Fitting ecological dynamic models to data

This set of lectures aims to provide an introduction to fitting and inference with population dynamic models. The basic motivation is that often we know enough about a biological system to formulate models describing how we think it works, but not quite enough to pin down the precise numerical values of all the parameters (or in some cases functions) that it contains. In this circumstance it's useful to be able to make inferences about the unknown parts of the model by fitting to data.

The structure of the lectures is as follows. I'll start by reviewing the basic theory of linear statistical models, and then give an example of how this theory can be used directly to fit simple structured population models of the type popularised by Caswell. This example highlights some interesting general problems. Having proposed solutions to these I'll then proceed to develop a slightly more interesting continuous counterpart of the matrix model approach, to further illustrate the applications of strictly linear theory. The final part generalizes further to deal with the non-linear models that usually arise when modelling on the basis of biological mechanisms.

1 Linear modelling

Most of the model fitting that I'll talk about in this introductory series of lectures is based in some way on the statistical theory of linear models - even when dealing with non-linear models later on, I'll still use approximating linear models to actually facilitate fitting and inference. It's therefore important to start with a solid understanding of the basic theory of linear modelling. Accordingly I'll start with the simplest linear model that can usefully illustrate the theory and then generalize.

1.1 A simple linear model

The simplest "interesting" linear model relates a *response variable*, Y_i, to a *predictor variable* x_i like this:

$$Y_i = \beta x_i + \epsilon_i$$

where β is a *parameter* describing the way that Y_i is related to x_i and ϵ_i is a random "error" term - the part of Y_i that can't be explained by x_i. The ϵ_i's are modelled as mutually independent[1] random variables, all with mean zero and the same variance σ^2. *i.e.* $E[\epsilon_i] = 0$, $\text{var}[\epsilon_i] = \sigma^2$. The model is an example of a **linear model** (or linear regression model) because the r.v.'s ϵ_i and the parameter β enter the model in a linear way.

Typically we would like to estimate β (and possibly σ^2) using a set of observations of Y_i, x_i: i.e. a set of observations y_i, x_i. Note the distinction between Y_i and an observation of it y_i. Y_i is a *random variable*, so each time I observe it I will expect to get a different answer (even for fixed x_i), y_i is just one particular observation of Y_i.

To estimate β we can use the method of *least squares.* We simply seek the value of β that minimises the mean square difference between βx_i and y_i (averaged across all i). i.e. we minimise Q given by:

$$Q = \sum (y_i - \beta x_i)^2$$

Note that is we define the *residuals* as $e_i = y_i - \beta x_i$, then we are choosing β to minimise the sum of squared residuals:

$$Q = \sum e_i^2$$

It's straightforward to see that Q is quadratic in β: if you plot Q against β for any particular dataset you'll get a quadratic plot, with a minimum at the least squares estimate of β. To find this point algebraically we just seek the turning point of Q, i.e. the point at which its gradient w.r.t. β is zero:

$$\frac{\partial Q}{\partial \beta} = \sum -2x_i(y_i - \beta x_i)$$

[1]Recall that mutual independence of the ϵ_i's means that, for $i \neq j$, knowing ϵ_i doesn't give you any information about the value that ϵ_j might have.

Setting this to zero gives the equation that must be solved to find the estimate of β:

$$\sum -2x_i(y_i - \hat{\beta}x_i) = 0 \Rightarrow \sum y_i x_i - \sum \hat{\beta}x_i^2 = 0$$

so

$$\hat{\beta} = (\sum x_i^2)^{-1} \sum x_i y_i.$$

An estimate on its own is little use unless we have some idea of how reliable it is. The key point here is that the data y_i, x_i only represent one realisation of the process being modelled: if we gathered new data, we'd expect a somewhat different y_i for each x_i. Since we use the y_i's to calculate $\hat{\beta}$, this must mean that we'd get a different β estimate every time we repeated the data gathering process. To avoid treating any particular $\hat{\beta}$ as being more certain than is implied by this spread of values that it *could* have taken, it would be good to calculate the variance of $\hat{\beta}$.

