we can extend this idea so that the function returns its value, its gradient, *and* appropriate starting values all automatically.

- In pressure1.R we create a self starting function SSPresMod(). The first step is to code the function PresMod to return the expectation function of the model and its derivatives. This is done with the deriv() function.

- Next we code a function PresModInit() to automate the process we just went through to obtain starting values. Then SSPresMod is created by combining PresMod() with the initial value function PresModInit() using the selfStart() function. See the R online help for selfStart() and also see §8.1.2 of Pinheiro and Bates (2000) for more information on creating self starting functions.

- Once we have created SSPresMod() we can check that it works by asking it to produce starting values for our model and the observed data. This is done with the getInitial() function and you can see that SSPresMod() does return the values we obtained less automatically.

- We can now fit our nonlinear regression model using SSPresMod() in the nls() function. We no longer have to specify starting values because they are computed automatically. In addition, any new, similarly-behaving data set can be fit with this model without computing a new set of starting values "by hand".

Several self starting functions for commonly used nonlinear regression models (e.g., biexponential, Michaelis-Menten, many more) are included in S-PLUS through the nlme3 library. Appendix C in Pinheiro and Bates (2000, on reserve) contains descriptions of these self starting functions, including details concerning how starting values are obtained and interpretations of the expectation functions of the models. This appendix is a very useful reference.

Example — Pasture Yield vs. Growing Time

Ratkowsky (1983) reports a small data set on pasture regrowth over time following cutting. The data are as follows:

Growing Time	Pasture Yield
9	8.93
14	10.80
21	18.59
28	22.33
42	39.35
57	56.11
63	61.73
70	64.62
79	67.08

We consider fitting a *Gompertz Model* to these data of the form

$$y_i = \theta_1 - e^{\theta_2 - \theta_3 x_i} + e_i, \quad i = 1, \ldots, n = 9,$$

where $y = \log(\text{pasture yield})$, $x =$time, and $e_1, \ldots, e_9$ are i.i.id mean zero each with variance σ^2.

Starting values:

First note that increasing yields over time imply that

$$0 < \frac{\partial f(x, \boldsymbol{\theta})}{\partial x} = \theta_3 e^{\theta_2 - \theta_3 x}, \qquad \Rightarrow \quad \theta_3 > 0.$$

Therefore, as $x \to \infty$, $f(x, \boldsymbol{\theta}) \to \theta_1$, so θ_1 is the upper asymptote. So, we take

$$\hat{\theta}_1^0 = \log(67.08) = 4.21$$

Now to get other starting values, we could transform to linearity. However, for illustrative purposes, we'll do something else.

As $x \to 0$, $f(x, \boldsymbol{\theta}) \to \theta_1 - e^{\theta_2}$. So, plugging in $\hat{\theta}_1^0$ for θ_1 and $\min(y) = \log(8.93) = 2.19$ for $f(0, \boldsymbol{\theta})$ we can solve the following equation for θ_2

$$4.21 - e^{\theta_2} = 2.19 \quad \Rightarrow \quad \theta_2 = \log(4.21 - 2.19) = .70$$

and take $\hat{\theta}_2^0 = .70$.

Note that

$$\begin{aligned} \bar{y} &= \theta_1 - \frac{1}{n}\sum_{i=1}^{n} e^{\theta_2 - \theta_3 x} + \bar{e} \\ &\approx \theta_1 - e^{\theta_2 - \theta_3 \bar{\mathbf{x}}}, \end{aligned}$$

the approximation holding roughly because the plot of y versus x is approximately linear.

Substituting the initial values of for θ_1 and θ_2 and $\bar{y} = 3.42$ and $\bar{\mathbf{x}} = 42.56$ we solve

$$3.42 = 4.21 - e^{.70 - \theta_3(42.56)} \quad \Rightarrow \quad .70 - \theta_3(42.56) = \log(4.21 - 3.42)$$

to get $\hat{\theta}_3^0 = .022$.

- We use these starting values, $\hat{\boldsymbol{\theta}}^0 = (4.21, .70, .022)^T$, to fit the Gompertz model to these data in handout Pasture1.

4. Parameter Transformations:

In nonlinear regression models, choosing a "good parameterization" is a much more significant and difficult problem than in linear regression.

Parameter transformations can be effective means to reduce the parameter-effects nonlinearity of a model. This improves the linear approximation, which increases the accuracy of approximate inference methods and speeds convergence.

We will come back to the issue of decreasing parameter-effects nonlinearity later. For now, we focus on parameter transformations

a. to enforce constraints on parameters; and

b. to speed convergence.

Constrained Parameters:

In most nonlinear models, parameters are restricted to regions that make sense scientifically.

For example, in growth and yield models, asymptote parameters typically must be positive; in exponential models, parameters in the exponent (often with minus signs in front) must be positive.

Often, it is possible to ignore parameter constraints in the fitting process. One simply checks the converged parameter estimates to ensure that they are in their feasible regions. If so, and the estimated model fits the data well, then the NLSE/MLE has been reached and there is no problem.

- It can happen, though, that during the iterations to fit the model, parameter estimates can go "out of range". This may slow or stop the algorithm, or lead to convergence to incorrect solutions.

- A general solution to this is to use constrained optimization (maximization or minimization) rather than an unconstrained method like G-N.

    - E.g., for equality constraints, can use method of Lagrange multipliers. For inequality constraints (more common), fancier, more difficult methods are necessary.

An easier approach is simply to reparameterize so that, under the reparamterization, the constraint is enforced.

- The most common use of this approach is to enforce positivity. If θ must be positive, reparameterize to $\phi = \log(\theta)$. This ensures that throughout the iterations, $\theta = e^{\phi}$ remains positive.

- An interval constraint of the form

$$a \leq \theta \leq b$$

  can be enforced by a logistic transformation of the form

$$\theta = a + \frac{b-a}{1+e^{-\phi}}.$$

- Although less commonly used, one can even use a transformation (see reference given on p.78 of Bates & Watts) to enforce an order constraint of the form

$$a \leq \theta_j \leq \theta_{j+1} \leq \cdots \leq \theta_k \leq b$$

Besides scientific meaningfulness, a parameter constraint can arise to force **identifiability** of a model.

For a nonlinear regression model

$$y_i = f(\mathbf{x}_i; \boldsymbol{\theta}) + e_i,$$

where the e_i's are i.i.d. with $\mathrm{E}(e_i) = 0$, $\mathrm{var}(e_i) = \sigma^2$, the model is said to be *identifiable* if

$$f(\mathbf{x}_i; \boldsymbol{\theta}) = f(\mathbf{x}_i; \tilde{\boldsymbol{\theta}}) \quad \text{implies } \boldsymbol{\theta} = \tilde{\boldsymbol{\theta}}.$$

Recall this was not the case in the unconstrained biexponential model because for $\boldsymbol{\theta} = (1, 2, 3, 4)^T$ and $\tilde{\boldsymbol{\theta}} = (3, 4, 1, 2)^T$,

$$f(x, \boldsymbol{\theta}) = f(x, \tilde{\boldsymbol{\theta}}) = 1e^{-2x} + 3e^{-4x}$$

- This problem can be fixed by requiring

  $$0 \leq \theta_2 \leq \theta_4$$

  which can be done with the transformation

  $$\begin{aligned} \theta_2 &= e^{\phi_2} \\ \theta_4 &= e^{\phi_2}(1 + e^{\phi_4}) \end{aligned}$$

Another example of a nonidentifiable model:

$$f(\mathbf{x}, \boldsymbol{\theta}) = \exp(-\theta_2\theta_3 x_i) + \frac{\theta_1}{\theta_2}\{1 - \exp(-\theta_2\theta_3 x_1)\}x_2.$$

Here, θ_3 only occurs in the model through the product $\theta_2\theta_3$. Therefore, $\boldsymbol{\theta} = (c\theta_1, c\theta_2, \theta_3/c)^T$ gives the same value of $f(\mathbf{x}, \boldsymbol{\theta})$ for any nonzero constant c.

A solution is to reparameterize so that the model has only two (nonredundant) parameters. Under the reparameterization, take

$$\phi_1 = \theta_2\theta_3, \qquad \phi_2 = \theta_1/\theta_2$$

to fix the problem and make the model identifiable.

Facilitating Convergence:

Two simple methods to facilitate convergence are **centering** and **scaling**.

By centering, we mean replacing the explanatory variables $x_{i1}, \ldots, x_{im}$ by their deviations from their respective means, $\bar{x}_1, \ldots, \bar{x}_m$.

For example, Bates and Watts suggest reparameterizing the exponential model

$$f(x, \boldsymbol{\theta}) = \theta_1 e^{-\theta_2 x} \qquad (*)$$

by replacing x by $x - \bar{x}$. That is, (*) can be written as

$$f(x, \boldsymbol{\theta}) = \theta_1 e^{-\theta_2(x-\bar{x}+\bar{x})} = \theta_1 e^{-\theta_2 \bar{\mathbf{x}}} e^{-\theta_2(x-\bar{\mathbf{x}})} = \phi_1 e^{-\phi_2(x-\bar{\mathbf{x}})}$$

where we have reparameterized using

$$\phi_1 = \theta_1 e^{-\theta_2 \bar{\mathbf{x}}}, \qquad \phi_2 = \theta_2.$$

A centering reparameterization such as this can make the columns of the derivative matrix $\mathbf{V}$ less collinear (linearly dependent) than they would be otherwise and thus can stabilize the GN algorithm and facilitate convergence.

It's also often helpful to scale the parameters and the data to ensure good conditioning of the derivative matrix. For example, if the model was

$$f(x, \boldsymbol{\theta}) = \theta_1 e^{-\theta_2 x}$$

with the response in the range $0 < y < 100$ and the regressor x in the range $0 < x < 1000$, and $\theta_1 \approx 100$ while $\theta_2 \approx 0.001$, then it would be prudent to write

$$f(x, \boldsymbol{\theta}) = 100\phi_1 e^{-\phi_2 (x/1000)}.$$

This way both ϕ_1 and ϕ_2 are approximately 1 and the derivatives are of similar magnitude.

- In linear and nonlinear regression, centering and scaling are helpful to reduce the multicollinearity (near linear dependence) among the columns of the derivative matrix ($\mathbf{X}$ in the linear case, $\mathbf{V}$ in the nonlinear case). This **multicollinearity** or **ill conditioning** of the regression problem can be thought of as "approximate non-identifiability."

Ill-Conditioning:

Consider the linear model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{e}, \quad \mathrm{E}(\mathbf{e}) = \mathbf{0}, \mathrm{var}(\mathbf{e}) = \sigma^2 \mathbf{I}.$$

- This model is identifiable (i.e., $\boldsymbol{\beta}$ is estimable) if and only if $\mathbf{X}$ is of full rank. $\mathbf{X}$ is full rank just means that the columns of $\mathbf{X}$ are linearly independent, or, mathematically, that $\mathbf{Xa} = \mathbf{0}$ if and only if $\mathbf{a} = \mathbf{0}$.

If $\mathbf{X}$ is not of full rank (and so has linearly dependent columns), there exists a vector $\mathbf{c} \neq \mathbf{0}$ so that $\mathbf{Xc} = \mathbf{0}$ and hence

$$\|\mathbf{Xc}\|^2 = (\mathbf{Xc})^T(\mathbf{Xc}) = \mathbf{c}^T(\mathbf{X}^T\mathbf{X})\mathbf{c} = 0$$

If $\mathbf{X}$ has nearly linearly dependent columns then there exists a vector $\mathbf{c} \neq \mathbf{0}$ so that

$$\|\mathbf{Xc}\|^2 = \mathbf{c}^T(\mathbf{X}^T\mathbf{X})\mathbf{c} \approx 0$$

I.e., for any $\boldsymbol{\beta}$ there exists a $\tilde{\boldsymbol{\beta}} \neq \boldsymbol{\beta}$ so that $\boldsymbol{\beta} - \tilde{\boldsymbol{\beta}} \neq \mathbf{0}$ and

$$\begin{aligned} \|\mathbf{X}(\boldsymbol{\beta} - \tilde{\boldsymbol{\beta}})\|^2 = (\boldsymbol{\beta} - \tilde{\boldsymbol{\beta}})^T(\mathbf{X}^T\mathbf{X})(\boldsymbol{\beta} - \tilde{\boldsymbol{\beta}}) &\approx 0 \\ \Rightarrow \qquad \|\mathbf{X}\boldsymbol{\beta} - \mathbf{X}\tilde{\boldsymbol{\beta}}\|^2 &\approx 0 \\ \Rightarrow \qquad \mathbf{X}\boldsymbol{\beta} &\approx \mathbf{X}\tilde{\boldsymbol{\beta}} \end{aligned}$$

- The model is identifiable if $\mathbf{X}\boldsymbol{\beta} = \mathbf{X}\tilde{\boldsymbol{\beta}}$ implies $\boldsymbol{\beta} = \tilde{\boldsymbol{\beta}}$. Under multicollinearity, we have just seen that we obtain *approximately* equal expectation functions for distinct values $\boldsymbol{\beta}$ and $\tilde{\boldsymbol{\beta}}$. Hence multicollinearity can be thought of as *approximate nonidentifiability*.
- Two measures of multicollinearity are the determinant of $\mathbf{X}^T\mathbf{X}$ and the minimum eigenvalue of $\mathbf{X}^T\mathbf{X}$. Both of these quantities will be 0 under linear dependence and will be close to 0 under approximate linear dependence (or multicollinearity).
- When this happens $\mathbf{X}^T\mathbf{X}$ will be nearly singular (uninvertible) and we say $\mathbf{X}^T\mathbf{X}$ is *ill-conditioned.*

Consequences:

a. Numerical Instability:

   It's possible to obtain two substantially different estimators $\hat{\boldsymbol{\beta}}_1$ and $\hat{\boldsymbol{\beta}}_2$ that give nearly equal expectation functions: $\mathbf{X}\hat{\boldsymbol{\beta}}_1 \approx \mathbf{X}\hat{\boldsymbol{\beta}}_2$. This means it's hard to find the "right" $\hat{\boldsymbol{\beta}}$.

   This is minimized by using a good algorithm to obtained the LSE (e.g., use the QR decompostion).

b. Large variances for some parameters (or for some linear combinations of the parameters).

   This can be seen from the fact that $\text{var}(\hat{\boldsymbol{\beta}}) = \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}$ and $(\mathbf{X}^T\mathbf{X})^{-1}$ is equal to $1/\det(\mathbf{X}^T\mathbf{X})$ times some matrix. If $\det(\mathbf{X}^T\mathbf{X}) \approx 0$ then the elements of this var-cov matrix will blow up.

- In linear regression, the cure for multicollinearity is to eliminate one or more of the explanatory variables.

In nonlinear regression, the matrix $\mathbf{V}(\boldsymbol{\theta})$ plays the role of $\mathbf{X}$. If the columns of $\mathbf{V}(\boldsymbol{\theta})$ for $\boldsymbol{\theta}$ near $\hat{\boldsymbol{\theta}}$ are approximately linearly dependent, then we may have

a. Numerical instability;

b. Large variances; and

c. Slow or non-convergence.

- In the nonlinear case, this ill-conditioning is difficult to correct because it could be due to

   i. the data (as in the linear case); and/or

   ii. the model.

Example — Corn Yield

The following table contains data on R, the mean dry kernel weight (4 plants) of corn, at various levels of x, the time since silking.

Time Since Silking	Mean Kernel Weight
17.125	14.26
25.625	50.51
29.625	60.83
39.625	104.78
46.375	94.46
54.250	97.02
62.125	172.41

For these data we consider the model

$$y_i = \theta_1 - \theta_4 \log(1 + e^{\theta_2 - \theta_3 x_i}) + e_i, \quad i = 1, \ldots, n = 7$$

where $y_i = \log(R_i)$ and we make the usual error assumptions.

- This model is a four-parameter form of the *Richards Model* with multiplicative error.
- See the handout corn1. Here, we attempt to fit the above model. Methods for obtaining starting values for Richards models are discussed in Ratkowsky (1983, §8.3.3) and Seber & Wild (1989, §7.3.6), but we ignore that issue in this example.
- In corn1, we use the gnls() function rather than the nls() function, simply because it allows greater control over the fitting algorithm and because the returnObject option allows the function to return the "fitted model" at the end of the fitting algorithm, even if the algorithm has not converged. The model we are fitting is an OLS-type model, though, with homoscedastic, independent errors as usual.

- Notice that the algorithm did not converge. We could reduce the convergence criteria in this model further and we'd see that the paramter estimates would continue to jump around without settling down.

- Notice that at the point at which the algorithm stopped, the correlations between the parameter estimates were very high in magnitude. In particular, $\hat{\text{corr}}(\hat{\theta}_2, \hat{\theta}_4) = -1.000$ (to three decimal places). Large magnitude correlations like this are indicators of ill-conditioning and often suggest that the model is *overparameterized*.

    - As a rule of thumb, correlations $> .99$ in absolute value are strong indicators of overparameterization.

- In corn1 we also compute the determinant and eigenvalues of $\{\mathbf{V}(\hat{\boldsymbol{\theta}})\}^T\mathbf{V}(\hat{\boldsymbol{\theta}})$ taking $\hat{\boldsymbol{\theta}}$ to be the value of $\boldsymbol{\theta}$ when the G-N algorithm stopped. The determinant is 4.65e-6 and the minimum eigenvalue is 2.61e-9, both tiny, indicating ill-conditioning.

- The solution here is to simplify the model. A four-parameter model is overkill here. Four parameters are sometimes necessary in growth curve models like the Richards model above to establish the shape of the entire curve from beginning to end of the growth process. In our data, we're not seeing the convex part of the sigmoidal growth curve, and we're certainly not seeing all of the "tail" behavior, so we really don't have data appropriate to estimate parameters related to the point of inflection and the asymptotes. Doing so is essentially extrapolating from the data.

- A three-parameter version of this model arises by setting $\theta_4 = 1$. The resulting model is a reparameterization of the 3-parameter logistic regression model. We succesfully fit that model in corn1 and it appears to fit the data well.

Q: Why do high correlations among the $\hat{\theta}_j$'s indicate ill-conditioning?

A: Consider the linear case. There, it is not hard to show $\text{corr}(\hat{\beta}_j, \hat{\beta}_k)$ is equal to -1 times the partial correlation between $\mathbf{x}_j$ and $\mathbf{x}_k$, the j^{th} and k^{th} explanatory variables, respectively.

- Recall that the partial correlation between $\mathbf{x}_j$ and $\mathbf{x}_k$ measures the linear association between $\mathbf{x}_j$ and $\mathbf{x}_k$ after removing the effects of all other explanatory variables.

- Therefore, $|\hat{\text{corr}}(\hat{\beta}_j, \hat{\beta}_k)| \approx 1$ means that $\mathbf{x}_j$ and $\mathbf{x}_k$ have almost perfect (positive or negative) partial correlation (i.e., they are collinear).

- The situation is somewhat more complex in the nonlinear case, but the result is the same: large magnitude correlations among the $\hat{\theta}_j$'s indicate collinearity among the columns of $\mathbf{V}(\boldsymbol{\theta})$ and ill-conditioning of the $\{\mathbf{V}(\boldsymbol{\theta})\}^T\mathbf{V}(\boldsymbol{\theta})$ matrix.