To work out the variance of $\hat{\beta}$ we need to treat it as a random variable, so we write it as a function of the Y_i, rather than particular observations y_i:

$$\hat{\beta} = (\sum x_i^2)^{-1} \sum x_i Y_i$$

and then re-write slightly again:

$$\hat{\beta} = \sum a_i Y_i \quad \text{where} \quad a_i = x_i(\sum x_i^2)^{-1}.$$

Now the only random component of the Y_i's are the ϵ_i's, so the Y_i's must also be independent with variance σ^2. But there is a completely general result that if Y_i are a independent random variables and a_i are some constants then:

$$\text{var}\left(\sum a_i Y_i\right) = \sum a_i^2 \text{var}(Y_i)$$

Applying this to $\hat{\beta}$ we have:

$$\text{var}(\hat{\beta}) = \sum x_i^2 \left(\sum x_i^2\right)^{-2} \sigma^2 = \left(\sum x_i^2\right)^{-1} \sigma^2$$

If we make the additional assumption that $\epsilon_i \sim N(0, \sigma^2)$ then it's also easy to show that $\hat{\beta} \sim N(\beta, \text{var}(\hat{\beta}))$. Notice that nothing done so far has estimated σ^2. It isn't possible to estimate σ^2 by least squares, so we estimate it instead from the variance of the model residuals ($e_i = y_i - \hat{\beta}x_i$):

$$\hat{\sigma}^2 = \frac{1}{n-1} \sum (y_i - \hat{\beta}x_i)^2$$

notice how we've actually chosen $\hat{\beta}$ to minimise this quantity. The $n - 1$ term ensures that $\hat{\sigma}^2$ is an unbiased estimator - which means that the average value of $\hat{\sigma}^2$, calculated over an infinite set of replicate datasets, would be σ^2. $\hat{\beta}$ is also unbiased.

1.2 Linear models in general

The theory of linear models can be generalized in a straightforward manner to any number of predictor variables per response variable. The results of this generalization look remarkably similar to the results from the previous section, except that β becomes a vector and we use matrices full of explanatory variables, rather than x_i.

The basic trick for generalizing is to write the linear model in the general form:

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

where $\mathbf{Y}$ and $\boldsymbol{\epsilon}$ are the vectors of response variables and "error" terms respectively, $\boldsymbol{\beta}$ is a parameter vector and $\mathbf{X}$ is a matrix containing the explanatory variable arranged in some way. The assumptions about the elements ϵ_i of $\boldsymbol{\epsilon}$ are unchanged.

For example the "straight line" model:

$$Y_i = \beta_1 + \beta_2 x_i + \epsilon_i$$

becomes:

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \cdot \\ \cdot \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \cdot & \cdot \\ \cdot & \cdot \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \cdot \\ \cdot \end{bmatrix}$$

while the "plane":

$$Y_i = \beta_1 + \beta_2 x_i + \beta_3 z_i + \epsilon_i$$

can be written:

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \cdot \\ \cdot \end{bmatrix} = \begin{bmatrix} 1 & x_1 & z_1 \\ 1 & x_2 & z_2 \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \cdot \\ \cdot \end{bmatrix}$$

Another example would be a polynomial model, e.g.

$$Y_i = \beta_1 + \beta_2 x_i + \beta_3 x_i^2 + \epsilon_i$$

which can be written:

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \cdot \\ \cdot \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \cdot \\ \cdot \end{bmatrix}$$

This last model re-emphasises a key point: these models are linear because the parameters and error term enter the model in a linear way - the model is quite free to depend on the explanatory variables in a non-linear fashion.

These general linear models can be fitted using the same principles as we used for the simple one parameter example. Namely we seek the parameters that minimise the sum of squared residuals (the sum of squared differences between model predictions and data). To facilitate this, note that:

$$\sum_i e_i^2 = \mathbf{e}^T\mathbf{e} \;\text{ where }\; \mathbf{e} = \mathbf{y} - \mathbf{X}\boldsymbol{\beta}$$

Writing $S(\boldsymbol{\beta})$ for the the sum of squares of residuals we have:

$$\begin{aligned} S(\boldsymbol{\beta}) &= \sum e_i^2 \\ &= (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \\ &= \mathbf{y}^T\mathbf{y} - 2\boldsymbol{\beta}^T\mathbf{X}^T\mathbf{y} + \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} \end{aligned}$$

and this is minimised by $\hat{\boldsymbol{\beta}}$ satisfying:

$$\mathbf{X}^T\mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}^T\mathbf{y} \tag{1}$$

The latter is easily proven, by proving that $S(\boldsymbol{\beta}) \geq S(\hat{\boldsymbol{\beta}})$ for any $\boldsymbol{\beta}$. First note that:

$$S(\boldsymbol{\beta}) - S(\hat{\boldsymbol{\beta}}) = \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} - 2\boldsymbol{\beta}^T\mathbf{X}^T\mathbf{y} + 2\hat{\boldsymbol{\beta}}^T\mathbf{X}^T\mathbf{y} - \hat{\boldsymbol{\beta}}^T\mathbf{X}^T\mathbf{X}\hat{\boldsymbol{\beta}}$$

Now we can substitute for the terms $\boldsymbol{X}^T\mathbf{y}$ from (1):

$$S(\boldsymbol{\beta}) - S(\hat{\boldsymbol{\beta}}) = \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} - 2\boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\hat{\boldsymbol{\beta}} + \hat{\boldsymbol{\beta}}^T\mathbf{X}^T\mathbf{X}\hat{\boldsymbol{\beta}}$$

and it's not hard to see that:

$$S(\boldsymbol{\beta}) - S(\hat{\boldsymbol{\beta}}) = [\mathbf{X}(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}})]^T[\mathbf{X}(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}})]$$

but this latter term is clearly a sum of squares, which must be ≥ 0, so the result is proven. i.e. the least squares estimates of the parameters are given by:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Now from the model assumptions, we have that $E(\boldsymbol{\epsilon}) = \mathbf{0}$ and the covariance matrix of $\boldsymbol{\epsilon}$ is $\mathbf{V}_\epsilon = \mathbf{I}\sigma^2$. Since the only random element of $\mathbf{Y}$ is $\boldsymbol{\epsilon}$, $\mathbf{V}_\epsilon$ is also the covariance matrix of $\mathbf{Y}$. First let's show that $\hat{\boldsymbol{\beta}}$ is unbiased.

$$E(\boldsymbol{\epsilon}) = \mathbf{0} \Rightarrow E(\mathbf{Y}) = \mathbf{X}\boldsymbol{\beta}$$

so

$$E(\hat{\boldsymbol{\beta}}) = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T E(\mathbf{Y}) = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} = \boldsymbol{\beta}$$

i.e. the expected value of the estimator $\hat{\boldsymbol{\beta}}$ is $\boldsymbol{\beta}$, or in other words: on average the estimator gets it right.

It's nice to know the variance of the estimator as well, and it's quite straightforward to find the covariance matrix, $\mathbf{V}_{\hat{\beta}}$ of $\hat{\boldsymbol{\beta}}$. In general if $\mathbf{a}$ is a vector of random variables with covariance matrix $\mathbf{V}_a$ and $\mathbf{b} = \mathbf{B}\mathbf{a}$, where $\mathbf{B}$ is any matrix of coefficients, then the covariance matrix of the vector of random variables $\mathbf{b}$ is $\mathbf{V}_b = \mathbf{B}\mathbf{V}_a\mathbf{B}^T$. Using this result directly we have that:

$$\mathbf{V}_{\hat{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{I}\sigma^2\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1} = (\mathbf{X}^T\mathbf{X})^{-1}\sigma^2$$

(it's the fact that σ^2 is scalar that lets us move it in the above expression; note also that $\mathbf{X}^T\mathbf{X}$ is symmetric). σ^2 is estimated from the residuals as in the one parameter case:

$$\hat{\sigma}^2 = \frac{1}{n-p}\sum_{i=1}^{n} e_i^2$$

where $\mathbf{e} = \mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}$ is the vector of observed residuals, and p is the length of $\boldsymbol{\beta}$ (i.e. number of model parameters). It's actually possible to show that out of all unbiased estimators of $\boldsymbol{\beta}$ that are linear in the data, the least squares estimator $\hat{\boldsymbol{\beta}}$ has the lowest variance.