5. Alternative Fitting Algorithms:

There are several alternatives to the GN algorithm for finding $\hat{\boldsymbol{\theta}}$, the value of $\boldsymbol{\theta}$ that minimizes

$$S(\boldsymbol{\theta}) = \|\mathbf{y} - \boldsymbol{\eta}(\boldsymbol{\theta})\|^2.$$

A. Newton-Raphson Algorithm.

$\hat{\boldsymbol{\theta}}$ satisfies the normal equation

$$\frac{\partial S}{\partial \boldsymbol{\theta}}(\hat{\boldsymbol{\theta}}) = \mathbf{0}.$$

where $\frac{\partial S}{\partial \boldsymbol{\theta}}(\hat{\boldsymbol{\theta}}) \equiv \frac{\partial S(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}}$.

Using a linear Taylor series approximation of $\frac{\partial S}{\partial \boldsymbol{\theta}}(\hat{\boldsymbol{\theta}})$ we have

$$\mathbf{0} = \frac{\partial S}{\partial \boldsymbol{\theta}}(\hat{\boldsymbol{\theta}}) \approx \frac{\partial S}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}) + \frac{\partial^2 S}{\partial \boldsymbol{\theta}\partial \boldsymbol{\theta}^T}(\boldsymbol{\theta})(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta})$$

for $\boldsymbol{\theta}$ close to $\hat{\boldsymbol{\theta}}$.

Rearranging,

$$\hat{\boldsymbol{\theta}} - \boldsymbol{\theta} \approx -\underbrace{\left\{\frac{\partial^2 S}{\partial \boldsymbol{\theta}\partial \boldsymbol{\theta}^T}(\boldsymbol{\theta})\right\}^{-1}}_{p\times p}\underbrace{\frac{\partial S}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta})}_{p\times 1}.$$

This approximation suggests the Newton-Raphson updating formula: Given a starting value $\hat{\boldsymbol{\theta}}^0$, we update via

$$\hat{\boldsymbol{\theta}}^j = \hat{\boldsymbol{\theta}}^{j-1} - \left\{\frac{\partial^2 S}{\partial \boldsymbol{\theta}\partial \boldsymbol{\theta}^T}(\hat{\boldsymbol{\theta}}^{j-1})\right\}^{-1} \frac{\partial S}{\partial \boldsymbol{\theta}}(\hat{\boldsymbol{\theta}}^{j-1}), \quad j = 1, 2, \ldots, \text{convergence},$$

to obtain $\hat{\boldsymbol{\theta}}$.

- This approach is valid for any minimization/maximization problem.

In nonlinear regression,

$$S(\boldsymbol{\theta}) = \{\mathbf{y} - \boldsymbol{\eta}(\boldsymbol{\theta})\}^T\{\mathbf{y} - \boldsymbol{\eta}(\boldsymbol{\theta})\} = \sum_{i=1}^{n}\{y_i - f(\mathbf{x}_i, \boldsymbol{\theta})\}^2$$

so

$$\begin{aligned}
\frac{\partial S}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}) &= -2\sum_{i=1}^{n}\{y_i - \eta_i(\boldsymbol{\theta})\}\frac{\partial f(\mathbf{x}_i, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \\
&= -2\begin{pmatrix} \frac{\partial f(\mathbf{x}_1, \boldsymbol{\theta})}{\partial \theta_1} & \cdots & \frac{\partial f(\mathbf{x}_n, \boldsymbol{\theta})}{\partial \theta_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial f(\mathbf{x}_1, \boldsymbol{\theta})}{\partial \theta_p} & \cdots & \frac{\partial f(\mathbf{x}_n, \boldsymbol{\theta})}{\partial \theta_p} \end{pmatrix}\begin{pmatrix} y_1 - f(\mathbf{x}_1, \boldsymbol{\theta}) \\ \vdots \\ y_n - f(\mathbf{x}_n, \boldsymbol{\theta}) \end{pmatrix} \\
&= -2\{\mathbf{V}(\boldsymbol{\theta})\}^T\{\mathbf{y} - \boldsymbol{\eta}(\boldsymbol{\theta})\}
\end{aligned}$$

In addition, the second derivative matrix has $(j,k)^{\text{th}}$ element

$$\begin{aligned}
\frac{\partial^2 S(\boldsymbol{\theta})}{\partial\theta_j\partial\theta_k} &= \frac{\partial}{\partial\theta_k}\left(\frac{\partial S(\boldsymbol{\theta})}{\partial\theta_j}\right) \\
&= \frac{\partial}{\partial\theta_k}\left(-2\sum_i\{y_i - f(\mathbf{x}_i, \boldsymbol{\theta})\}\frac{\partial f(\mathbf{x}_i, \boldsymbol{\theta})}{\partial\theta_j}\right) \\
&= 2\sum_{i=1}^{n}\frac{\partial f(\mathbf{x}_i, \boldsymbol{\theta})}{\partial\theta_k}\frac{\partial f(\mathbf{x}_i, \boldsymbol{\theta})}{\partial\theta_j} - 2\sum_{i=1}^{n}\{y_i - f(\mathbf{x}_i, \boldsymbol{\theta})\}\frac{\partial^2 f(\mathbf{x}_i, \boldsymbol{\theta})}{\partial\theta_j\partial\theta_k}
\end{aligned}$$

or, in matrix notation, the entire second derivative matrix is

$$\frac{\partial S(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}\partial\boldsymbol{\theta}^T} = 2\{\mathbf{V}(\boldsymbol{\theta})\}^T\{\mathbf{V}(\boldsymbol{\theta})\} - 2\overbrace{\underbrace{\frac{\partial\{\mathbf{V}(\boldsymbol{\theta})\}^T}{\partial\boldsymbol{\theta}}}_{\text{a 3-dim array of derivatives}}\{\mathbf{y} - \boldsymbol{\eta}(\boldsymbol{\theta})\}}^{\equiv\mathbf{D}}$$

So in the context of nonlinear least squares regression, the Newton-Raphson update is given by

$$\hat{\boldsymbol{\theta}}^j = \hat{\boldsymbol{\theta}}^{j-1} + \left(\{\mathbf{V}(\hat{\boldsymbol{\theta}}^{j-1})\}^T \{\mathbf{V}(\hat{\boldsymbol{\theta}}^{j-1})\} - \mathbf{D} \right)^{-1} \{\mathbf{V}(\hat{\boldsymbol{\theta}}^{j-1})\}^T \{\mathbf{y} - \boldsymbol{\eta}(\hat{\boldsymbol{\theta}}^{j-1})\}$$

Recall that the GN update was

$$\hat{\boldsymbol{\theta}}^j = \hat{\boldsymbol{\theta}}^{j-1} + \left(\{\mathbf{V}(\hat{\boldsymbol{\theta}}^{j-1})\}^T \{\mathbf{V}(\hat{\boldsymbol{\theta}}^{j-1})\} \right)^{-1} \{\mathbf{V}(\hat{\boldsymbol{\theta}}^{j-1})\}^T \{\mathbf{y} - \boldsymbol{\eta}(\hat{\boldsymbol{\theta}}^{j-1})\}$$

- These algorithms are obviously very similar. As compared with NR, the GN algorithm just replaces

  $$\frac{\partial S(\boldsymbol{\theta})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T} = 2\{\mathbf{V}(\boldsymbol{\theta})\}^T \{\mathbf{V}(\boldsymbol{\theta})\} - 2\mathbf{D}$$

  by its expected value, $2\{\mathbf{V}(\boldsymbol{\theta})\}^T \{\mathbf{V}(\boldsymbol{\theta})\}$. That is, GN ignores the term $\mathbf{D}$, which has mean $\mathbf{0}$, in the NR update formula above.

- In a more general context, the GN modification of the NR algorithm is often called *Newton-Raphson with Fisher scoring,* or sometimes simply, *the Fisher scoring algorithm.*

- Advantage of GN: only requires first derivatives of the expectation function.

- Advantage of NR: sometimes has better convergence properties than GN.

- Often, the two algorithms will perform about the same.

B. Quasi-Newton Methods.

- The main drawback to the N-R method is having to compute the second derivatives. QN methods avoid this by using a numerical approximation to the second derivative matrix.*

- This approximation starts out crudely, but is updated (improved) after each step of the algorithm to get progressively more accurate.

Let

$$\mathbf{H}(\boldsymbol{\theta}) = \frac{\partial S}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T}(\boldsymbol{\theta}) \quad \text{and} \quad g(\boldsymbol{\theta}) = \frac{\partial S}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta})$$

('H' for Hessian, 'g' for gradient).

In the NR algorithm we had the updating formula

$$\hat{\boldsymbol{\theta}}^{j+1} = \hat{\boldsymbol{\theta}}^j - \{\mathbf{H}(\hat{\boldsymbol{\theta}}^j)\}^{-1} g(\hat{\boldsymbol{\theta}}^j).$$

In the QN algorithm we want to avoid computing second derivatives, so we replace $\{\mathbf{H}(\hat{\boldsymbol{\theta}}^j)\}^{-1}$ by an approximating matrix $\mathbf{B}^j$ and use

$$\hat{\boldsymbol{\theta}}^{j+1} = \hat{\boldsymbol{\theta}}^j - \mathbf{B}^j g(\hat{\boldsymbol{\theta}}^j). \qquad (\heartsuit)$$

- At step 0, $\mathbf{B}^0$ is often taken to be something quite simple and crude, such as $\mathbf{B}^0 = \mathbf{I}$ and then improved at each step of the algorithm.

  How do we update the approximate inverse Hessian $\mathbf{B}^j$?

* (or, to be more accurate, its inverse, which is really what is needed)

Note that a first-order Taylor approximation of the gradient vector gives

$$g(\hat{\boldsymbol{\theta}}^{j+1}) \approx g(\hat{\boldsymbol{\theta}}^{j}) + \mathbf{H}(\hat{\boldsymbol{\theta}}^{j})(\hat{\boldsymbol{\theta}}^{j+1} - \hat{\boldsymbol{\theta}}^{j})$$

for $\hat{\boldsymbol{\theta}}^{j+1}$ close to $\hat{\boldsymbol{\theta}}^{j}$.

Rearranging, we have

$$\{\mathbf{H}(\hat{\boldsymbol{\theta}}^{j})\}^{-1}\{g(\hat{\boldsymbol{\theta}}^{j+1}) - g(\hat{\boldsymbol{\theta}}^{j})\} \approx \underbrace{(\hat{\boldsymbol{\theta}}^{j+1} - \hat{\boldsymbol{\theta}}^{j})}_{\text{depends on } \mathbf{B}^{j}}$$

or

$$\{\mathbf{H}(\hat{\boldsymbol{\theta}}^{j})\}^{-1}\mathbf{q}^{j} \approx \mathbf{p}^{j}, \qquad (\spadesuit)$$

where $\mathbf{q}^{j} = g(\hat{\boldsymbol{\theta}}^{j+1}) - g(\hat{\boldsymbol{\theta}}^{j})$ and $\mathbf{p}^{j} = \hat{\boldsymbol{\theta}}^{j+1} - \hat{\boldsymbol{\theta}}^{j}$.

- Note that ($\spadesuit$) holds exactly if $S(\boldsymbol{\theta})$ (the objective function) is quadratic in $\boldsymbol{\theta}$ (e.g., in a linear model).

The Quasi-Newton method uses ($\spadesuit$) to obtain an updated estimate of the inverse Hessian of the form

$$\mathbf{B}^{j+1} = \mathbf{B}^{j} + \mathbf{E}^{j} \qquad (\diamondsuit)$$

- Here, $\mathbf{E}^{j}$ updates $\mathbf{B}^{j}$ to give $\mathbf{B}^{j+1}$ which is used as an estimate of $\{\mathbf{H}(\hat{\boldsymbol{\theta}}^{j+1})\}^{-1}$ for the next step of the algorithm .

Turning ($\spadesuit$) into an equality and substituting ($\diamondsuit$), we get

$$(\mathbf{B}^{j} + \mathbf{E}^{j})\mathbf{q}^{j} = \mathbf{p}^{j} \qquad (1)$$

Equation (1) now gives a means for choosing $\mathbf{E}^j$, the update matrix to improve our estimate of the inverse Hessian. There is no unique $\mathbf{E}^j$ satisfying (1), but typically, $\mathbf{E}^j$ is taken to be of the form

$$\mathbf{E}^j = a\mathbf{u}\mathbf{u}^T + b\mathbf{v}\mathbf{v}^T$$

where $a\mathbf{u}\mathbf{u}^T$ and $b\mathbf{v}\mathbf{v}^T$ are symmetric matrices each of rank one chosen so that (1) is satisfied.

- QN methods with $b = 0$ are called rank one update methods, QN methods with $b \neq 0$ are known as rank two methods.

- Rank two methods are the most widely used, with the Davidon-Fletcher-Powell (DFP) and, especially, the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update methods most common in modern implementations of the QN method.

The BFGS update takes the form:

$$\mathbf{E}^j = \left(1 + \frac{\mathbf{q}^{jT}\mathbf{B}^j\mathbf{q}^j}{\mathbf{p}^{jT}\mathbf{q}^j}\right)\frac{\mathbf{p}^j\mathbf{p}^{jT}}{\mathbf{p}^{jT}\mathbf{q}^j} - \frac{\mathbf{p}^j\mathbf{q}^{jT}\mathbf{B}^j + \mathbf{B}^j\mathbf{q}^j\mathbf{p}^{jT}}{\mathbf{p}^{jT}\mathbf{q}^j}.$$

- As in the GN method, QN methods usually make use of a step size λ. That is, instead of ($\heartsuit$), we update via

  $$\hat{\boldsymbol{\theta}}^{j+1} = \hat{\boldsymbol{\theta}}^j - \lambda\mathbf{B}^j g(\hat{\boldsymbol{\theta}}^j).$$

  where λ is chosen that ensures a decrease in the objective function or, alternatively, a decrease in the length of the gradient vector.

- Typically, a *line search* is done to obtain a near optimal value of λ, not just one that works. E.g., instead of just picking a λ value that decreases the objective function, we attempt to find the λ value that produces the greatest decrease in the objective function.

Comparisons with GN, NR:

  - Speed of convergence: (slowest) GN≤QN≤NR (fastest).
  - Difficulty of use: (easiest) GN=QN≤NR (hardest).

- Convergence of GN, QN approaches that of NR when $S(\boldsymbol{\theta})$ is approximately quadratic in $\boldsymbol{\theta}$.
- QN appropriate for a problem that is not highly nonlinear but where second derivatives are tedious to calculate.

C. Steepest Descent.

Iterative methods for calculating $\hat{\boldsymbol{\theta}}$ have the general form:

$$\hat{\boldsymbol{\theta}}^{j+1} = \hat{\boldsymbol{\theta}}^{j} + \boldsymbol{\delta}^{j},$$

where the increment $\boldsymbol{\delta}^{j}$ usually has 2 components:

i. the direction of the step $\mathbf{d}^{j}$; and

ii. the length of the step, λ^{j};

where $\boldsymbol{\delta}^{j} = \lambda^{j}\mathbf{d}^{j}$.

- In GN algorithm, we chose λ^{j} so that

$$S(\hat{\boldsymbol{\theta}}^{j+1}) < S(\hat{\boldsymbol{\theta}}^{j}).$$

$\mathbf{d}^{j}$ is a descent direction if

$$\frac{\partial}{\partial\lambda} S(\hat{\boldsymbol{\theta}}^{j} + \lambda\mathbf{d}^{j})\big|_{\lambda=0} < 0.$$

If $\mathbf{d}^{j}$ is a descent direction, then for small enough λ^{j}, the sum of squares will decrease. I.e.,

$$S(\hat{\boldsymbol{\theta}}^{j} + \lambda^{j}\mathbf{d}^{j}) < S(\hat{\boldsymbol{\theta}}^{j}).$$

Note that

$$\frac{\partial}{\partial\lambda} S(\hat{\boldsymbol{\theta}}^{j} + \lambda\mathbf{d}^{j})\big|_{\lambda=0} = \left\{ S'(\hat{\boldsymbol{\theta}}^{j} + \lambda\mathbf{d}^{j})\big|_{\lambda=0} \right\}^{T} \mathbf{d}^{j} = \underbrace{\left\{ \frac{\partial}{\partial\boldsymbol{\theta}} S(\hat{\boldsymbol{\theta}}^{j}) \right\}^{T} \mathbf{d}^{j}}_{=(\clubsuit)}$$

- The method of steepest descent says choose $\mathbf{d}^{j}$ to minimize ($\clubsuit$) (make it as negative as possible).

Equivalently, we can minimize

$$\frac{1}{\|\mathbf{d}^j\|}\left\{\frac{\partial}{\partial\boldsymbol{\theta}}S(\hat{\boldsymbol{\theta}}^j)\right\}^T\mathbf{d}^j$$

over all vectors $\mathbf{d}^j$.

By the Cauchy-Schwartz Inequality ($|\mathbf{x}^T\mathbf{y}| \leq \|\mathbf{x}\|\|\mathbf{y}\|$ with equality iff $\mathbf{x} = c\mathbf{y}$) we have

$$\left|\underbrace{\left\{\frac{\partial}{\partial\boldsymbol{\theta}}S(\hat{\boldsymbol{\theta}}^j)\right\}^T\mathbf{d}^j}_{<0}\right| \leq \|\mathbf{d}^j\|\|\frac{\partial}{\partial\boldsymbol{\theta}}S(\hat{\boldsymbol{\theta}}^j)\|.$$

Therefore, we have

$$\frac{\left\{\frac{\partial}{\partial\boldsymbol{\theta}}S(\hat{\boldsymbol{\theta}}^j)\right\}^T\mathbf{d}^j}{\|\mathbf{d}^j\|} \geq \underbrace{-\|\frac{\partial}{\partial\boldsymbol{\theta}}S(\hat{\boldsymbol{\theta}}^j)\|}_{\text{lower bound}}$$

with equality if and only if $\mathbf{d}^j$ is a multiple of $-\frac{\partial}{\partial\boldsymbol{\theta}}S(\hat{\boldsymbol{\theta}}^j)$.

Thus, the steepest descent iteration becomes

$$\hat{\boldsymbol{\theta}}^{j+1} = \hat{\boldsymbol{\theta}}^j - \lambda\frac{\partial}{\partial\boldsymbol{\theta}}S(\hat{\boldsymbol{\theta}}^j) \qquad (*)$$

where λ is chosen to minimize $S(\hat{\boldsymbol{\theta}}^{j+1})$.

Note that in (*), $\frac{\partial}{\partial \boldsymbol{\theta}} S(\hat{\boldsymbol{\theta}}^j)$ is just the gradient of $S(\boldsymbol{\theta})$ evaluated at $\hat{\boldsymbol{\theta}}^j$.

Thus, we have the following updates:

$$\hat{\boldsymbol{\theta}}^{j+1} = \hat{\boldsymbol{\theta}}^j - \lambda[\mathrm{E}\{\mathbf{H}(\hat{\boldsymbol{\theta}}^j)\}]^{-1} g(\hat{\boldsymbol{\theta}}^j) \qquad (GN)$$

$$\hat{\boldsymbol{\theta}}^{j+1} = \hat{\boldsymbol{\theta}}^j - \lambda\{\mathbf{H}(\hat{\boldsymbol{\theta}}^j)\}^{-1} g(\hat{\boldsymbol{\theta}}^j) \qquad (NR)$$

$$\hat{\boldsymbol{\theta}}^{j+1} = \hat{\boldsymbol{\theta}}^j - \lambda \mathbf{B}^j g(\hat{\boldsymbol{\theta}}^j) \qquad (QN)$$

$$\hat{\boldsymbol{\theta}}^{j+1} = \hat{\boldsymbol{\theta}}^j - \lambda \mathbf{I} g(\hat{\boldsymbol{\theta}}^j) \qquad (SD)$$

- Advantages to Steepest Descent: Relatively insensitive to the starting values. May converge when GN and NR do not.
- Disadvantage: Can be (very) slow relative to other methods.