If we were to add the assumption that the ϵ_i are normally distributed then $\hat{\boldsymbol{\beta}} \sim N(\boldsymbol{\beta}, \mathbf{V}_{\hat{\beta}})$, and we could do things like find confidence intervals and test hypotheses.

1.3 Geometry of linear model fitting

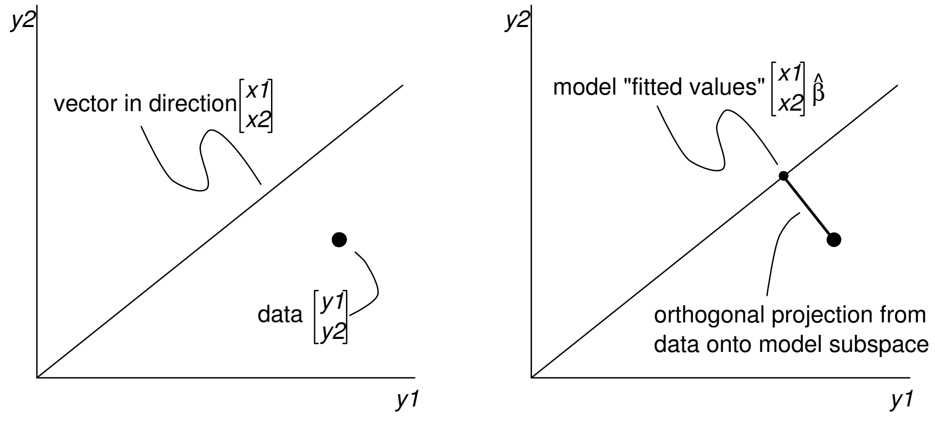
It's useful to have a geometric picture of how linear model fitting works, especially when it comes to dealing with non-linear models later. The basic idea is that the length n data vector $\mathbf{y}$ can be though of as a single point in an n dimensional Euclidean space. Within this space the columns of $\mathbf{X}$ define a "model subspace" in which the model fitted values must lie — varying the parameter vector $\boldsymbol{\beta}$ moves the fitted values around within this space. Fitting by least squares amounts to finding the point in the model subspace that is closest to the data. These ideas are most easily illustrated by considering a very simple example model, for which it's easy to draw everything. Consider the model:

$$y_i = \beta x_i + e_i$$

fitted to 2 data. This can be written:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \beta + \begin{bmatrix} e_1 \\ e_2 \end{bmatrix}$$

The following two panels illustrates how this model is fitted.


y2
vector in direction $\begin{bmatrix} x1 \\ x2 \end{bmatrix}$
data $\begin{bmatrix} y1 \\ y2 \end{bmatrix}$
y1
y2
model "fitted values" $\begin{bmatrix} x1 \\ x2 \end{bmatrix}\hat{\beta}$
orthogonal projection from data onto model subspace
y1

Both panels show a two dimensional space in which the data vector $\mathbf{y}$ is a single point. The line shows the subspace spanned by the columns of $\mathbf{X}$ — just a vector in the direction $(x_1, x_2)^T$. Least squares fitting amounts to taking the orthogonal projection of $\mathbf{y}$ onto this subspace — the projection is at $\hat{\beta}(x_1, x_2)^T$, from which $\hat{\beta}$ can be deduced. More data simply increases the dimension of the data space, while more parameters/columns of $\mathbf{X}$ just increase the dimension of the model subspace.

1.4 Simple biological model fitting by linear least squares

To see how the general statistical theory, given above, can be used for estimation with biological models consider a simple example employing a Caswell type matrix model for a size structured population. To keep things as simple as possible, suppose that the population consists of just two classes, juvenile and adult, say. Let $n_1(t)$ and $n_2(t)$ denote the juvenile and adult populations respectively, and consider the model:

$$\begin{bmatrix} n_1 \\ n_2 \end{bmatrix}_{t+1} = \begin{bmatrix} P_1 & B \\ G_1 & P_2 \end{bmatrix} \begin{bmatrix} n_1 \\ n_2 \end{bmatrix}_t + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \end{bmatrix}_t$$

where the ϵ are all independent with zero mean and the same variance. The model is strictly appropriate for a situation in which the population is subject to process error, but can be observed without error. Suppose that the population is observed at times $0, 1, 2, 3, \ldots$. To use the theory from the previous section, we need to be able to write the model in the form of a linear model, which means that the parameters need to be in one parameter vector, which when pre-multiplied by an $\mathbf{X}$ matrix yields the model predictions of the observed data. This is actually not too hard to do:

$$\begin{bmatrix} n_1(1) \\ n_2(1) \\ n_1(2) \\ n_2(2) \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} = \begin{bmatrix} n_1(0) & n_2(0) & 0 & 0 \\ 0 & 0 & n_1(0) & n_2(0) \\ n_1(1) & n_2(1) & 0 & 0 \\ 0 & 0 & n_1(1) & n_2(1) \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix} \begin{bmatrix} P_1 \\ B \\ G_1 \\ P_2 \end{bmatrix} + \begin{bmatrix} \epsilon_1(0) \\ \epsilon_2(0) \\ \epsilon_1(1) \\ \epsilon_2(1) \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}$$

i.e. the general form is:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

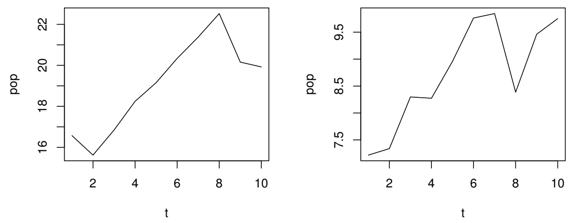
and we can estimate everything by least squares.

1.4.1 Numerical Example

It's always worth checking estimation methods by simulation. The reason for this is simple: models and/or fitting procedures *always* involve assumptions and approximations, and it is important to investigate how these effect your scientific conclusions. In this section then, let's try out the simple matrix model fitting method using some simulated data. I simulated data for 10 timesteps from the model:

$$\begin{bmatrix} n_1 \\ n_2 \end{bmatrix}_{t+1} = \begin{bmatrix} 0.3 & 1.6 \\ 0.2 & 0.6 \end{bmatrix} \begin{bmatrix} n_1 \\ n_2 \end{bmatrix}_t + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \end{bmatrix}_t$$

with $\epsilon_i \sim N(0,1)$ and independent for all i and t and the $\mathbf{n}_0 = (5,10)^T$. Example data are shown here, with stage 1 on the left and stage 2 on the right (note that I haven't rounded to whole individuals!):



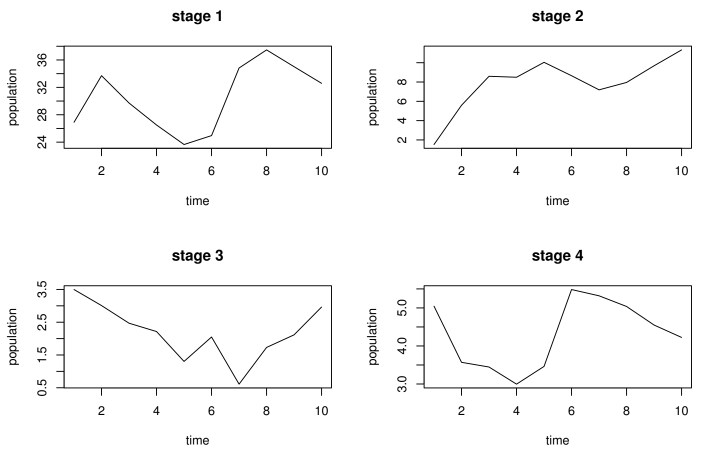
... fitting to these data, exactly as described above yields the following estimates for the parameters (left) and parameter estimator standard errors (right)

$$\begin{bmatrix} 0.34 & 1.49 \\ 0.18 & 0.63 \end{bmatrix} \qquad \begin{bmatrix} 0.04 & 0.08 \\ 0.04 & 0.08 \end{bmatrix}$$