D. Levenberg-Marquardt Method.

If $\{\mathbf{V}(\boldsymbol{\theta})\}^T\{\mathbf{V}(\boldsymbol{\theta})\}$ is close to singular for $\boldsymbol{\theta}$ near $\hat{\boldsymbol{\theta}}$, the GN method may become erratic. This problem can be diminished by a good algorithm for obtaining the GN increment, but not completely solved.

One helpful approach is to modify the GN increment to

$$\boldsymbol{\delta}^j(k) \equiv [\{\mathbf{V}(\hat{\boldsymbol{\theta}}^j)\}^T\{\mathbf{V}(\hat{\boldsymbol{\theta}}^j)\} + k\mathbf{B}]^{-1}\{\mathbf{V}(\hat{\boldsymbol{\theta}}^j)\}^T\{\mathbf{y} - \boldsymbol{\eta}(\hat{\boldsymbol{\theta}}^j)\}$$

where $\mathbf{B}$ is a diagonal matrix and k is a constant called the *conditioning factor.*

Two standard choices of $\mathbf{B}$ are

i. $\mathbf{B} = \mathbf{I}$ (Levenberg); and

ii. $\mathbf{B}$ a diagonal matrix with diagonal elements taken from the diagonal of $\{\mathbf{V}(\hat{\boldsymbol{\theta}}^j)\}^T\{\mathbf{V}(\hat{\boldsymbol{\theta}}^j)\}$ (Marquardt).

- When $k = 0$ this method (either choice of $\mathbf{B}$) reduces to the G-N method.
- For $\mathbf{B} = \mathbf{I}$ and $k \to \infty$ this method becomes steepest descent.
- Therefore, the LM method is a compromise between G-N and steepest descent.

To choose k, the following method is often used:

i. Start with small k.

ii. If $S(\hat{\boldsymbol{\theta}}^0 + \boldsymbol{\delta}^0(k)) < S(\hat{\boldsymbol{\theta}}^0)$, then decrease k on next iteration.

iii. If $S(\hat{\boldsymbol{\theta}}^0 + \boldsymbol{\delta}^0(k)) \geq S(\hat{\boldsymbol{\theta}}^0)$, then increase k until (ii) occurs.

- For well-behaved problems, this approach will lead to $k \to 0$ and the method will behave like GN.
- For ill-conditioned problems, k will increase toward ∞ and the method will become a variant of steepest descent.

- Advantage: LM method is useful when $\{\mathbf{V}(\hat{\boldsymbol{\theta}}^j)\}^T\{\mathbf{V}(\hat{\boldsymbol{\theta}}^j)\}$ is ill-conditioned.

- Disadvantage: None, really. This is a good default method.

E. Golub-Pereyra Algorithm for Partially Linear Models.

In partially linear models, the conditionally linear parameters can be estimated by linear least-squares, conditional on the values of the nonlinear parameters.

That is, if we partition $\boldsymbol{\theta}$ as $\boldsymbol{\theta} = (\boldsymbol{\beta}^T, \boldsymbol{\phi}^T)^T$ where $\boldsymbol{\beta}$ $(p_1 \times 1)$ contains the conditionally linear elements of $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ $((p - p_1) \times 1)$ the remaining elements of $\boldsymbol{\theta}$, then we can write

$$\boldsymbol{\eta}(\boldsymbol{\theta}) = \boldsymbol{\eta}(\boldsymbol{\beta}, \boldsymbol{\phi}) = \mathbf{A}(\boldsymbol{\phi})\boldsymbol{\beta}$$

where $\mathbf{A}(\boldsymbol{\phi})$ is an $n \times p_1$ matrix depending only on the nonlinear parameters.

For fixed $\boldsymbol{\phi}$, we can estimate $\boldsymbol{\beta}$ as

$$\underbrace{\hat{\boldsymbol{\beta}}(\boldsymbol{\phi})}_{\text{depends on } \boldsymbol{\phi}} = [\{\mathbf{A}(\phi)\}^T\{\mathbf{A}(\phi)\}]^{-1}\{\mathbf{A}(\phi)\}^T\mathbf{y}$$

Golub and Pereyra formulated a version of the GN algorithm to minimize the least-squares criterion expressed solely as a function of $\boldsymbol{\phi}$:

$$\mathbf{S}_2(\boldsymbol{\phi}) = \|\mathbf{y} - \mathbf{A}(\boldsymbol{\phi})\hat{\boldsymbol{\beta}}(\boldsymbol{\phi})\|^2.$$

The nonlinear least squares estimator of $\boldsymbol{\theta}$ is then $\hat{\boldsymbol{\theta}} = (\hat{\boldsymbol{\phi}}^T, \hat{\boldsymbol{\beta}}(\hat{\boldsymbol{\phi}})^T)^T$, where $\hat{\boldsymbol{\phi}}$ is the minimizer of $\mathbf{S}_2(\boldsymbol{\phi})$.

- Advantages: Reduces the number of starting values needed. Reduces the dimension of the nonlinear estimation.

- Disadvantages: Requires derivatives $\frac{\partial \mathbf{A}(\boldsymbol{\phi})}{\partial \boldsymbol{\phi}}$ which can be complicated. Only applies to partially linear models so is not a competitor to the other methods in general.

Example — Hahn Data:

To illustrate the Golub-Pereyra algorithm, we consider data from the National Institute of Standards and Technology (NIST) pertaining to a study of the thermal expansion of copper. The data set contains the following variables:

$$y = \text{coefficient of thermal expansion}$$
$$x = \text{temperature (Kelvin)}$$

and we consider the model

$$y_i = \frac{\theta_1 + \theta_2 x_i + \theta_3 x_i^2 + \theta_4 x_i^3}{1 + \theta_5 x_i + \theta_6 x_i^2 + \theta_7 x_i^3} + e_i.$$

- See handout Hahn1. The data and fitted curve are displayed on the last page of this handout.

- Starting values for these data were provided by NIST. We use these starting values along with the usual G-N algorithm to fit the model as m1Hahn.nls. The trace of this model fitting procedure reveals that the G-N algorithm took 10 steps to converge, with objective function $S(\hat{\boldsymbol{\theta}}^0) = 2,275,330$ at step 0.

- Alternatively, we can use the Golub-Pereyra algorithm to fit this model by specifying the option, algorithm=plinear, in the call to nls(). We also must change the model specification so that the right-hand side is the derivative matrix of the linear parameters.

- Notice that when using the plinear algorithm in m2Hahn.nls, we need only specify starting values for the nonlinear parameters (3 of the 7). In fact, we can even get away with specifying the starting values of 0 for these parameters and still achieve convergence to the correct solution (see m3Hahn.nls).

- In addition, the plinear algorithm converges after only 7 iterations. Furthermore, the objective function is $S_2(\hat{\boldsymbol{\phi}}^0) = 52.38$ at initialization. This smaller objective function speeds and stabilizes the algorithm.

Implementations of Various Fitting Algorithms:

S-PLUS/R: nls()

- GN (default)
- Golub-Pereyra (plinear)

SAS: PROC NLIN:

- GN (METHOD=GAUSS, the default)
- NR (METHOD=NEWTON)
- LM (METHOD=MARQUARDT)
- Steepest descent (METHOD=GRADIENT)

SAS: PROC NLMIXED:

- QN (TECHNIQUE=QUANEW, the default). Can control the line search and update methods separately.
- NR w/o line search (TECHNIQUE=NEWRAP). Also uses "ridging", which is essentially the technique of the LM method applied to the Hessian.
- NR w/ line search (TECHNIQUE=NRRIDG) Also uses "ridging".
- Several other optimization methods (trust region, double dogleg, conjugate gradient, Nelder-Mead simplex).

In addition, there are a variety of equation solvers and function optimizers available that are generic; that is, they are not designed specifically for nonlinear regression or even statistical estimation problems, but are general tools. Nevertheless, they can be quite useful.

- In R, the function optim() will do general function optimization via either QN, Nelder-Mead, or conjugate-gradient methods. Also available for S-PLUS in the MASS library.

- In S-PLUS, the function nlmin() implements the QN method using numerical derivatives for the gradient, and nlminb() uses trust-region optimation.

- Matlab has an optimization toolbox with several different functions including fsolve (equation solving), fminunc (unconstrained minimization), and fmincon (constrained minimization). fminunc does unconstrained optimization via the QN method.

6. Obtaining Convergence

When using any iterative method, one of the following occurs:

1. Convergence to "reasonable" parameter estimates.

2. Convergence to "unreasonable" parameter estimates or divergence to extreme values (e.g., values toward $\pm\infty$).

3. Failure of the algorithm to converge.

Under outcome 2, divergence to extremes is typically easy to identify as an incorrect solution. Convergence to a local minimum of the objective function can be a bit harder to detect. These problems are typically due to

   a. incorrectly entered data; or

   b. incorrectly entered, or otherwise incorrect, starting values.

Suppose we converge to a parameter estimate $\hat{\boldsymbol{\theta}}$. Is it a reasonable value for $\boldsymbol{\theta}$?

To answer this:

1. Use our knowledge of the subject matter/science of the problem.

2. Plot the estimated (fitted) expectation function $f(\mathbf{x}; \hat{\boldsymbol{\theta}})$ along with the observed data.

   - If $\mathbf{x} = x$ is one-dimensional, then plot $f(x, \hat{\boldsymbol{\theta}})$ as a function of x.

   - If $\mathbf{x}$ is multi-dimensional, then plot the fitted values $\hat{y} = f(\mathbf{x}; \hat{\boldsymbol{\theta}})$ vs. each component of $\mathbf{x}$ along with the observed data.

Example — Pressure vs. Temperature in Saturated Steam

- See pressure1.sas. Recall we considered a model of the form

$$\text{pres}_i = \theta_1 \exp\{\theta_2 \text{temp}_i/(\theta_3 + \text{temp}_i)\} + e_i$$

- In pressure1.sas we fit that model again using "bad" starting values: $\hat{\boldsymbol{\theta}}^0 = (.1, .1, .1)^T$. Although these starting values lead to divergence in nls(), in PROC NLIN we converge to $\hat{\boldsymbol{\theta}} = (13.89, 3.88e10, 9.65e11)^T$. Recall that using "good" starting values, the parameter estimate was $\hat{\boldsymbol{\theta}} = (5.27, 19.72, 295.00)^T$.

- By plotting the fitted curve under each parameter estimate, we see that the latter estimate (from the "good" starting values) fits the data best. Often, spurious converged parameter estimates will give much worse fits than the correct parameter estimates.

- Here, the best criterion to choose between the parameter estimates is scientific reasonableness, but fit also plays a role.

- If the algorithm converges to an "unreasonable" estimate, the most likely reason is bad starting values.

- It is a good idea to plot $f(x, \hat{\boldsymbol{\theta}}^0)$, the curve generated by the starting values, to judge the quality of those starting values.

Outcome 3, failure to converge can be a much more stubborn problem. Again, it may be due to any of several problems:

a. Bad, or incorrectly specified starting values.

b. the expectation function and/or derivatives have been incorrectly coded.

c. $\{\mathbf{V}(\boldsymbol{\theta})\}^T\mathbf{V}(\boldsymbol{\theta})$ is singular or nearly singular for $\boldsymbol{\theta}$ near $\hat{\boldsymbol{\theta}}$; or

- If (a) starting values are good, and (b) the model and its derivatives are correct, then it can be shown that (c) $\{\mathbf{V}(\boldsymbol{\theta})\}^T\mathbf{V}(\boldsymbol{\theta})$ *must be* singular, or nearly so.

§3.6, p.86 of Bates & Watts provides a useful list of questions to ask/things to check when convergence is a problem:

- Is the expectation function correctly specified and coded?
- Are the derivatives correctly specified and coded?
- Are the data entered correctly and are they "reasonable"?
- Are the response and explanatory variables correctly identified?
- Do the starting values correctly identified with the corresponding parameters?

If the answer to all of these questions is yes, then tracing the details of the fitting algorithm can help diagnose the problem (e.g., use the trace=T or verbose=T options in nls() and gnls(), respectively).

Other points to consider:

- Are the data rich enough to fit the model that you have specified? The model may have parameters that describe features of the expectation function that the data don't reveal (e.g., an asymptote parameter when no data are collected in the asymptotic region of the curve). In this case, simplification of the model may be necessary.

- If the estimation algorithm sends parameter values into infeasible regions, then reparameterization of the model may help.

- For large problems, it is often difficult to fit the final model on the first try, but success may be achieved by building up gradually from smaller, simpler models to the eventual large final model.

  - E.g., if the data set involves data from several subjects, try fitting the model to data from one or two subjects first, then add in subjects gradually, refitting several times to progressively bigger data sets.

  - Or if the model involves many parameters, it may be fruitful to first fit a simplified model with fewer parameters and then introduce additional parameters one step at a time.

7. Model Extensions — Heteroscedasticity and Correlation.

As in the linear regression model, it is possible to relax the assumption $\mathbf{e} \sim N(\mathbf{0}, \sigma^2\mathbf{I})$ to account for heteroscedasticity and correlation among the errors/responses.

Consider the nonlinear regression model

$$\mathbf{y} = \mathbf{f}(\mathbf{X}, \boldsymbol{\theta}) + \mathbf{e}, \quad \mathbf{e} \sim N_n(\mathbf{0}, \sigma^2\Lambda) \qquad (\dagger)$$

where Λ is a *known*, positive-definite matrix. Here,

$$\mathbf{f}(\mathbf{X}, \boldsymbol{\theta}) = \begin{pmatrix} f(\mathbf{x}_1, \boldsymbol{\theta}) \\ \vdots \\ f(\mathbf{x}_n, \boldsymbol{\theta}) \end{pmatrix}, \qquad \mathbf{X} = \begin{pmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_n^T \end{pmatrix}.$$

Because Λ is positive definite (the matrix version of a number being positive), it has an invertible square root $\Lambda^{1/2}$ with inverse $\Lambda^{-1/2}$ such that

$$\Lambda = \Lambda^{T/2}\Lambda^{1/2} \quad \text{and} \quad \Lambda^{-1} = \Lambda^{-1/2}\Lambda^{-T/2}.$$

Let

$$\begin{aligned}\mathbf{y}^* &= \Lambda^{-T/2}\mathbf{y},\\ \mathbf{f}^*(\mathbf{X},\boldsymbol{\theta}) &= \Lambda^{-T/2}\mathbf{f}(\mathbf{X},\boldsymbol{\theta})\\ \mathbf{e}^* &= \Lambda^{-T/2}\mathbf{e}\end{aligned}$$

Then we can premultiply both sides of our original model (†) to get an equivalent transformed model

$$\mathbf{y}^* = \mathbf{f}^*(\mathbf{X},\boldsymbol{\theta}) + \mathbf{e}^*, \quad \mathbf{e}^* \sim N_n(\Lambda^{-T/2}\mathbf{0}, \sigma^2\Lambda^{-T/2}\Lambda\Lambda^{-1/2}) = N(\mathbf{0},\sigma^2\mathbf{I}) \tag{$*$}$$

Since this model is of the homoscedastic, uncorrelated form for which nonlinear least squares is appropriate, we can fit model (†) by simply applying (ordinary) NLS to (*). That is, choose $\boldsymbol{\theta}$ to minimize

$$\begin{aligned}\|\mathbf{y}^* - \mathbf{f}^*(\mathbf{X},\boldsymbol{\theta})\|^2 &= \{\mathbf{y}^* - \mathbf{f}^*(\mathbf{X},\boldsymbol{\theta})\}^T\{\mathbf{y}^* - \mathbf{f}^*(\mathbf{X},\boldsymbol{\theta})\}\\ &= \{\mathbf{y} - \mathbf{f}(\mathbf{X},\boldsymbol{\theta})\}^T\Lambda^{-1}\{\mathbf{y} - \mathbf{f}(\mathbf{X},\boldsymbol{\theta})\},\end{aligned}$$

which is known as the *generalized least-squares criterion.*

The major limitation of this approach is that it is unusual that we know Λ. Typically, we don't know Λ, although it will often be the case that we will be willing to make an assumption about the form of Λ.

- That is, we will often be willing to assume that $\Lambda = \Lambda(\boldsymbol{\lambda})$ is a function of an unknown parameter $\boldsymbol{\lambda}$ typically of much smaller dimension than the number of non-repeated elements of Λ ($n(n+1)/2$).
- Under such an assumption, the model parameters are now $\boldsymbol{\theta}, \boldsymbol{\lambda}$, and σ^2.

There are a variety of methods that can be used to estimate such an extended nonlinear regression model, but for now, we focus on maximum likelihood estimation.

ML Estimation in the Extended Nonlinear Regresison Model:

Recall that in ML estimation, we choose as our estimator of a parameter, the value that minimizes the joint probability density function of the data, viewed as a function of the parameter.

For the model (†), the joint density of $\mathbf{y}$ is a multivariate normal density. Thus the loglikelihood function is

$$\ell(\boldsymbol{\theta}, \sigma^2, \boldsymbol{\lambda}; \mathbf{y}) = -\frac{1}{2}\left\{ n \log(2\pi\sigma^2) + \log\det\{\Lambda(\boldsymbol{\lambda})\} + \frac{\|\mathbf{y}^* - \mathbf{f}^*(\mathbf{X}, \boldsymbol{\theta})\|^2}{\sigma^2} \right\}.$$

For fixed $\boldsymbol{\theta}$ and $\boldsymbol{\lambda}$, the maximum likelihood estimator of σ^2 is

$$\hat{\sigma}^2(\boldsymbol{\theta}, \boldsymbol{\lambda}) = \frac{1}{n}\|\mathbf{y}^* - \mathbf{f}^*(\mathbf{X}, \boldsymbol{\theta})\|^2.$$

Plugging this value into the loglikelihood $\ell(\boldsymbol{\theta}, \sigma^2, \boldsymbol{\lambda}; \mathbf{y})$ we obtain the *profile likelihood* which is the log likelihood as a function of $\boldsymbol{\theta}$ and $\boldsymbol{\lambda}$ only:

$$\begin{aligned} p\ell(\boldsymbol{\theta}, \boldsymbol{\lambda}; \mathbf{y}) &\equiv \ell(\boldsymbol{\theta}, \hat{\sigma}^2(\boldsymbol{\theta}, \boldsymbol{\lambda}), \boldsymbol{\lambda}; \mathbf{y}) \\ &= -\frac{1}{2}\left\{ n[\log(2\pi/n) + 1] + \log\det\{\Lambda(\boldsymbol{\lambda})\} + n \log \|\mathbf{y}^* - \mathbf{f}^*(\mathbf{X}, \boldsymbol{\theta})\|^2 \right\} \end{aligned}$$

The MLEs of $\boldsymbol{\theta}$ and $\boldsymbol{\lambda}$ are obtained by maximizing $p\ell(\boldsymbol{\theta}, \boldsymbol{\lambda}; \mathbf{y})$ with respect to these parameters. This is done by alternating between fixing $\boldsymbol{\lambda}$ and maximizing w.r.t. $\boldsymbol{\theta}$ and fixing $\boldsymbol{\theta}$ and maximizing w.r.t. $\boldsymbol{\lambda}$. This procedure is iterated to convergence.