... these results look quite encouraging, but the test is really a bit too easy — in most cases the data will be subject to observational errors as well as process errors so that our model will not actually be quite right, and often we'll be interested in more than 2 stages, so let's look at a more realistic case, the following model:

$$\begin{bmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \end{bmatrix}_{t+1} = \begin{bmatrix} 0.3 & 0.0 & 0.0 & 5.0 \\ 0.2 & 0.3 & 0.0 & 0.0 \\ 0.0 & 0.2 & 0.3 & 0.0 \\ 0.0 & 0.0 & 0.3 & 0.9 \end{bmatrix} \begin{bmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \end{bmatrix}_t + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \end{bmatrix}_t$$

with $\mathbf{n}_0 = (5,5,5,5)^T$ and $\epsilon_i \sim N(0,1)$ independent for all i and t. Here are some data simulated from this model:

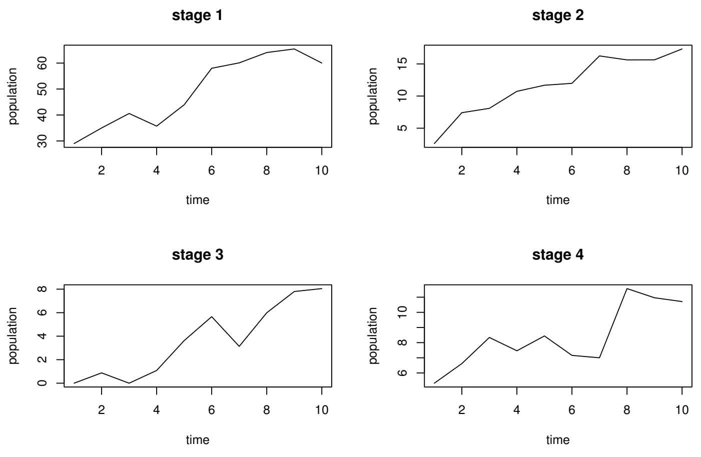


Estimating the model coefficients from these data yields:

$$\begin{bmatrix} 0.31 & 0.0 & 0.0 & 5.00 \\ 0.18 & 0.39 & 0.0 & 0.0 \\ 0.0 & 0.084 & 0.64 & 0.0 \\ 0.0 & 0.0 & -0.07 & 1.00 \end{bmatrix} \qquad \begin{bmatrix} 0.03 & 0.0 & 0.0 & 0.18 \\ 0.03 & 0.10 & 0.0 & 0.0 \\ 0.0 & 0.059 & 0.17 & 0.0 \\ 0.0 & 0.0 & 0.24 & 0.14 \end{bmatrix}$$

— again estimates are on the left with standard errors on the right. Notice two problems. Firstly, one of the estimates is biological nonsense - it's negative. Secondly the estimates deteriorate rapidly as we advance through the stages: this phenomenon is rather general when modelling stage structured populations. To see the problem just consider estimating mortality for the moment. If you underestimate it for one stage then you'll imply higher recruitment to the next stage than actually occurred — to match the data for the next stage, you'll have to compensate for this by overestimating the mortality rate in this next stage in order to "kill off" all the extra recruits. The problem is that the average number of individuals per unit age interval can only decrease as you advance through the stages, so to kill off extra recruits accumulated by underestimating per capita mortality rate in one stage requires a slightly bigger overestimation of mortality rate in the next stage (same applies if the sequence is over, then under). Hence estimation errors increase with stage!