That is, at the k^{th} iteration, we fix $\boldsymbol{\lambda}$ at its current estimate $\hat{\boldsymbol{\lambda}}^{k-1}$ and maximize $p\ell(\boldsymbol{\theta}, \hat{\boldsymbol{\lambda}}^{k-1}; \mathbf{y})$ w.r.t. $\boldsymbol{\theta}$. Note that the resulting maximizer, $\hat{\boldsymbol{\theta}}^k$ is the minimizer of

$$\|\mathbf{y}^{*,k-1} - \mathbf{f}^{*,k-1}(\mathbf{X}, \boldsymbol{\theta})\|^2 = \|\{\Lambda(\hat{\boldsymbol{\lambda}}^{k-1})\}^{-T/2}\{\mathbf{y} - \mathbf{f}(\mathbf{X}, \boldsymbol{\theta})\}\|^2$$

where $\hat{\boldsymbol{\lambda}}^{k-1}$ is fixed. So, this step is just a usual nonlinear least squares problem which can be solved by the G-N method.

Then we fix $\boldsymbol{\theta}$ at $\hat{\boldsymbol{\theta}}^k$ and obtain $\hat{\boldsymbol{\lambda}}^k$ by minimizing $p\ell(\hat{\boldsymbol{\theta}}^k, \boldsymbol{\lambda}; \mathbf{y})$ w.r.t. $\boldsymbol{\lambda}$.

We then repeat these steps until convergence.

The MLE of σ^2 can be obtained at convergence as $\hat{\sigma}^2 = \sigma^2(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\lambda}})$ (by simply plugging in the MLEs $\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\lambda}}$ into the formula from the previous page). However, to reduce the bias of this estimator, we use instead and MSE-type estimator:

$$\tilde{\sigma}^2 = \frac{\|\hat{\Lambda}^{-T/2}\{\mathbf{y} - \mathbf{f}(\mathbf{X}, \hat{\boldsymbol{\theta}})\}\|^2}{n - p},$$

where $\hat{\Lambda} = \Lambda(\hat{\boldsymbol{\lambda}})$.

Inference for $\boldsymbol{\theta}$ is based on "classical" asymptotic theory for ML estimation. The asymptotic distribution of $\hat{\boldsymbol{\theta}}$ is

$$\hat{\boldsymbol{\theta}} \overset{a}{\sim} N_p\left(\boldsymbol{\theta}, \sigma^2\left[\{\mathbf{V}(\boldsymbol{\theta})\}^T\Lambda^{-1}\mathbf{V}(\boldsymbol{\theta})\right]^{-1}\right)$$

with var-cov matrix that can be consistently estimated as

$$\hat{\text{avar}}(\hat{\boldsymbol{\theta}}) = \sigma^2\left[\{\mathbf{V}(\hat{\boldsymbol{\theta}})\}^T\hat{\Lambda}^{-1}\mathbf{V}(\hat{\boldsymbol{\theta}})\right]^{-1}$$

In addition, $\frac{n-p}{\sigma^2}\tilde{\sigma}^2 \overset{a}{\sim} \chi^2(n-p)$ and $\hat{\boldsymbol{\theta}}$ and $\tilde{\sigma}^2$ are asymptotically independent, so the usual approximate F and t tests for inference on $\boldsymbol{\theta}$ can be performed.

We can take advantage of this methodology to fit a much broader class of nonlinear models than we have so far considered. To describe this broader class, it's convenient to decompose $\text{var}(\mathbf{e}) = \sigma^2\Lambda$ as follows:

$$\text{var}(\mathbf{e}) = \sigma^2\Lambda = \sigma^2\mathbf{V}^{1/2}\mathbf{C}\mathbf{V}^{1/2}$$

where

$$\mathbf{V}^{1/2} = \frac{1}{\sigma}\begin{pmatrix} \sqrt{\text{var}(e_1)} & 0 & 0 & \cdots & 0 \\ 0 & \sqrt{\text{var}(e_2)} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \sqrt{\text{var}(e_n)} \end{pmatrix}$$

and

$$\mathbf{C} = \text{corr}(\mathbf{e}) = \begin{pmatrix} 1 & \text{corr}(e_1, e_2) & \text{corr}(e_1, e_3) & \cdots & \text{corr}(e_1, e_n) \\ \text{corr}(e_2, e_1) & 1 & \text{corr}(e_2, e_3) & \cdots & \text{corr}(e_2, e_n) \\ \vdots & \vdots & \vdots & \ddots & \\ & \vdots & & & \\ \text{corr}(e_n, e_1) & \text{corr}(e_n, e_2) & \text{corr}(e_1, e_3) & \cdots & 1 \end{pmatrix}$$

Heteroscedasticity specification: We assume, for now, that

$$\text{var}(e_i) = \sigma^2 g^2(\mathbf{v}_i, \boldsymbol{\delta})$$

where $\mathbf{v}_i$ is a vector of *variance covariates*, $\boldsymbol{\delta}$ is a vector of *variance parameters* to be estimated (part of $\boldsymbol{\lambda}$), and $g^2(\cdot)$ is a known *variance function.*

- Later, we will allow that $g^2(\cdot)$ depend also on $\mu_i = \text{E}(y_i)$, the mean response, but that will take us out of the context of ML estimation.

Correlated errors are often appropriate when there is some temporal or spatial dependence among the observations.

For example, suppose we have data on CO_2-uptake in plants measured over a one hour period in which the plant was exposed to different light intensities. Suppose these measurements were taken during consecutive one hour periods. E.g., suppose our data are as follows:

Plant	CO_2-uptake	Light Intensity	Time (hrs)
1	0	0	1
1	.33	20	2
1	2.5	80	3
1	.	120	4
1	6.1	150	5
1	6.3	250	6
2	0	0	1
⋮			

In such a situation, we may expect there to be a *serial dependence* structure to the errors, where observations close together in time are correlated with the strength of the correlation decreasing with the time lag.

E.g., we may use an Autoregressive structure of order 1 (AR(1)) where $\text{corr}(e_i, e_j) = \rho^{|t_i - t_j|}$ where t_i is an integer-valued measurement time for response y_i.

Correlation Specification: In general, our correlation model will be

$$\text{corr}(e_i, e_j) = h\{d(\mathbf{p}_i, \mathbf{p}_j), \boldsymbol{\rho}\}$$

where $\boldsymbol{\rho}$ is a vector of *correlation parameters*, $h(\cdot)$ is a known *correlation function*, $\mathbf{p}_i, \mathbf{p}_j$ are position vectors (often scalars for serial (time) correlation) for observations $\mathbf{y}_i, \mathbf{y}_j$, and $d(\cdot, \cdot)$ is a known *distance function*.

- The correlation function $h(\cdot)$ is assumed continuous in $\boldsymbol{\rho}$, returning values in $[-1, +1]$. In addition, $h(0, \boldsymbol{\rho}) = 1$, so that observations that are 0 distance apart (identical observations) are perfectly correlated.
- In our example above, $\mathbf{p}_i = t_i$, $d(t_i, t_j) = |t_i - t_j|$ and $\boldsymbol{\rho}$ was a scalar ρ.
- Vector valued positions $\mathbf{p}_i$ arise when modeling spatial data. E.g., $\mathbf{p}_i$ could be bivariate containing the longitude and latitude of y_i.

A wide variety of models can be specified within this general framework and fit using ML estimation as described above.

These models all assume $\text{var}(\mathbf{e}) = \sigma^2 \Lambda(\boldsymbol{\lambda})$ where the variance-covariance parameter $\boldsymbol{\lambda}$ consists of two parts: $\boldsymbol{\delta}$, the variance parameter; and $\boldsymbol{\rho}$, the correlation parameter. That is, $\boldsymbol{\lambda} = (\boldsymbol{\delta}^T, \boldsymbol{\rho}^T)^T$.

Variance Functions Available in the nlme Software (e.g., in gnls()):

- Variance functions in the nlme software are described in §5.2.1 in Pinheiro and Bates (2000) (see also ?varClasses in the nlme documentation). Here, we give only brief descriptions.

1. varFixed. The varFixed variance function is $g^2(v_i) = v_i$. That is,

$$\text{var}(e_i) = \sigma^2 v_i,$$

   the error variance is proportional to the value of a covariate. This is the common weighted least squares form.

2. varIdent. This variance specification corresponds to different variances at each level of some stratification variable s. That is, suppose s_i takes values in the set $\{1, 2, \ldots, S\}$ corresponding to S different groups (strata) of observations. Then we assume that observations in stratum 1 have variance σ^2, observations in stratum 2 have variance $\sigma^2\delta_1$, ..., and observations in startum S have variance $\sigma^2\delta_S$.

   That is,

$$\text{var}(e_i) = \sigma^2\delta_{s_i}, \qquad \text{so that } g^2(s_i, \boldsymbol{\delta}) = \delta_{s_i}$$

   where, for identifiability we take $\delta_1 = 1$.

3. varPower. This generalizes the varFixed function so that the error variance can be a to-be-estimated power of the magnitude of a variance covariate:

$$\text{var}(e_i) = \sigma^2 |v_i|^{2\delta} \qquad \text{so that } g^2(v_i, \delta) = |v_i|^{2\delta}.$$

The power is taken to be 2δ rather than δ so that $\text{s.d.}(e_i) = \sigma |v_i|^{\delta}$.

A very useful specification is to take the variance covariate to be the mean response. That is,

$$\text{var}(e_i) = |\mu_i|^{2\delta}$$

However, this corresponds to $g^2(\mu_i, \delta) = |\mu_i|^{2\delta}$ depending upon the mean. Such a model is fit with a variant of the ML estimation algorithm. However, this technique is not maximum likelihood, and indeed ML estimation is not recommended for such a model. Instead the method is what is known as *pseudo likelihood estimation*.

4. varConstPower. The idea behind this specification is that varPower can often be unrealistic when the variance covariate takes values close to 0. The varConstPower model specifies

$$\text{var}(e_i) = \sigma^2 \left(\delta_1 + |v_i|^{\delta_2}\right)^2, \quad \delta_1 > 0.$$

That is, for $\delta_2 > 0$ (as is usual), the variance function is approximately constant and equal to δ_1^2 for values of the variance covariate close to 0, and then it increases as a power of $|v_i|$ as v_i increases in magnitude away from 0.

5. varExp. The variance model for varExp is

$$\text{var}(e_i) = \sigma^2 \exp(2\delta v_i)$$

6. varComb. Finally, the varComb class allows the preceding variance classes to be combined so that the variance function of the model is a product of two or more component variance functions.

Correlation Structures Available in the nlme Software (e.g., in gnls()):

- nlme includes correlation structures to account for time dependence (serial correlation structures) and spatial dependence (spatial correlation structures). It also has a couple of generally applicable correlation structures.

- Correlation structures in the nlme software are described in §5.3 in Pinheiro and Bates (2000) (see also ?corClasses in the nlme documentation). Here, we give brief descriptions of the serial and general correlation structures.

Serial Correlation Structures:

The work-horse class of models in time-series analysis is the class of **Autoregressive-Moving Average** (ARMA) models.

We will apply these models to the errors, e_i, but for notational convenience let's index e by t to indicate time.

In an Autoregressive (AR) model, we assume the current observation e_t is a linear function of previous observations plus "white noise" (a mean zero, constant variance error term):

$$e_t = \phi_1 e_{t-1} + \cdots + \phi_p e_{t-p} + a_t, \quad \mathrm{E}(a_t) = 0, \mathrm{var}(a_t) = \sigma^2.$$

- The number of previous observations on which e_t depends, p, is called the order of the process and we write $AR(p)$.

- The simplest, and practically most useful, AR model is an AR(1):

$$e_t = \phi e_{t-1} + a_t, \quad \text{where } -1 < \phi < +1.$$

- For an AR(1) model,

$$\mathrm{corr}(e_t, e_s) = \phi^{|t-s|}$$

  and ϕ represents the correlation between two observations one time unit apart.

A Moving-Average model is one in which the observation e_t at time t is a linear combination (weighted average, in some sense) of past independent and identically distributed white noise error terms plus a current time white noise error:

$$e_t = \theta_1 a_{t-1} + \cdots + \theta_q a_{t-q} + a_t$$

- The number of past errors on which e_t depends is the *order* of the process, so above we have an MA(q) process.
- Again, an order 1, in this case MA(1), process is often useful. For an MA(1),

$$\text{corr}(e_t, e_s) = \begin{cases} 1, & \text{if } s = t; \\ \theta_1/(1+\theta_1^2) & \text{if } |s-t| = 1; \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

- In general, MA(q) processes have nonzero correlation for observations $\leq q$ time units apart and 0 correlation for observations $> q$ time units apart.

Combining an AR(p) process with a MA(q) process we get an ARMA(p, q) process:

$$e_t = \sum_{i=1}^{p} \phi_i e_{t-i} + \sum_{j=1}^{q} \theta_j a_{t-j} + a_t.$$

- It is always possible to model any autocovariance structure to an arbitrarily small level of precision with a high enough order AR or MA process. Often, we will find that a very low order AR, MA, or ARMA model will suffice.

1. corAR1. This correlation structure is specified as corAR1(value, form = *one-sided formula*), where value specifies an (optional) inital value for estimating the AR(1) parameter ϕ and *one-sided formula* is a formula of the form:

   $$\sim \text{covariate}|\text{Groupingvariable}$$

   Here, the covariate is an integer-valued time index and |Groupingvariable is an optional group specification. Groups are specified to be units of observations on which repeated measurements through time are taken.

- For example, in the example CO_2-uptake example above, the specification corAR1(.8, form = $\sim$ time | Plant) would yield the following correlation matrix for plant 1:

$$\mathbf{C} = \begin{pmatrix} 1 & \rho & \rho^2 & \rho^4 & \rho^5 \\ & 1 & \rho & \rho^3 & \rho^4 \\ & & 1 & \rho^2 & \rho^3 \\ & & & 1 & \rho \\ & & & & 1 \end{pmatrix}$$

  with initial value of .8 for ρ.

2. corCAR1. This correlation structure is a continuous-time version of an AR(1) correlation structure. The specification is the same as in corAR1, but now the covariate indexing time can take any non-negative non-repeated value and we restrict $\phi \geq 0$.

3. corARMA. This correlation structure corresponds to an ARMA(p, q) model. AR(p) and MA(q) models can be specified with this function, but keep in mind that the corAR1 specification is more efficient than specifying corARMA with $p = 1$ and $q = 0$.

   We can specify an ARMA(1,1) model with initial values of $\phi = .8, \theta = .4$ via corARMA(value = c(.8,.4), form = $\sim$ covariate|Groupingvariable, p=1, q=1).

General Correlation Structures:

1. corCompSymm. In this structure,

$$\text{corr}(e_i, e_j) = \begin{cases} 1 & \text{if } i = j; \text{ and} \\ \rho & \text{if } i \neq j. \end{cases}$$

That is, the correlation between any two distinct observations is the same. Like many of the correlation structures, this structure is often useful within groups. For our example, the specification corCompSymm(value=.3, form = $\sim 1 \mid$ Plant) defines the correlation matrix

$$\mathbf{C} = \begin{pmatrix} 1 & \rho & \rho & \rho & \rho \\ & 1 & \rho & \rho & \rho \\ & & 1 & \rho & \rho \\ & & & 1 & \rho \\ & & & & 1 \end{pmatrix}$$

with initial value of .3 for ρ.

2. corSymm. Specifies a completely general correlation structure with a separate parameter for every non-redundant correlation. E.g., in our example corSymm(form = $\sim 1 \mid$ Plant) specifies the correlation matrix

$$\mathbf{C} = \begin{pmatrix} 1 & \rho_1 & \rho_2 & \rho_3 & \rho_4 \\ & 1 & \rho_5 & \rho_6 & \rho_7 \\ & & 1 & \rho_8 & \rho_9 \\ & & & 1 & \rho_{10} \\ & & & & 1 \end{pmatrix}$$

where initial values for $\boldsymbol{\rho}$ can be supplied with an optional value= specification.

Spatial Correlation Structures:

- A classic reference on spatial statistics is Cressie, *Statistics for Spatial Data.* The following material is based on Pinheiro and Bates (2000, §5.3), who base their treatment on material in Cressie's book.

Let $e_{\mathbf{p}}$ denote the observation (error term in our nonlinear model) corresponding to position $\mathbf{p} = (p_1, p_2, \ldots, p_r)^T$.

- Often $r = 2$ and $\mathbf{p} = (p_1, p_2)^T$ gives two dimensional coordinates.

Time series correlation structures are typically described by their autocorrelation function (which we've denoted $h(\cdot)$ above). Spatial correlation structures are usually described by their *semivariogram.*

For a given distance function $d(\cdot)$, the semivariogram is a function γ of the distance between two points $e_{\mathbf{p}}$ and $e_{\mathbf{q}}$ say, and a parameter $\boldsymbol{\rho}$, that measures the association between two points that distance apart:

$$\gamma\{d(e_{\mathbf{p}}, e_{\mathbf{q}}), \boldsymbol{\rho}\} = \frac{1}{2}\mathrm{var}(e_{\mathbf{p}} - e_{\mathbf{q}})$$

We assume the observations have been standardized to have $\mathrm{E}(e_{\mathbf{p}}) = 0$ and $\mathrm{var}(e_{\mathbf{p}}) = 1$ for all $\mathbf{p}$. Such a standardization does not alter the correlation structure.

In that case, it is easy to see the relationship between the semivariogram $\gamma(\cdot)$ and the autocorrelation function $h(\cdot)$:

$$\gamma(s, \boldsymbol{\rho}) = 1 - h(s, \boldsymbol{\rho}).$$

From this relationship it is clear that observations 0 distance apart have $h(0, \boldsymbol{\rho}) = 1$ and thus $\gamma(0, \boldsymbol{\rho}) = 0$. The autocorrelation function h increases continuously to 1 as the distance decreases to 0. Hence the semivariogram increases continuously to 0 as distance decreases to 0.

In some applications it is useful to violate this by introducing a *nugget effect* into the definition of the semivariogram. This nugget effect is a parameter c_0 that forces $\gamma(0, \boldsymbol{\rho}) = c_0$ where $0 < c_0 < 1$ rather than $\gamma(0, \boldsymbol{\rho}) = 0$ when the distance between the observations is 0.

The following spatial correlation structures are implemented in the nlme software in R. All have a scalar-valued correlation parameter ρ. This parameter is known as the *range* in the spatial literature.

1. corExp. (Exponential) This structure corresponds to the semivariogram
$$\gamma(s, \rho) = 1 - \exp(-s/\rho)$$
and the autocorrelation function $h(s, \rho) = \exp(-s/\rho)$.

2. corGauss. (Gaussian) This structure corresponds to the semivariogram
$$\gamma(s, \rho) = 1 - \exp\{-(s/\rho)^2\}$$
and the autocorrelation function $h(s, \rho) = \exp\{-(s/\rho)^2\}$.

3. corLinear. (Linear) This structure corresponds to the semivariogram
$$\gamma(s, \rho) = 1 - (1 - s/\rho)1_{\{s<\rho\}}$$
and the autocorrelation function $h(s, \rho) = (1 - s/\rho)1_{\{s<\rho\}}$. Here $1_{\{A\}}$ represents the indicator variable that equals 1 when condition A is true, 0 otherwise.

4. corRatio. (Rational Quadratic) This structure corresponds to the semivariogram
$$\gamma(s, \rho) = \frac{(s/\rho)^2}{1 + (s/\rho)^2}$$
and the autocorrelation function $h(s, \rho) = \{1 + (s/\rho)^2\}^{-1}$.

5. corSpher. (Spherical) This structure corresponds to the semivariogram
$$\gamma(s, \rho) = 1 - \{1 - 1.5(s/\rho) + .5(s/\rho)^3\}1_{\{s<\rho\}}.$$

- A nugget effect can be added to any of these structures. With a nugget effect c_0, the semivariogram with the nugget effect $\gamma_{\text{nugg}}(\cdot)$ is defined in terms of the semivariogram without the nugget effect $\gamma(\cdot)$ as follows:

$$\gamma_{\text{nugg}}(s, c_0, \rho) = \begin{cases} c_0 + (1 - c_0)\gamma(s, \rho), & \text{if } s > 0; \text{ and} \\ 0, & \text{otherwise.} \end{cases}$$

- When using the above spatial correlation structures, the user can choose between distance metrics. Currently implemented distance metrics are *Euclidean distance*, $d(e_{\mathbf{p}}, e_{\mathbf{q}}) = \|\mathbf{p}-\mathbf{q}\| = \sqrt{\sum_{i=1}^{r}(p_i - q_i)^2}$, *Manhattan distance*, $d(e_{\mathbf{p}}, e_{\mathbf{q}}) = \sum_{i=1}^{r} |p_i - q_i|$, and *maximum distance*, $d(e_{\mathbf{p}}, e_{\mathbf{q}}) = \max_{i=1,\ldots,r} |p_i - q_i|$.

- One can get a feel for these various semivariogram models by examining them as functions of distance for different choices of the range parameter ρ and the nugget effect c_0. The 5 semivariograms listed above are plotted below for $\rho = 1$, $c_0 = .1$.

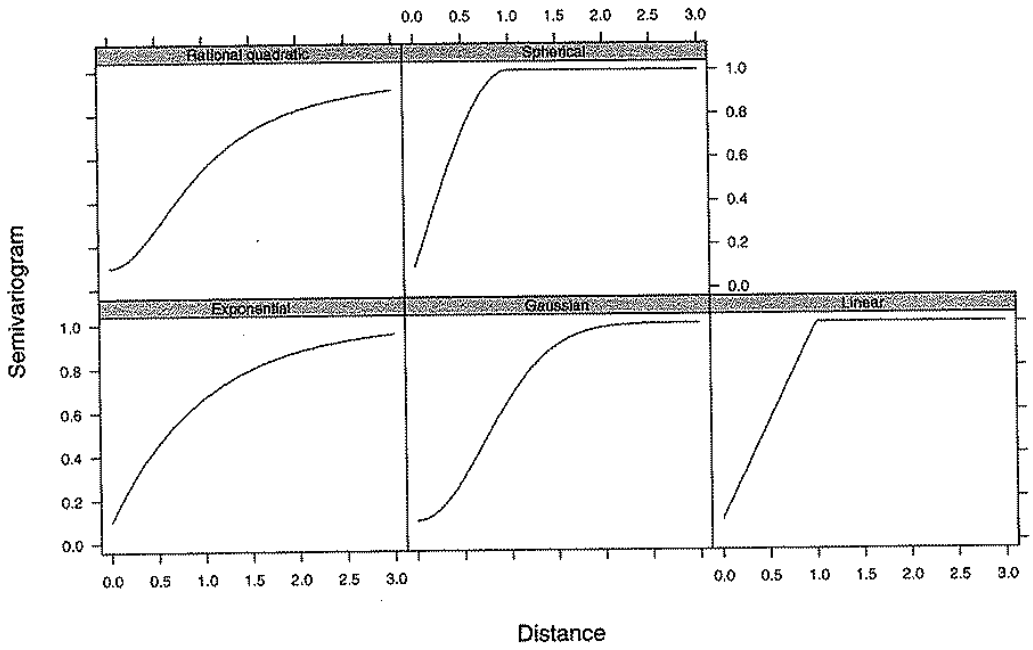


FIGURE 5.9. Plots of semivariogram versus distance for the isotropic spatial correlation models in Table 5.2 with range = 1 and nugget effect = 0.1.

Q: *How do we choose a correlation structure?*

A: This is a hard question that would be the focus of several weeks worth of study in a time series or spatial statistics course.

In a regression context, inference on the regression parameters is the primary interest. We need to account for a correlation structure if one exists to get those inferences right, but we're typically not interested in the correlation structure in and of itself. Therefore, we opt for simple correlation structures that capture "most of the correlation" without getting caught up in extensive correlation modeling.

In a time-dependence context, AR(1) models are often sufficient.

If we are willing to consider other ARMA models, two tools that are useful in selecting the right ARMA model are the *sample autocorrelation function* (ACF) and the *sample partial autocorrelation function* (PACF).

Let

$$r_i = \frac{y_i - \hat{y}_i}{\sqrt{\hat{\text{var}}(e_i)}}$$

denote the standardized residuals from a fitted nonlinear model.

The sample autocorrelation at lag ℓ is defined as

$$\hat{\rho}(\ell) = \frac{\sum_{k=1}^{n-\ell} r_k r_{k+\ell}/(n-\ell)}{\sum_{k=1}^{n} r_k^2/n}, \qquad \ell = 1, 2, \ldots$$

The sample partial autocorrelation at lag ℓ is a sample estimate of the correlation between observation e_t and $e_{t-\ell}$ after removing the effects of $e_{t-1}, \ldots, e_{t-\ell+1}$. The estimate is obtained by a recursive formula not worth reproducing here.

- AR(p) models have PACFs that are non-zero for lags $\leq p$ and 0 for lags $> p$. Therefore, we can look at the magnitude of the sample PACF to try to identify the order of an AR process that will fit the data. The number of "significant" partial autocorrelations is a good guess at the order of an appropriate AR process.

- MA(q) models have ACFs that are nonzero for lags $\leq q$ and 0 for lags $> q$. Again, we can look at the sample ACF to choose q.

- ARMA(p, q) models will have sample ACFs and PACFs that don't fit these simple rules. The following table (from Seber & Wild, Ch. 6, p.320) describes the general behavior of the ACF and PACF functions for various ARMA models.

Table 6.3 Properties of the ACF and the PACF for Various ARMA Models[a]

Model	ACF	PACF
AR(1)	Exponential or oscillatory decay	$\phi_{kk} = 0$ for $k > 1$
AR(2)	Exponential or sine wave decay	$\phi_{kk} = 0$ for $k > 2$
AR(q_1)	Exponential and/or sine-wave decay	$\phi_{kk} = 0$ for $k > q_1$
MA(1)	$\rho_k = 0$ for $k > 1$	Dominated by damped exponential
MA(2)	$\rho_k = 0$ for $k > 2$	Dominated by damped exponential or sine wave
MA(q_2)	$\rho_k = 0$ for $k > q_2$	Dominated by linear combination of damped exponentials and/or sine waves
ARMA(1, 1)	Tails off. Exponential decay from lag 1	Tails off. Dominated by exponential decay from lag 1
ARMA(q_1, q_2)	Tails off after $q_2 - q_1$ lags Exponential and/or sine wave decay after $q_2 - q_1$ lags	Tails off after $q_1 - q_2$ lags Dominated by damped exponentials and/or sine waves after $q_1 - q_2$ lags

[a]From Abraham and Ledolter [1983] with permission from John Wiley and Sons.

Similarly, in a spatial setting, we can estimate and plot the semivariogram to help choose an appropriate spatial correlation model. The Variogram() function in the nlme library will compute either one of two semivariogram estimators: the classical estimator, and a robust estimator that reduces the influence of outliers. See Pinheiro and Bates (2000, p.231) for formulas.

Example — Cumulative Bole Volume of a Sweetgum Tree

For purposes of forest inventory and planning, it is useful to have bole-volume equations to predict the volume of trees while standing based on easily measured tree size variables such as diameter at breast height (DBH) and stem height. Merchantable volume is the volume from stump height to a specified upper-bole diameter which establishes the merchantability limit. Since this limit is subject to change according to technological capabilities and economic conditions, it is useful to predict cumulative volume to an upper diameter d that is variable. Typically, a sample of trees spanning the range of relevant sizes is felled. For each tree, cumulative bole volumes V_d and stem diameters d are measured at a series of ascending heights, and the DBH and total stem height H are measured. In this example, we consider data from just one tree (tree #5) in a larger data set analyzed by Gregoire and Schabenberger (1996, *JABES*) and Davidian & Giltinan (1995, §11.3).

A partial listing of the data is given below:

Tree No.	DBH	Height (H)	Stem Diameter (d)	Measurement Height	Cumulative Volume
5	11.8	97.6	10.7	7.2	5.27
5	11.8	97.6	10.4	10.2	7.09
5	11.8	97.6	10.3	13.2	8.85
⋮	⋮	⋮	⋮	⋮	⋮
5	11.8	97.6	1.1	88.2	27.09

- It seems reasonable to expect that cumulative volume measurements here will be subject to spatial correlation. Measurements taken at locations that are close to one another, especially observations at adjacent locations, can be expected to be correlated, with correlation that we expect to decrease with distance.

- Although the correlation is spatial, there is only one dimension here — length along the stem — and measurements are taken at equally spaced locations along this dimension. Therefore, we can handle these data as though they represent a time series, with time index $t = 1, 2, \ldots, 24$ ($t = (\text{Measurement Ht} - 4.2)/3$).

- See handout sweetgum1.

- In this R script we follow Davidian and Giltinan in considering models for cumulative bole volume V as a function of $x = \log(DBH - d)$. We first plot V versus x. This plot reveals a sigmoidal form. Since the response is cumulative volume, this form is not unexpected. It has the same general form as a cumulative distribution function.

- While a variety of sigmoidal curves might be chosen to model these data, we follow Davidian and Giltinan in considering models with a logistic expectation function,

$$f(x, \boldsymbol{\theta}) = \frac{\theta_1}{1 + \exp\{(\theta_2 - x)/\theta_3\}}.$$

- We first consider m1, a logistic model with spherical errors. A plot of the residuals doesn't show a pattern suggestive of heteroscedasticity. However, to investigate possible heteroscedasticity, we refit the model, now as model m2, with variance function

$$g^2(\mu_i, \delta) = |\mu_i|^{2\delta}.$$

  The anova() function provides a LRT, as well as AIC and BIC information criteria to compare models m1 and m2. According to the test and the criteria, m2 does not fit significantly better than m1 and the residual plot for m2 looks almost identical to that for m1. Therefore, we reject model m2 and conclude that the data are homoscedastic.

- Next we obtain plots of the sample ACF and PACF for model m1. The nlme library contains an ACF() function that computes the sample ACF directly from a fitted gnls object. We specify the number of lags for which we want autocorrelations computed with the maxLag= option. When ploting the ACF we can request error bars on the ACF function for any given confidence level $100(1-\alpha)\%$. These error bars are placed at

$$\pm z_{1-\alpha/2}/n(\ell), \quad \ell = 1, 2, \ldots,$$

  where $n(\ell)$ = the number of residual pairs that went into the cacculation of the autocorrelation at lag ℓ.

- There is no PACF function in nlme, but we can obtain the PACF* by first extracting the standardized residuals from our model (e.g., resid(m1, type="p")) and inputting that into the ar() function. The results of the ar() function can then be input into acf.plot to yield the PACF plot. This plot also contains a 95% error bar by default. (The error bars here replace $n(\ell)$ with n in the formula above, so they stay constant over increasing lags. There seems to be some difference of opinion in the literature on which is more appropriate, $n(\ell)$ or n).

- The sample ACF and PACF plots are not definitive in this example. What we'd really like to see is no significant (partial) autocorrelations, or only one significant (partial) autocorrelation at lag 1. Without a clear indication of the true correlation structure from these plots, we consider several ARMA structures, using a trial and error approach.

---

* Note that this approach is only appropriate for data taken over an equally spaced time index *from a single subject/unit.* This is not appropriate for longitudinal/repeated measures from several subjects in the same dataset.

- In models m3–m7 we fit an AR(1) model, an AR(2), an ARMA(1,1), an AR(3), and an ARMA(2,1) model, respectively. ACF plots are produced for all of these models, and a PACF model is produced for m3. According to the AIC criterion, model m3 (the AR(1) model) is the winner, and its ACF and PACF plots look pretty good.

- Its very important to note that the PACF and ACF plots for models with correlation should always be based on the *normalized residuals*. The vector of normalized residuals is defined as

$$\mathbf{r} = \hat{\sigma}^{-1}(\hat{\Lambda}^{-T/2})(\mathbf{y} - \hat{\mathbf{y}}).$$

  If the estimated variance-covariance structure is correct, $\mathbf{r}$ should be approximately mean $\mathbf{0}$ with var-cov matrix $\mathbf{I}$. Therefore, if the mean, correlation and heteroscedasticity models are correct, then the ACF from $\mathbf{r}$ should reflect independent white noise (no pattern and no significant autocorrelations).

- Note that while the parameter estimates don't differ much between model m1 and m3, their standard errors do. It is important to account for correlation (and heteroscedasticity) to obtain correct inferences.

- The general strategy to choosing an error var-cov structure that we have taken here is to combine residual analysis with LRTs and information criteria.

- Finally, note that the ACF() function in the nlme library will work both for a single time series like this one and for grouped data, where we essentially have several time series that we are analyzing simultaneously (e.g., if we had been modeling the cumulative volume of several trees at once).

  - However, for data from multiple time series (i.e. longitudinal data), the steps we took above to plot the PACF **will not work** and should not be followed!

  - At present there is no easy way to plot the PACF for a model fit to longitudinal data in S-PLUS, R.

Because of its usefulness, consider again the AR(1) model.

Suppose the homoscedastic AR(1) model holds. That is, suppose the true model is

$$y_t = f(\mathbf{x}_t, \boldsymbol{\theta}) + e_t, \quad \text{where} \quad e_t = \phi e_{t-1} + a_t,$$

for $t = 1, \ldots, n$.

A simple approach to fitting this model is to transform it to a model with uncorrelated errors by subtracting ϕ times the model at the previous time period:

$$y_t - \phi y_{t-1} = f(\mathbf{x}_t, \boldsymbol{\theta}) - \phi f(\mathbf{x}_{t-1}, \boldsymbol{\theta}) + \underbrace{e_t - \phi e_{t-1}}_{=a_t}, \quad t = 2, \ldots, n.$$

Equivalently,

$$y_t = \phi y_{t-1} + f(\mathbf{x}_t, \boldsymbol{\theta}) - \phi f(\mathbf{x}_{t-1}, \boldsymbol{\theta}) + a_t, \quad t = 2, \ldots, n. \qquad (*)$$

where a_t has mean 0 and constant variance.

Thus, we can estimate $\boldsymbol{\theta}$ and the AR(1) parameter ϕ by fitting (*) with ordinary nonlinear least-squares. Note that we fit the model to observations $2, \ldots, n$. This approach is known as *conditional least squares.*

Because we must throw out one observation in this approach, we can expect it to be less efficient than ML estimation, expecially when n is small.

- Although this is an appealing, simple approach, it is not recommended in general.

A big improvement to CLS can be made without necessitating anything more sophisticated than ordinary nonlinear least squares. The technique is as follows

0. Fit the model

$$y_t = f(\mathbf{x}_t, \boldsymbol{\theta}) + a_t, \quad t = 1, \ldots, n$$

with ordinary NLS.

1. Compute the moment estimate $\hat{\phi}$ of the AR(1) parameter ϕ based on the residuals $\{\hat{a}_t\}$ of the most recently fit model:

$$\hat{\phi} = \frac{\sum_{t=2}^{n} \hat{a}_t \hat{a}_{t-1}}{\sum_{t=1}^{n} \hat{a}_t^2}.$$

(This is just the sample ACF at lag 1.)

2. Fit the model

$$z_t = g(\mathbf{x}_t, \boldsymbol{\theta}) + a_t, \quad t = 1, \ldots, n,$$

where

$$z_1 = \sqrt{1 - \hat{\phi}^2} y_1, \quad g(\mathbf{x}_1, \boldsymbol{\theta}) = \sqrt{1 - \hat{\phi}^2} f(\mathbf{x}_1, \boldsymbol{\theta})$$

and for $t = 2, 3, \ldots, n$,

$$z_t = y_t - \hat{\phi} y_{t-1}, \quad g(\mathbf{x}_t, \boldsymbol{\theta}) = f(\mathbf{x}_t, \boldsymbol{\theta}) - \hat{\phi} f(\mathbf{x}_{t-1}, \boldsymbol{\theta}).$$

3. Go to step 1. Repeat until convergence.

- This method is called *iterated two-stage estimation* and will usually give an answer very similar to ML estimation. In fact, the one-step version (omit step 3) will often work well.

Sweetgum Example, Continued:

- We illustrate CLS and two-stage estimation on the sweetgum example. In model m8 we refit the AR(1) model using CLS.
- We see that the residual ACF plot for the CLS model m8 looks about the same as the residual ACF plot for the AR(1) model m3. In addition, the regression parameters estimators are similar in the two models. However, there is some disagreement in the standard errors.
- Finally, we implement two-stage estimation in model m9 by taking $\hat{\phi} = .4047$ to be the ML estimator of ϕ from the AR(1), model m3, and performing only one iteration. Notice that the results for model m9 are very similar to those for m3.

An interesting result in our sweetgum example is the effect of accounting for a non-zero correlation structure on the parameter estimates and their standard errors.

For the two homoscedastic models m1 and m3 with independence and AR(1) correlation structures, respectively we had:

m1 (Indep)

```
Coefficients:
        Value Std.Error  t-value
Asym 30.12624 0.6464298 46.60404
xmid  0.99812 0.0328612 30.37395
scal  0.56360 0.0289948 19.43791
```

m3 (AR(1))

```
Coefficients:
        Value Std.Error  t-value
Asym 30.17798  0.9306283 32.42753
xmid  0.99676  0.0491328 20.28710
scal  0.57493  0.0414744 13.86223
```

- The parameter estimates are similar, but the s.e.'s increase substantially when we account for correlation. *Why?*

- Roughly speaking, n dependent observations "contain less information" about the marginal mean than n independent observations. If we fit n dependent observations with an independence model, we "overestimate" the sample size/precision of our parameter estimates.

- In general, for serially correlated data fit with an independence model, parameters associated with time invariant explanatory variables tend to have their s.e.'s underestimated, parameters associated with time varying explanatory variables tend to have their s.e.'s overestimated.

8. Comparison of Models.

Suppose we are considering two or more models and we wish to choose between them to select the model that best describes the data.

We will break this problem into two cases: Nested vs. Non-Nested Models.

Let $f_1(\mathbf{x}, \boldsymbol{\theta})$, $f_2(\mathbf{x}, \boldsymbol{\phi})$ denote two possible expectation functions. That is we are faced with two possible models:

$$\begin{aligned} y_i &= f_1(\mathbf{x}_i, \boldsymbol{\theta}) + e_i, \\ y_i &= f_2(\mathbf{x}_i, \boldsymbol{\phi}) + \epsilon_i, \end{aligned} \qquad i = 1, \ldots, n,$$

where we assume

i. the error var-cov structure is the same in the two models, and

ii. the error var-cov structure is spherical in the two models.

- We will eventually drop assumption (ii.), but we keep it for now.

We say the models are **nested** if one is a special case of the other.

Examples:

1. Obviously nested models:

$$f_1(x, \boldsymbol{\theta}) = \frac{\theta_1}{\theta_2 + x}$$
$$f_2(x, \boldsymbol{\phi}) = \phi_3 + \frac{\phi_1}{\phi_2 + x}$$

   Clearly, f_1 is a special case of f_2 corresponding to $\phi_1 = \theta_1, \phi_2 = \theta_2, \phi_3 = 0$.

2. Not-so-obviously nested models:

$$f_1(x, \boldsymbol{\theta}) = \frac{\theta_1}{\theta_2 + x}$$
$$f_2(x, \boldsymbol{\phi}) = \frac{1}{\phi_1 + \phi_2 x + \phi_3 x^2}$$

   These are nested as well, because we can re-parameterize f_1 as

$$f_1 = \frac{1}{\phi_1 + \phi_2 x}$$

   where $\phi_1 = \theta_2/\theta_1$ and $\phi_2 = 1/\theta_1$.

3. Non-nested models:

$$f_1(x, \boldsymbol{\theta}) = \theta_1(1 - e^{-\theta_2 x})$$
$$f_2(x, \boldsymbol{\phi}) = \frac{\phi_1}{1 + e^{\phi_2 - \phi_3 x}}$$

   To show these models are non-nested one need only show that for one choice of $\boldsymbol{\theta}$ it is not possible to find a $\boldsymbol{\phi}$ so that $f_1 = f_2$.

   E.g., for $\boldsymbol{\theta} = (1, 1)^T$,

$$f_1(x, \boldsymbol{\theta}) = 1 - e^{-x} \neq \frac{\phi_1}{1 + e^{\phi_2 - \phi_3 x}} \quad \text{for any } \boldsymbol{\phi}.$$

1. Nested Models.

   Let f_1, f_2 denote expectation functions and suppose f_1 is a special case of f_2.

   Q: Is the added complexity of f_2 needed?

   That is, we wish to test

   $$H_0 : (f_1 \text{ and } f_2 \text{ both hold}), \quad \text{versus} \quad H_1 : (\text{only } f_2 \text{ holds}).$$

   By analogy with the linear model, we might consider the test statistic

   $$F = \frac{(\mathrm{SS_{E1}} - \mathrm{SS_{E2}})/(\mathrm{dfE_1} - \mathrm{dfE_2})}{\mathrm{SS_{E2}}/\mathrm{dfE_2}}$$

   where $\mathrm{SS_{Ej}}$ and dfE_j are the error sum of squares and error d.f., respectively, under model f_j, $j = 1, 2$.

   For an α-level test of H_0 versus H_1, we compare this test statistics with the upper αth quantile of its distribution under H_0.

   Under H_0, $F \dot{\sim} F(\mathrm{dfE_1} - \mathrm{dfE_2}, \mathrm{dfE_2})$, so an approximate test has rejection rule: reject H_0 if

   $$F > F_{1-\alpha}(\mathrm{dfE_1} - \mathrm{dfE_2}, \mathrm{dfE_2}).$$

- Note that this test is a *likelihood ratio test.*

  Quite generally in a model with a parametric likelihood function $L(\boldsymbol{\gamma}; \mathbf{y})$ depending on parameter $\boldsymbol{\gamma}$ and data $\mathbf{y}$, under certain regularity conditions nested models can be tested by comparing the values of the likelihood when maximized under the competing models.

That is, we reject H_0 for large values of the ratio

$$\lambda \equiv \frac{L(\hat{\boldsymbol{\gamma}}; \mathbf{y})}{L(\hat{\boldsymbol{\gamma}}_0; \mathbf{y})}$$

where

$$\begin{aligned} \hat{\boldsymbol{\gamma}}_0 &= \text{MLE under } H_0 \text{ (under null, or partial model)} \\ \hat{\boldsymbol{\gamma}} &= \text{MLE under } H_1 \text{ (under alternative, or full model)} \end{aligned}$$

Logic: If the observed data are much less likely under H_0 than under H_1 then $\lambda >> 1$ and we should reject H_0.

In our spherical errors nonlinear regression model, suppose we wish to test

$$H_0 : \mathbf{A}\boldsymbol{\theta} = \mathbf{b} \qquad \text{versus} \qquad H_1 : \mathbf{A}\boldsymbol{\theta} \neq \mathbf{b}$$

where $\mathbf{A}$ is $k \times p$.

The likelihood function is

$$L(\boldsymbol{\theta}, \sigma^2; \mathbf{y}) = (2\pi\sigma^2)^{-n/2} \exp\left\{ -\frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{f}(\mathbf{X}, \boldsymbol{\theta})\|^2 \right\}$$

(cf. p.16).

Therefore, the LR is

$$\begin{aligned} \lambda &= \frac{L(\hat{\boldsymbol{\theta}}, \hat{\sigma}^2; \mathbf{y})}{L(\hat{\boldsymbol{\theta}}_0, \hat{\sigma}_0^2; \mathbf{y})} \\ &= \frac{(2\pi\hat{\sigma}^2)^{-n/2} \exp\left\{ -\frac{1}{2\hat{\sigma}^2} \|\mathbf{y} - \mathbf{f}(\mathbf{X}, \hat{\boldsymbol{\theta}})\|^2 \right\}}{(2\pi\hat{\sigma}_0^2)^{-n/2} \exp\left\{ -\frac{1}{2\hat{\sigma}_0^2} \|\mathbf{y} - \mathbf{f}(\mathbf{X}, \hat{\boldsymbol{\theta}}_0)\|^2 \right\}} \\ &= \frac{(2\pi\hat{\sigma}^2)^{-n/2} \exp\left\{ -\frac{1}{2\frac{1}{n}\|\mathbf{y} - \mathbf{f}(\mathbf{X}, \hat{\boldsymbol{\theta}})\|^2} \|\mathbf{y} - \mathbf{f}(\mathbf{X}, \hat{\boldsymbol{\theta}})\|^2 \right\}}{(2\pi\hat{\sigma}_0^2)^{-n/2} \exp\left\{ -\frac{1}{2\frac{1}{n}\|\mathbf{y} - \mathbf{f}(\mathbf{X}, \hat{\boldsymbol{\theta}}_0)\|^2} \|\mathbf{y} - \mathbf{f}(\mathbf{X}, \hat{\boldsymbol{\theta}}_0)\|^2 \right\}} \\ &= \left(\frac{\hat{\sigma}_0^2}{\hat{\sigma}^2} \right)^{n/2} = \left(\frac{\|\mathbf{y} - \mathbf{f}(\mathbf{X}, \hat{\boldsymbol{\theta}}_0)\|^2}{\|\mathbf{y} - \mathbf{f}(\mathbf{X}, \hat{\boldsymbol{\theta}})\|^2} \right)^{n/2} = \left(\frac{S(\hat{\boldsymbol{\theta}}_0)}{S(\hat{\boldsymbol{\theta}})} \right)^{n/2} \end{aligned}$$

We reject H_0 if λ is large compared to its distribution under H_0.

Equivalently, we can reject if some increasing function of λ is large compared with its distribution (which may be easier to calculate). In particular, we reject at level α if

$$(\lambda^{2/n} - 1)\frac{n-p}{k} = \frac{[S(\hat{\boldsymbol{\theta}}_0) - S(\hat{\boldsymbol{\theta}})]/k}{S(\hat{\boldsymbol{\theta}})/(n-p)} = F \dot{\sim} F(k, n-p),$$

which is the same test we presented before arguing simply by analogy.

- Transforming λ to a ratio of mean squares gives us a test statistic with an approximate F distribution. Alternatively, there is a famous result known as Wilks' Theorem that gives the asymptotic distribution of $2\log(\lambda)$ under quite general conditions. The results says

$$2\log(\lambda) \dot{\sim} \chi^2(\#\text{ of nonredundant restrictions on } \boldsymbol{\gamma} \text{ made by } H_0).$$

- Therefore, in our spherical errors nonlinear regression model, an alternative test of $H_0 : \mathbf{A}\boldsymbol{\theta} = \mathbf{b}$ is to reject H_0 at level α if

$$2\log(\lambda) > \chi^2_{1-\alpha}(k)$$

- This test is asymptotically equivalent to the F-test version of the LRT given above, but the F version of this test performs better than the chi-square version in small samples.

- In linear regression, the F test given above was exact, and was algebraically equivalent to the F test based on

$$\frac{(\mathbf{A}\hat{\boldsymbol{\beta}} - \mathbf{b})^T\{\mathbf{A}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{A}\}^{-1}(\mathbf{A}\hat{\boldsymbol{\beta}} - \mathbf{b})}{ks^2} \sim F(k, n-p). \qquad (*)$$

(cf. p.23).

- In the nonlinear regression context, (*) becomes

$$\frac{(\mathbf{A}\hat{\boldsymbol{\theta}} - \mathbf{b})^T\{\mathbf{A}(\hat{\mathbf{V}}^T\hat{\mathbf{V}})^{-1}\mathbf{A}\}^{-1}(\mathbf{A}\hat{\boldsymbol{\theta}} - \mathbf{b})}{ks^2} \dot{\sim} F(k, n-p). \qquad (**)$$

  However, the equivalence no longer holds!

- The statistic (**) is a special case of a *Wald test* rather than a LRT. It can still be used as an approximate F-test statistic. However, the validity of the Wald test is affected by parameter-effects nonlinearity and intrinsic nonlinearity whereas the LRT is affected only by intrinsic nonlinearity. This makes the LRT the preferred choice for testing nested models.

  What if the errors are heteroscedastic and/or correlated?

Consider the problem of testing

$$H_0 : \mathbf{A}\boldsymbol{\theta} = \mathbf{b} \quad vs. \quad H_1 : \mathbf{A}\boldsymbol{\theta} \neq \mathbf{b}$$

based on the (full) model

$$\mathbf{y} = \mathbf{f}(\mathbf{X}, \boldsymbol{\theta}) + \mathbf{e}, \quad \text{where } \text{var}(\mathbf{e}) = \sigma^2\Lambda(\boldsymbol{\lambda}).$$

- When testing such a hypothesis on the mean parameter $\boldsymbol{\theta}$ using a LRT, the variance-covariance model should be kept fixed!

The reason for this is that we do not want to confound our question concerning the mean structure with issues concerning the variance-covariance structure.

Q: Where do we fix the value of $\Lambda(\boldsymbol{\lambda})$?

A: At $\Lambda(\hat{\boldsymbol{\lambda}})$ where $\hat{\boldsymbol{\lambda}}$ is our best estimate of λ.

- $\hat{\boldsymbol{\lambda}}$ should be obtained from the full (alternative) model under consideration in the hypothesis test, or, better yet, the fullest model among all of those under consideration in the analysis.

- The degrees of freedom for the test is still given by k, because we are not fixing $\boldsymbol{\lambda}$ at a known value, but rather an estimated value. We still have just as many variance-covariance parameters to estimate in both the null and alternative models.

Example — High-Flux Hemodialyzer Ultrafiltration Rates

Vonesh and Carter (1992, *Biometrics*) describe and analyze data measured on high-flux hemodialyzers to assess their *in vivo* ultrafiltration characteristics. The ultrafiltration rates (in ml/hr) of 20 high-flux dialyzers were measured at 7 ascending transmembrane pressures (in dmHg). The *in vivo* evaluation of the dialyzers used bovine blood at flow rates of either 200 dl/min or 300 dl/min. These data are described in Appendix A.6 of Pinheiro and Bates (2000) and analyzed in their book in §5.2.2, §5.4, and §8.3.3. The data are included as the groupedData object Dialyzer in the nlme library for S-PLUS. We give a partial listing of the data below.

Obs. No.	Subject	Blood Flow Rate (QB)	Transmembrane Pressure	Ultrafiltration Rate	W/in Subject Index
1	1	200	0.240	0.645	1
2	1	200	0.505	20.115	2
3	1	200	0.995	38.460	3
4	1	200	1.485	44.985	4
5	1	200	2.020	51.765	5
6	1	200	2.495	46.575	6
7	1	200	2.970	40.815	7
8	2	200	0.240	3.720	1
9	2	200	0.540	18.885	2
⋮	⋮	⋮	⋮	⋮	⋮
139	20	300	2.510	53.625	6
140	20	300	3.000	56.430	7

The model we consider for these data is an asymptotic regression model with an offset (as in Appendix C.2 of Pinheiro and Bates (2000)). We model y = ultrafiltration rate as the following function of x =transmembrane pressure:

$$y_i = \theta_1\{1 - \exp[-e^{\theta_2}(x_i - \theta_3)]\} + e_i, \quad i = 1, \ldots, n. \qquad (*)$$

The parameters here have the folowing interpretations: θ_1 = the maximum ultrafiltration rate that can be obtained (the upper asymptote), θ_2 = the log of the hydrolic permeability transport rate; and θ_3 = the transmembrane pressure required to offset the oncotic pressure.

- See handout dialyzer1.
- Since Dialyzer is a groupedData object it is very easy to plot using the simple command, plot(Dialyzer, outer = $\sim$ QB). This yields plots of ultrafiltration rate (y) versus transmembrane pressure (x) separately by blood flow rate (QB=200 vs. QB=300). From this plot we can see the asymptotic form of the relationship with possibly different values of the parameters for the two QB groups.
- The nlme library contains a handy function nlsList() that allows one to fit separate nonlinear regressions within each of several groups using nls to fit each separate model. We use this function to fit separate models of the form (*) for each value of QB. We see that θ_1 and θ_2 appear to change across QB groups, but θ_3 does not.
- A useful technique to compare the $\boldsymbol{\theta}$ values across groups is to plot 95% intervals for each component of $\boldsymbol{\theta}$, separately by group. This is easily done by extracting the intervals from m1Dial.lis with the intervals() function and feeding that into the plot function. Clearly, there is little overlap in the intervals for θ_1 and θ_2, but considerable overlap for θ_3.
- These results suggest fitting a model with dummy variables to allow θ_1 and θ_2 to change with QB.

Dummy Variables:

Suppose we have grouped data y_{ij} where y_{ij} represents the j^{th} observation from the i^{th} group:

$$y_{ij}, \quad \begin{array}{l} i = 1, \ldots, a \\ j = 1, \ldots, n_i \end{array}$$

Suppose our model for y_{ij} involves a parameter γ, say, that we wish to change across groups. That is, we want the value of γ associated with y_{ij} to change with i but not j.

There are many different ways to choose a parameterization to accomplish this. Perhaps the simplest is to replace γ with

$$\gamma_i = \gamma_1 1_{\{i=1\}} + \gamma_2 1_{\{i=2\}} + \cdots + \gamma_a 1_{\{i=a\}}$$

where

$$1_{\{A\}} = \begin{cases} 1, & \text{if condition } A \text{ is true;} \\ 0, & \text{otherwise.} \end{cases}$$

- Here, γ_i has the interpretation as the γ-parameter associated with group i, $i = 1, \ldots, a$.

Alternatively, we can replace γ with

$$\gamma_1 + \phi_2 1_{\{i=2\}} + \phi_3 1_{\{i=3\}} + \cdots + \phi_a 1_{\{i=a\}}$$

- Here, the interpretations as follows:

$$\begin{aligned} \gamma_1 &= \text{the } \gamma \text{ parameter for group 1} \\ \gamma_1 + \phi_2 &= \text{the } \gamma \text{ parameter for group 2} \\ \gamma_1 + \phi_3 &= \text{the } \gamma \text{ parameter for group 3} \\ &\vdots \\ \gamma_1 + \phi_a &= \text{the } \gamma \text{ parameter for group } a \end{aligned}$$

  Hence, ϕ_i has the interpretation as the additive effect of being in group i versus being in group 1. This is an especially convenient parameterization when group 1 corresponds to a standard of comparison (e.g., a control group, or the standard treatment group).

A third option is to replace γ with

$$\gamma_0 + \gamma_i = \gamma_0 + \gamma_1 1_{\{i=1\}} + \gamma_2 1_{\{i=2\}} + \cdots + \gamma_a 1_{\{i=a\}},$$

where $\gamma_1, \gamma_2, \ldots, \gamma_a$ are constrained to sum to 0.

- Here, the interpretations are as follows:

$$\begin{aligned}
\gamma_0 &= \text{the ``average' } \gamma \text{ parameter across all groups} \\
\gamma_1 &= \text{the effect up or down from } \gamma_0 \text{ associated with group 1} \\
\gamma_2 &= \text{the effect up or down from } \gamma_0 \text{ associated with group 2} \\
&\vdots \\
\gamma_a &= \text{the effect up or down from } \gamma_0 \text{ associated with group } a
\end{aligned}$$

In this *ANOVA-type* parameterization, note that for $a = 2$ groups we can use the the constraint $\gamma_1 + \gamma_2 = 0$ to write $\gamma_2 = -\gamma_1$ so that γ becomes

$$\begin{aligned}
\gamma_0 + \gamma_1 1_{\{i=1\}} + \gamma_2 1_{\{i=2\}} &= \gamma_0 + \gamma_1 1_{\{i=1\}} - \gamma_1 1_{\{i=2\}} \\
&= \gamma_0 + \gamma_1 (1_{\{i=1\}} - 1_{\{i=2\}}).
\end{aligned}$$

and the hypothesis of equal γ values across groups can be tested by testing $H_0 : \gamma_1 = 0$.

While all three approaches can be generalized to > 1 grouping factor, the ANOVA coding is particularly convenient. Suppose now we have two 2-level grouping factors, F1, and F2, and we have data y_{ijk}, the k^{th} observation at the i^{th} level of F1 combined with the j^{th} level of F2.

E.g., suppose we have three replicates at each combination of F1 and F2:

y_{ijk}	F1 (i)	F2 (j)	Replicate (k)
y_{111}	1	1	1
y_{112}	1	1	2
y_{113}	1	1	3
y_{121}	1	2	1
y_{122}	1	2	2
y_{123}	1	2	3
y_{211}	2	1	1
y_{212}	2	1	2
y_{213}	2	1	3
y_{221}	2	2	1
y_{222}	2	2	2
y_{223}	2	2	3

Then we can let γ differ across the four groups by replacing γ by

$$\gamma_0 + \alpha_i + \beta_j + \gamma_{ij} \qquad (\dagger)$$

where we constrain the parameters so that $\sum_i \alpha_i = \sum_j \beta_j = \sum_i \gamma_{ij} = \sum_j \gamma_{ij} = 0$. By substituting these constraints into (†) it's not hard to see that this simplifies to

$$\begin{aligned}\gamma_0 &+ \alpha_1(1_{\{i=1\}} - 1_{\{i=2\}}) + \beta_1(1_{\{j=1\}} - 1_{\{j=2\}}) \\ &+ \gamma_{11}(1_{\{i=1,j=1\}} - 1_{\{i=1,j=2\}} - 1_{\{i=2,j=1\}} + 1_{\{i=2,j=2\}})\end{aligned}$$

and we can test main effects and interaction between F1 and F2 with the following hypotheses:

$$\begin{aligned} H_0 : \gamma_{11} = 0 \quad &\Rightarrow \quad \text{no interaction between F1 and F2} \\ H_0 : \alpha_1 = 0 \quad &\Rightarrow \quad \text{no main effect of F1} \\ H_0 : \beta_1 = 0 \quad &\Rightarrow \quad \text{no main effect of F2} \end{aligned}$$

Back to the example:

- For now we assume all three parameters $\theta_1, \theta_2, \theta_3$ differ across QB groups and we use the reference group-type parameterization to do this. That is, we make the substitutions

$$\begin{aligned}\theta_1 &= \phi_1 + \gamma_1 Q_i \\ \theta_2 &= \phi_2 + \gamma_2 Q_i \\ \theta_3 &= \phi_3 + \gamma_3 Q_i\end{aligned}$$

  where

$$Q_i = \begin{cases} 0, & \text{if observation } i \text{ comes from the QB=200 group; and} \\ 1, & \text{if observation } i \text{ comes from the QB=300 group; and} \end{cases}$$

  Thus, our model becomes

$$y_i = (\phi_1 + \gamma_1 Q_i)\{1 - \exp[-e^{\phi_2 + \gamma_2 Q_i}\{x_i - (\phi_3 + \gamma_3 Q_i)\}]\} + e_i.$$

  We fit this model as m2Dial.gnls.

- Since we suspect that θ_3 does not depend on group, we may want to test this hypothesis. To do so, we can use either a Wald type test of $H_0 : \gamma_3 = 0$ or a LRT. Before doing so though, we must consider the adequacy of the assumed variance-covariance structure.

- m2Dial.gnls assumes a spherical var-cov structure. A plot of the residuals vs fitteds and vs the values of the covariate x =transmembrane pressure, suggests heterogeneity. We consider variance functions:

$$g^2 = |\mu_i|^{2\delta} \quad \text{and} \quad g^2 = |x_i|^{2\delta}$$

  in models m3Dial.gnls and m4Dial.gnls, respectively. The latter fits slightly better according to AIC and BIC, but either could be used.

- Next we examine the within-subject autocorrelation. Data were collected on increasing $x-$values that were increased over 7 consecutive equally-spaced measurement times. This may induce some serial correlation within subject, and we do indeed see evidence of strong autocorrelation in the ACF plot for model m4Dial.gnls.

- An AR(1) model within-subject is fit to the data in m5Dial.gnls. A LRT test comparing the models with and without autocorrelation (m5 and m4, respectively) indicates that the AR(1) model fits substantially better than the independence model. In addition, the residual ACF from model m5 indicates that the AR(1) model is sufficient.

- To test equal θ_3-values across QB groups we now test $H_0 : \gamma_3 = 0$. The Wald test of this hypothesis is given in the summary of model m5Dial.gnls as the t-statistic for γ_3: $t = 1.25$ ($p = .2131$).

- For the LRT, we drop γ_3 from the model and refit, fixing the heterogeneity and autocorrelation parameters at their estimated values from the full model m5Dial.gnls. This is done with the fixed= options in varPower() and corAR1(). The test statistic is then equal to 2[logLik(m5Dial.gnls)-logLik(m6Dial.gnls)]=1.6227 which we compare with a χ^2 distribution with $k = 1$ d.f. (We're testing 1 restriction: $\gamma_3 = 0$.) Note that the d.f. and p-value from the anova() function are incorrect here.

- Notice that the p-value of .2027 from this LRT is very close to the Wald test result, $p = .2131$. The Wald test is substantially easier to implement, but can be affected by parameter-effects nonlinearity. Thus there are pros and cons for the Wald vs. LRT approaches, but the LRT is generally preferrable on statistical grounds.

2. Non-nested Models.

Choosing between competing non-nested models using formal means such as hypothesis tests is a hard problem. See Seber & Wild (1989, §5.9.6) for a brief discussion of some approaches that have been used and some references. We take a less formal approach to the problem based on the following considerations:

i. Theory — any model suggested by theory should have some precedence.

ii. Parsimony — simpler models and/or models with fewer parameters should be favored

iii. Analysis of residuals — models with "more random/unstructured" patterns in the residual plots should be favored.

iv. Curvature — models with low parameter-effects nonlinearity should be preferred.

v. Model selection criteria — also useful are information/model selection criteria such as AIC and BIC. However, one should be aware of some misuses and caveats:

- AIC or BIC are not comparable across models that have different response variables. In particular, one cannot use information criteria to compare a model with response variable y to a model with response variable $g(y)$ (e.g., $\log(y)$ or any other transformation of y).

- Information criteria are not comparable across models involving different data sets. This may seem obvious, but this mistake is often made especially when small differences in the data set are present that the analyst may not realize.

  - E.g., model 1 might involve x_1, and model 2 might involve x_1 and x_2. If missing values exist on x_2 these data will often be omitted from the data to which model 2 is fit automatically by the software. In such a situation information criteria cannot be used to compare the models because they are fit to different data sets.

- There is some disagreement about how to define AIC and, especially, BIC in situations in which the data are not all independent.

  - In particular, in longitudinal data and other clustered data settings, it is not so clear whether the penalty term for BIC should involve n, the total number of observations (which are not all independent in this context), or the number of independent subjects (clusters) in the data set, or some intermediate quantity.

  - E.g., in mixed model software such as PROCs MIXED and NLMIXED in SAS, the penalty involves the number of levels of the "outermost" random effect (in a clusterd data context this will typically be the number of clusters). So, in two models fit to the same data set, one with cluster-specific random effects and one without, the penalty terms for BIC will differ and the criteria will not be comparable.

To choose among non-nested models, some combination of (i)–(v) should be used along with the judgement/experience of the analyst.

Example — Rabbit Eyes Again

Recall that previously we fit these data using the model

$$y_i = \theta_1 - \frac{\theta_2}{\theta_3 + x_i} + e_i, \quad i = 1, \ldots, n, \tag{1}$$

where $e_1, \ldots, e_n$ are i.i.d. with $\mathrm{E}(e_i) = 0$, $\mathrm{var}(e_i) = \sigma^2$ and

$$y = \log(\text{eye lens weight})$$
$$x = \text{age in days}$$

Since these data exhibit an apparently asymptotic form, we might also consider the asymptotic regression model as an alternative model. Here we parameterize it as in SSasymp, the self-starting version of the asymptotic regression model provided in the nlme software:

$$y_i = \theta_1 + (\theta_2 - \theta_1)\exp[-e^{\theta_3} x_i] + e_i, i = 1, \ldots, n, \tag{2}$$

with the same assumptions on the errors.

- See handout rabbit3.
- In rabbit3.R we refit model 1 (previously fit in rabbit1.R) as m1rabbit.nls and we fit model 2 as m2rabbit.nls. Before examining the fitted models, we should first determine whether one of these models has a theoretical motivation that would give it precedence. Model 1 is based on the model originally proposed by Dudzinski and Mykytowycz (1961). It would require going back to the original paper to determine whether Dudzinski and Mykytowycz's original model was a mechanistic one. We assume for illustration purposes that it was not.
- Since both models are 3-parameter models, neither is more parsimonious.

- In the first page of plots in rabbit3, the fitted curves for models 1 and 2 are displayed. In addition, on the second page of plots there are plots of the residuals versus fitteds for both models and residuals versus the covariate Age for both models.

- Notice that model 2 does not appear to fit the data as well near the elbow in the curve and for large values of Age. This poor fit is especially obvious in the residual plots for model 2. In the Residuals vs. Age plots, the homoscedasticity assumption appears to be violated in both models, with variance apparently decreasing with Age.

- To deal with this heteroscedasticity, we refit these models with gnls() and then add in heteroscedasticity of the form

$$\text{var}(e_i) = \sigma^2 |\text{Age}_i|^{2\delta} \qquad (*)$$

  using the varPower(form $= \sim$ Age) specification. The heteroscedastic versions of models 1 and 2 are m1arabbit.gnls and m2arabbit.gnls, respectively, and based on the LRTs and information criteria produced by the anova() function, these models fit substantially better than their homoscedastic counterparts.

- On the final page of plots in rabbit3, we reproduce the residual plots for the models with heteroscedasticity. Now model (1) appears to fit well, but model (2) still shows substantial misspecification in the expectation function.

- On this basis we prefer model (1) with heteroscedastic errors as in (*). Note that δ is estimated as $\hat{\delta} = -.266$ indicating that the error standard deviation decreases with Age, as expected.

Models Defined by Systems of Differential Equations

- Read Ch. 5 of Bates & Watts (handout). See also Ch. 8 of Seber & Wild.

An important and large subclass of nonlinear models occurs when the response is described by a system of ordinary differential equations. These models are used in a wide variety of fields, but one important area of application is pharmacokinetics, where they are called **compartment models.**

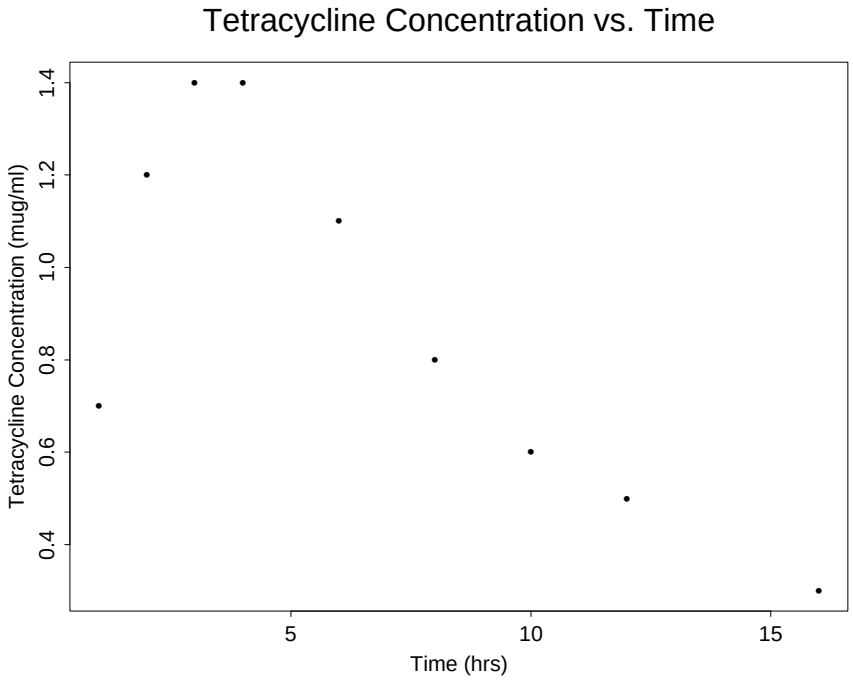
- These models were introduced briefly on pp. 54–57 of the notes and we recall some of that material now:

- Compartmental models are mechanistic models where one or more measurements of some physical process is related to time, inputs to the system, and other explanatory variables through a compartmental system.

- A compartmental system is "a system which is made up of a finite number of macroscopic subsystems, called compartments or pools, each of which is homogeneous and well mixed, and the compartments interact by exchanging materials. There may be inputs from the environment into one or more of the compartments, and there may be outputs (excretion) from one or more the compartments to the environment." (Seber & Wild, p.367)

- Compartmental models are common in chemical kinetics, toxicology, hydrology, geology, and pharmacokinetics.

As an example from pharmacokinetics, consider the data in the following scatterplot on tetracycline concentration over time.



The data come from a study in which a tetracycline compound was administered to a subject orally, and the concentration of tetracycline hydrochloride in the blood serum was measured over a period of 16 hours (the data are in Appendix A1.14 of our text).

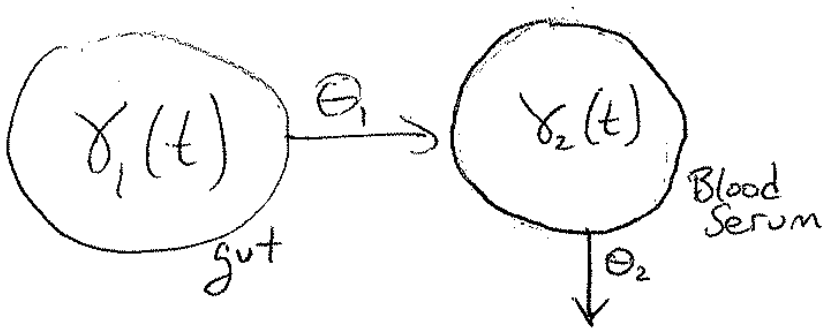
A simple compartmental model for the biological system determining tetracycline concentration in serum is one that hypothesizes

a. a gut compartment into which the chemical is introduced,

b. a blood compartment which absorbs the chemicals from the gut, and

c. an elimination path.

This simple two-compartment open model can be represented in a compartment diagram as follows:



Here, γ_1 and γ_2 represent the concentrations of the chemical in compartments 1 and 2, respectively, and θ_1 and θ_2 represents rates of transfer into and out of compartment 2, respectively.

- The model above is an example of an *open* compartment model. Compartment models with no interchange with the environment are said to be *closed*; otherwise they are open.

Under the assumption of *first-order (linear) kinetics*, it is assumed that at time t, the rate of elimination from any compartment is proportional to $\gamma(t)$, the concentration currently in that compartment.

Thus the rates of change in the concentrations in the two compartments in the model represented above are

$$\dot{\gamma}_1 \equiv \frac{\partial \gamma_1(t)}{\partial t} = -\theta_1 \gamma_1(t)$$
$$\dot{\gamma}_2 \equiv \frac{\partial \gamma_2(t)}{\partial t} = \theta_1 \gamma_1(t) - \theta_2 \gamma_2(t)$$

Here, the dot denotes differentiation with respect to time.

- We will restrict attention to first-order or linear compartment models.
- Another restriction of our scope is to *ordinary* differential equations. In particular, we exclude models based on systems of partial differential equations.

Differential equations solutions for linear compartmental models generally take the form of linear combinations of exponentials. Therefore, these models are nonlinear models that can be fit using methods similar to those used for yield-density models, growth curve models, etc.

- E.g., the biexponential model that we've worked with several times already comes up often in the analysis of compartment models.

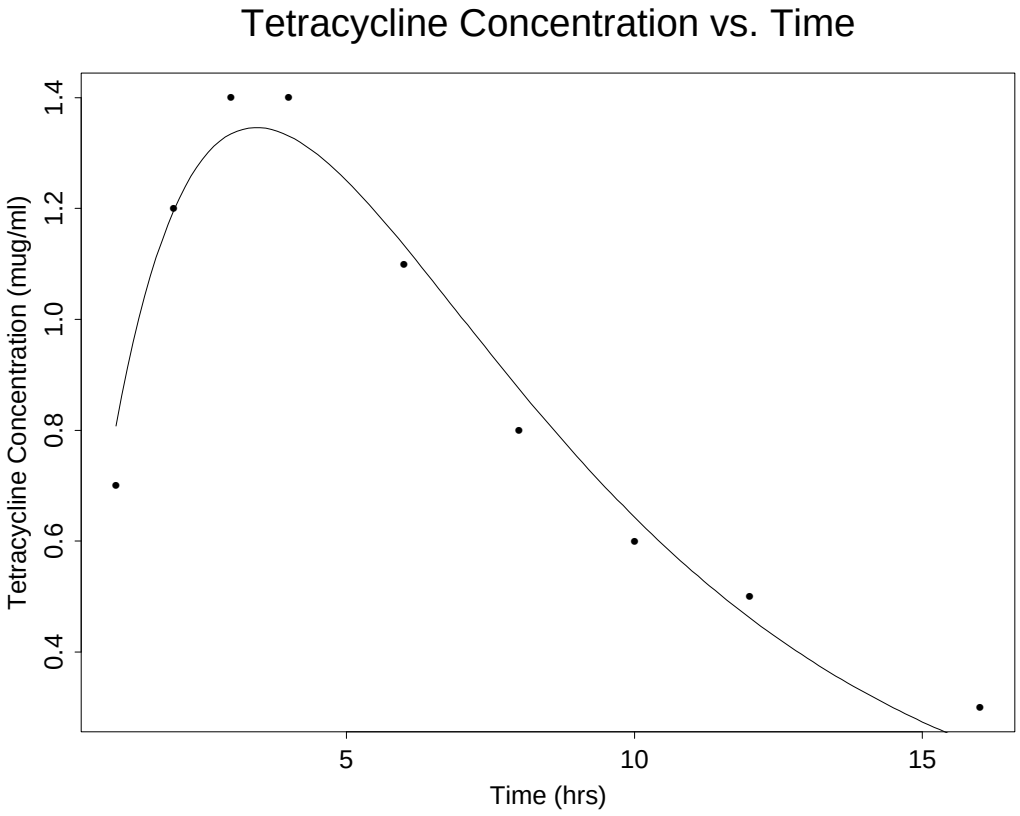
For example, under the assumptions that at time 0 $\gamma_1(0) = \theta_3$ and $\gamma_2(0) = 0$, the solution for $\gamma_2(t)$, the concentration in blood serum at time t is

$$\gamma_2(t) = \frac{\theta_3\theta_1(e^{-\theta_1 t} - e^{-\theta_2 t})}{\theta_2 - \theta_1}.$$

Therefore, we might try the additive error model

$$y_i = \frac{\theta_3\theta_1(e^{-\theta_1 t_i} - e^{-\theta_2 t_i})}{\theta_2 - \theta_1} + e_i, \quad i = 1, \ldots, n,$$

to model the tetracycline data. The resulting fitted regression curve is displayed below.



In the general compartment model with K compartments we write the concentrations at time t as

$$\boldsymbol{\gamma}(t) = \begin{pmatrix} \gamma_1(t) \\ \gamma_2(t) \\ \vdots \\ \gamma_K(t) \end{pmatrix}.$$

Assuming first-order kinetics with rate constants $\theta_1, \theta_2, \ldots, \theta_p$, the concentrations obey the linear system of differential equations

$$\frac{\partial \boldsymbol{\gamma}}{\partial t} = \dot{\boldsymbol{\gamma}}(t) = \mathbf{A}\boldsymbol{\gamma}(t) + \mathbf{i}(t)$$

where $\mathbf{A}$ is a $K \times K$ matrix known as the *transfer matrix* and $\mathbf{i}(t)$ is a $K \times 1$ vector-valued function of time representing inputs to the system.

- $\mathbf{A}$ contains the rate constants (elements of $\boldsymbol{\theta}$) and is determined by the model specification (assumed form of the model — how many compartments and how they exchange material).

- Input into the system at time t, $\mathbf{i}(t)$, is often assumed to be of the form

$$\mathbf{i}(t) = \begin{cases} \mathbf{i}, & \text{if } t \geq 0 \text{ (constant in time); and} \\ \mathbf{0}, & \text{if } t < 0. \end{cases}$$

  Such an input function specifies continuous infusion of material if t is continuous and step input if time is indexed discretely ($t = 0, 1, 2, 3, \ldots$).

- Another common input specification is a *bolus* or instantaneous injection of material. In that case $\mathbf{i}(t)$ can be replaced by a vector of initial conditions

$$\boldsymbol{\gamma}(0) = \boldsymbol{\gamma}_0.$$

Tetracycline Example:

In the tetracycline example, we can write the model-defining differential equations as

$$\underbrace{\begin{pmatrix} \dot{\gamma}_i(t) \\ \dot{\gamma}_2(t) \end{pmatrix}}_{=\dot{\boldsymbol{\gamma}}(t)} = \underbrace{\begin{pmatrix} -\theta_1 & 0 \\ \theta_1 & -\theta_2 \end{pmatrix}}_{=\mathbf{A}} \underbrace{\begin{pmatrix} \gamma_1(t) \\ \gamma_2(t) \end{pmatrix}}_{=\boldsymbol{\gamma}(t)}, \qquad t > 0$$

and

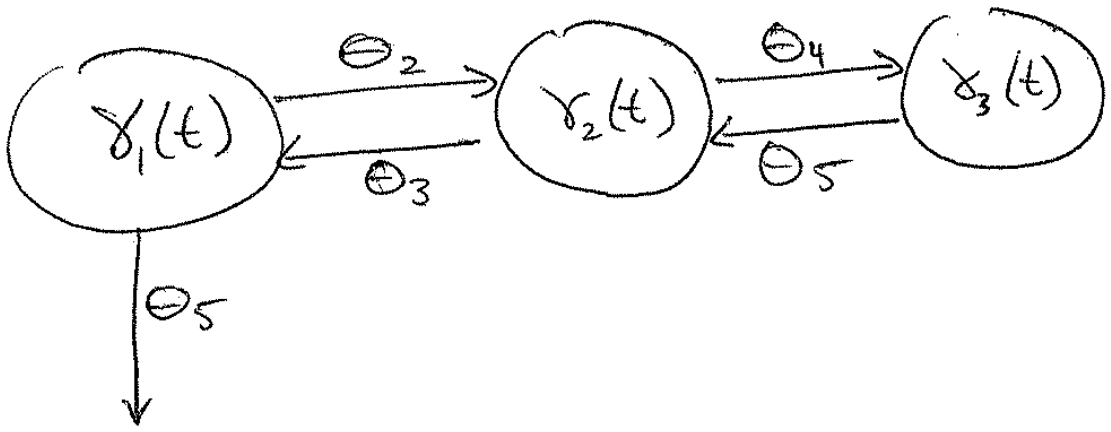
$$\boldsymbol{\gamma}(t) = \boldsymbol{\gamma}_0 = \begin{pmatrix} \theta_3 \\ 0 \end{pmatrix}, \qquad t = 0.$$

Another Example — Brunhilda Data

Measurements were taken on the radioactivity of blood samples taken from a baboon named Brunhilda at a number of specified times after a bolus injection of radiactive sulfate. The data are given below:

Time (min)	Count	Time (min)	Count	Time (min)	Count
2	151117	25	70593	90	53915
4	113601	30	67041	110	50938
6	97652	40	64313	130	48717
8	90935	50	61554	150	45996
10	84820	60	59940	160	44968
15	76891	70	57698	170	43602
20	73342	80	56440	180	42668

We consider a three-compartment open model of the following form for these data:



The measurements are treated as coming from compartment 1 and the bolus injection was taken as going into compartment 1.

The linear system defining the model is

$$\begin{aligned}
\dot{\gamma}_1 &= -(\theta_1 + \theta_2)\gamma_1(t) + \theta_3\gamma_2(t) \\
\dot{\gamma}_2 &= \theta_2\gamma_1(t) - (\theta_3 + \theta_4)\gamma_2(t) + \theta_5\gamma_3(t) \\
\dot{\gamma}_3 &= \theta_4\gamma_2(t) - \theta_5\gamma_3(t)
\end{aligned}$$

subject to $\boldsymbol{\gamma}(0) = \boldsymbol{\gamma}_0 = (\theta_6, 0, 0)^T$.

Here,

$$\mathbf{A} = \begin{pmatrix} -(\theta_1 + \theta_2) & \theta_3 & 0 \\ \theta_2 & -(\theta_3 + \theta_4) & \theta_5 \\ 0 & \theta_4 & -\theta_5 \end{pmatrix}.$$

Estimation in Compartment Models

There are several approaches to estimating parameters in compartment models:

1. Most obvious is to obtain the analytic solution to the system of differential equations and use that as the expectation function in an ordinary NLS fitting routine.

   – A drawback is that it is often difficult and sometimes impossible to derive closed form expressions for the expectation function and its derivatives with respect to the parameters.

2. Again we can use on ordinary NLS fitting routine but now with expectation function which is calculated numerically by solving the differential equations with quadrature (numerical integration).

3. A method of historical interest is as follows: A K-compartment model generally has a solution to its differential equations that takes the form of a sum of exponentials where the coefficients and exponents are functions of the rate parameters $\theta_1, \theta_2, \ldots, \theta_K$. Therefore, fit a generic sum-of-exponentials model of the form

$$\gamma_j(t) = \sum_{k=1}^{K} \beta_k e^{\lambda_k t} + e \qquad (*)$$

   then use the relationship between the exponential parameters $(\boldsymbol{\beta}, \boldsymbol{\lambda})$ and the system parameters $\boldsymbol{\theta}$ to solve for $\hat{\boldsymbol{\theta}}$ from $(\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\lambda}})$.

   – Since we need the analytic solution of the differential equations to know the relationship between $\boldsymbol{\theta}$ and $(\boldsymbol{\beta}, \boldsymbol{\lambda})$, it would seem that we would always prefer method (1) to this method. This is true if (*) were being fit by NLS because it would be just as easy to use NLS with (*) parameterized in terms of $\boldsymbol{\theta}$. However, historically (*) was fit using the method of exponential peeling (a method adequate for starting values, but pretty crude as a method of estimation), so (*) could be fit easily in the $(\boldsymbol{\beta}, \boldsymbol{\lambda})$-parameterization but not in the $\boldsymbol{\theta}$-parameterization.

- In addition to the crudeness of estimation of (*) with peeling, another problem is that there may be fewer system parameters in $\boldsymbol{\theta}$ than exponential parameters in $(\boldsymbol{\beta}, \boldsymbol{\lambda})$ so that it may not be possible to solve for $\boldsymbol{\theta}$.

- This approach is obsolete.

4. A fourth method, known as the *matrix exponential* approach, is very useful because it does not require an analytic solution to the system of differential equations and it computes the expectation function and its derivatives in a unified, efficient manner.

The Matrix Exponential Method:

The general solution to the linear differential equation system $\dot{\boldsymbol{\gamma}}(t) = \mathbf{A}\boldsymbol{\gamma}(t) + \mathbf{i}(t)$ is given by

$$\boldsymbol{\gamma}(t) = e^{\mathbf{A}t}\boldsymbol{\gamma}_0 + e^{\mathbf{A}t} * \mathbf{i}(t)$$

where the matrix exponential $e^{\mathbf{A}t}$ represents the convergent power series

$$e^{\mathbf{A}t} = \mathbf{I} + \frac{\mathbf{A}t}{1!} + \frac{(\mathbf{A}t)^2}{2!} + \cdots$$

and $*$ denoted the convolution,

$$e^{\mathbf{A}t} * \mathbf{i}(t) = \int_0^t e^{\mathbf{A}(t-u)}\mathbf{i}(u)du,$$

where the integration is performed componentwise.

- Therefore, if we can evaluate the matrix exponential $e^{\mathbf{A}t}$ and the convolution integral $e^{\mathbf{A}t} * \mathbf{i}(t)$, we can evaluation the expectation function of the model.

- Note that it is rarely useful to sum the power series representation of $e^{\mathbf{A}t}$ to compute it. In addition, the matrix convolution $e^{\mathbf{A}t} * \mathbf{i}(t)$ can be reduced to easier-to-compute scalar convolutions. Both computations are simplified by using the *spectral decomposition* of $\mathbf{A}$.

Suppose it is possible to decompose $\mathbf{A}$ as

$$\mathbf{A} = \mathbf{U}\Lambda\mathbf{U}^{-1}$$

where $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_K)$ is a diagonal matrix containing the eigenvalues of $\mathbf{A}$, and $\mathbf{U}$ contains as its columns the eigenvectors of $\mathbf{A}$. Then we can write

$$\begin{aligned} e^{\mathbf{A}t} &= \mathbf{I} + \frac{\mathbf{A}t}{1!} + \frac{(\mathbf{A}t)^2}{2!} + \cdots \\ &= \mathbf{I} + \frac{\mathbf{U}\Lambda\mathbf{U}^{-1}t}{1!} + \frac{(\mathbf{U}\Lambda\mathbf{U}^{-1}t)^2}{2!} + \cdots \\ &= \mathbf{U}\underbrace{(\mathbf{I} + \frac{\Lambda t}{1!} + \frac{(\Lambda t)^2}{2!} + \cdots)}_{\equiv \mathbf{M}}\mathbf{U}^{-1} \end{aligned}$$

where $\mathbf{M}$ has $(i,j)^{\text{th}}$ element given by

$$M_{ij} = \begin{cases} 0 & \text{if } i \neq j; \text{ and} \\ \underbrace{1 + \frac{\lambda_i t}{1!} + \frac{(\lambda_i t)^2}{2!} + \cdots}_{=e^{\lambda_i t}} & \text{if } i = j. \end{cases}$$

Therefore,

$$e^{\mathbf{A}t} = \mathbf{U}e^{\Lambda t}\mathbf{U}^{-1} \qquad (\dagger)$$

where

$$e^{\Lambda t} = \text{diag}(e^{\lambda_1 t}, \ldots, e^{\lambda_K t}).$$

- By using the spectral composition, we've reduce the matrix exponential to a simple matrix multiplication given by (†).

For a bolus input, this is all we really need, because the convolution integral $e^{\mathbf{A}t} * \mathbf{i}(t)$ drops out of the solution. That is, for a bolus input, the system of differential equations defining the model can be written

$$\begin{aligned} \dot{\boldsymbol{\gamma}}(t) &= \mathbf{A}\boldsymbol{\gamma}(t), \qquad \text{for } t > 0 \\ \boldsymbol{\gamma}(0) &= \boldsymbol{\gamma}_0, \qquad \text{for } t \leq 0. \end{aligned}$$

And this system of equations has solution

$$\boldsymbol{\gamma}(t) = e^{\mathbf{A}t}\boldsymbol{\gamma}_0 = \mathbf{U}e^{\Lambda t}\mathbf{U}^{-1}\boldsymbol{\gamma}_0.$$

For other types of input we need to evaluate the convolution integral $e^{\mathbf{A}t} * \mathbf{i}(t)$. Again, using the spectral decomposition of $\mathbf{A}$ we have

$$\begin{aligned} e^{\mathbf{A}t} * \mathbf{i}(t) &= \int_0^t e^{\mathbf{A}(t-u)}\mathbf{i}(u)du \\ &= \int_0^t \mathbf{U}e^{\Lambda(t-u)}\mathbf{U}^{-1}\mathbf{i}(u)du \\ &= \mathbf{U}\int_0^t e^{\Lambda(t-u)}\boldsymbol{\kappa}(u)du \\ &= \mathbf{U}\left[e^{\Lambda t} * \boldsymbol{\kappa}(t)\right], \end{aligned}$$

where $\boldsymbol{\kappa}(t) = \mathbf{U}^{-1}\mathbf{i}(t)$.

Thus if we define $\boldsymbol{\zeta}(t) = \mathbf{U}^{-1}\boldsymbol{\gamma}(t)$ we have

$$\begin{aligned} \boldsymbol{\zeta}(t) &= \mathbf{U}^{-1}\left(e^{\mathbf{A}t}\boldsymbol{\gamma}_0 + e^{\mathbf{A}t} * \mathbf{i}(t)\right) \\ &= e^{\Lambda t}\boldsymbol{\zeta}_0 + e^{\Lambda t} * \boldsymbol{\kappa}(t) \end{aligned}$$

where $\boldsymbol{\zeta}_0 = \mathbf{U}^{-1}\boldsymbol{\gamma}_0$.

In the case of continuous infusion/step input (recall this is when $\mathbf{i}(t) = \mathbf{i}$, $t \geq 0$), we have $\boldsymbol{\kappa}(t) = \boldsymbol{\kappa} = \mathbf{U}^{-1}\mathbf{i}$, and the convolution integral becomes

$$e^{\Lambda t} * \boldsymbol{\kappa}$$

which evaluates to a vector with i^{th} element

$$\{e^{\Lambda t} * \boldsymbol{\kappa}\}_i = \begin{cases} \kappa_i \frac{e^{\lambda_i t} - 1}{\lambda_i}, & \text{if } \lambda_i \neq 0; \text{ and} \\ \kappa_i t & \text{if } \lambda_i = 0. \end{cases}$$

- This gives us all we need to calculate the solution the system of differential equations defining the model (that is, to calculate the expectation function of our model) in an efficient manner.

- Note that this approach assumes that a spectral decomposition exists for $\mathbf{A}$ and that the eigenvalues of $\mathbf{A}$ are all real. These conditions do not always hold, and Bates and Watts give generalizations of this procedure to cover those situations in Appendix A5 of their text.

How about computing derivatives of the expectation function with respect to the parameters $\theta_1, \ldots, \theta_p$ for the G-N method?

These can be calculated using the same tools. But before we describe how that works, we need to talk about *dead time* because it can add a parameter to our model that we want to account for.

Dead time:

Often there is a delay or lag between time 0 when the system is initialized (e.g., the bolus dose is given) and when the system reacts (e.g, before drug concentration changes from its initial value). This delay is called dead time, and can be built into our model as an extra parameter to be estimated.

We denote the time at which the system reacts as t_0. In the presence of dead time, $t_0 > 0$ and the model-defining system of differential equations becomes

$$\begin{aligned} \dot{\boldsymbol{\gamma}}(\tau) &= \mathbf{A}\boldsymbol{\gamma}(\tau) + \mathbf{i}(\tau), \qquad \text{if } \tau > 0 \\ \boldsymbol{\gamma}(\tau) &= \boldsymbol{\gamma}_0, \qquad \text{if } \tau = 0, \end{aligned} \tag{$*$}$$

where

$$\tau = \begin{cases} t - t_0, & \text{if } t \geq t_0; \text{ and} \\ 0, & \text{otherwise.} \end{cases}$$

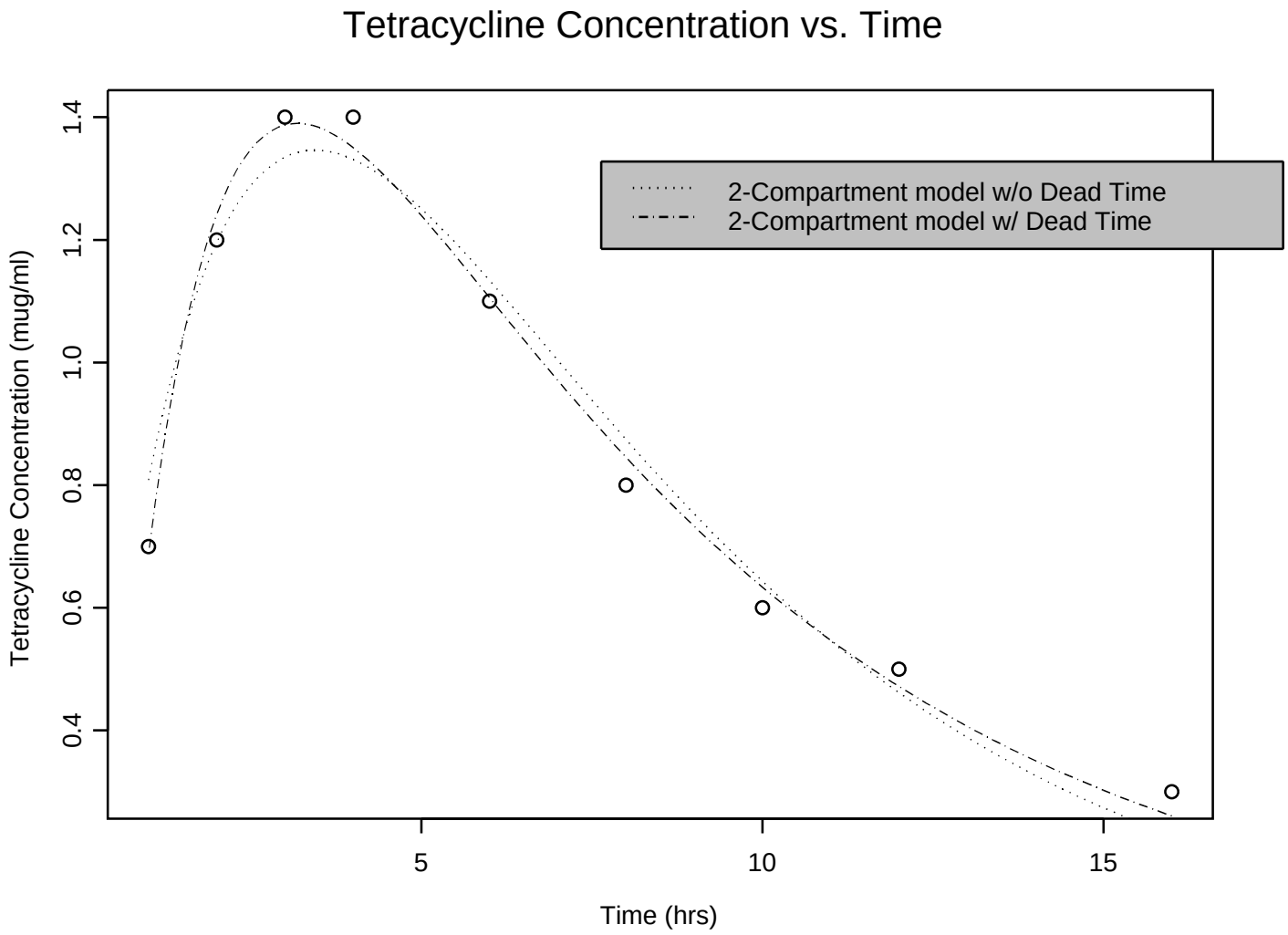
- Here t_0 can be assumed known or treated as an unknown parameter to be estimated.

With dead time, the general solution to (*) becomes

$$\boldsymbol{\gamma}(\tau) = \begin{cases} \boldsymbol{\gamma}_0, & \text{if } \tau = 0; \text{ and} \\ e^{\mathbf{A}\tau}\boldsymbol{\gamma}_0 + e^{\mathbf{A}\tau} * \mathbf{i}(\tau), & \text{if } \tau > 0. \end{cases} \tag{$**$}$$

- Evaluation of the expectation function of the model goes through essentially as before, but with t replaced by τ.

To illustrate how dead time can be useful, we refit the two-compartment model of p.181 to the tetracycline data with an unknown dead time parameter t_0 added to the model. The resulting fitted curve fits the data considerably better than the model without dead time:



Back to computing derivatives of the expectation function:

Notation: we will denote a derivative with respect to θ_j by a (j)-subscipt. E.g., the derivatives we are after are

$$\boldsymbol{\gamma}_{(j)}(\tau) \equiv \frac{\partial \boldsymbol{\gamma}(\tau)}{\partial \theta_j}, \qquad j = 1, \ldots, p.$$

In the general solution given by (**), $\boldsymbol{\gamma}(\tau)$ depends on $\boldsymbol{\gamma}_0$, $\mathbf{A}$, τ, and $\mathbf{i}$. For any given paramater θ_j for which we seek the derivative $\boldsymbol{\gamma}_{(j)}$, one, some or all of these quantities may depend upon θ_j. Therefore, we consider the formula for $\boldsymbol{\gamma}_{(j)}$ by cases.

Case 1: τ depends on θ_j; $\boldsymbol{\gamma}_0$, $\mathbf{A}$, and $\mathbf{i}$ do not.

In this case we use the chain rule. By the chain rule,

$$\begin{aligned}
\boldsymbol{\gamma}_{(j)}(\tau) &\equiv \frac{\partial \boldsymbol{\gamma}(\tau)}{\partial \theta_j} \\
&= \frac{\partial \boldsymbol{\gamma}(\tau)}{\partial \tau} \frac{\partial \tau}{\partial \theta_j} \\
&= \dot{\boldsymbol{\gamma}}(\tau)\tau_{(j)} = \tau_{(j)}[\mathbf{A}\boldsymbol{\gamma}(\tau) + \mathbf{i}(\tau)]
\end{aligned}$$

Case 2: Suppose $\mathbf{A}$, $\boldsymbol{\gamma}_0$, and/or $\mathbf{i}$ depend on θ_j, but τ does not. Then we can differentiate (*) to obtain

$$\dot{\boldsymbol{\gamma}}_{(j)}(\tau) = \mathbf{A}\boldsymbol{\gamma}_{(j)}(\tau) + \mathbf{A}_{(j)}\boldsymbol{\gamma}(\tau) + \mathbf{i}_{(j)} \qquad (\ddagger)$$

Since we're trying to compute $\boldsymbol{\gamma}_{(j)}(\tau)$ not $\dot{\boldsymbol{\gamma}}_{(j)}(\tau)$, (‡) does not directly give us what we seek.

However, (‡) does give us a differential equation whose solution is what we seek: $\boldsymbol{\gamma}_{(j)}(\tau)$.