The previous poor results occurred when the model was strictly correct in that we observed populations exactly. If, in addition to the process error in the above model, we simulate data in which the observation process is also uncertain then things get even worse. Here is some data simulated from the previous model but with additional $N(0, \sigma = 2)$ sampling error added to all population observations:



Here are the corresponding estimates:

$$\begin{bmatrix} 0.61 & 0.0 & 0.0 & 2.68 \\ 0.22 & 0.19 & 0.0 & 0.0 \\ 0.0 & 0.27 & 0.38 & 0.0 \\ 0.0 & 0.0 & -0.15 & 1.10 \end{bmatrix}$$

This time I have not given the standard errors calculated from the linear regression, since with observational error the model is no longer actually correct. Notice that the estimates have got even worse: G_3 is even more negative and P_4 is greater than 1: clearly not possible for a survival term! Again this sensitivity to observational noise is typical of this sort of approach.

2 Constrained and penalized linear modelling

In the simple example above we encountered some practical difficulties in the face of even quite low levels of noise in the data and the modelled processes. There are 3 ways to make progress:

1. We could make a somewhat radical change of approach, and use Bayesian statistical methods. The key feature of these methods is that we don't treat model parameters as fixed quantities to be estimated (with all uncertainty associated with our *estimates* of those parameters), but rather treat the parameters themselves as random variables. To make progress with such an approach we have to specify distributions for the parameters. The Bayesian approach then provides a recipe for updating our beliefs about the parameter distributions, given the data that we have observed. Hence we shift the objective from estimating unknown parameters to modifying prior believes about the parameters. If we have good prior knowledge then this approach will work well (although it's quite technically complicated), but if we don't then it will be as bad or worse than the method we already have.

2. If we are only interested in using the model for **prediction** (e.g. what happens at the next timestep?) then it may be OK to ignore the problems. The problems that we have in estimating parameters result from the fact that there are many parameter sets that give very similar dynamics. For prediction all we care about is that we can obtain one such set — and our estimation procedure doesn't have a problem doing that.

3. We could try introducing biologically plausible restrictions into the model. The simplest such restrictions involve using fewer parameters. For example we might believe that $P_1 = P_2$ in which case we can re-formulate slightly so that we are only estimating a single P in place of P_1 and P_2: this requires no new technology. Other restrictions do require extensions to the methods met so far. For example in the simple example we know that all parameters must be ≥ 0, and that $G_1 + P_1 < 1$. Imposing these restrictions as part of modelling limits the scope for the estimates to be very silly, but requires a constrained fitting method. We might also want to impose restrictions along the lines of $P_1 \approx P_2$ — this can be done, but again requires some additions to our existing theory.

In these notes approach 3 will be pursued.

2.1 Inequality constraints

The first sort of extra biological information that could be included is to insist that the model parameters have to be consistent with basic biological facts. For example, in the simple 2 stage matrix model given above we know that all the parameters must be non-negative. Furthermore, to avoid implied negative death rates the parameters must meet the conditions $P_1 + G_1 \leq 1$ and $P_2 \leq 1$. In fact if we collect together all such constraints we get the matrix inequality:

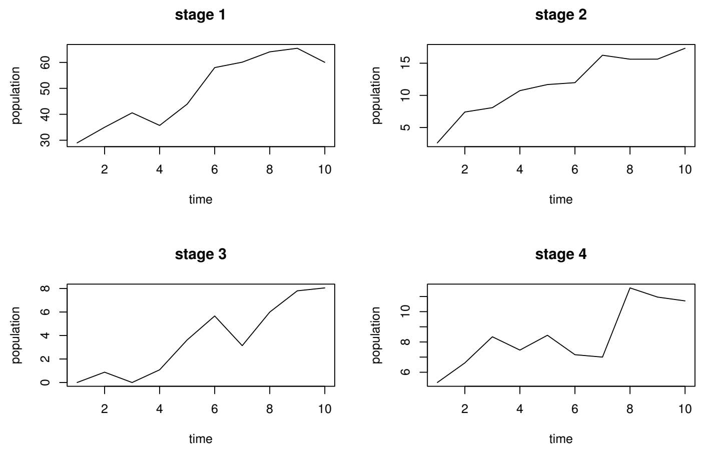
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} P_1 \\ B \\ G_1 \\ P_2 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -1 \\ -1 \end{bmatrix}$$

Or $\mathbf{C}\boldsymbol{\beta} \geq \mathbf{b}$, for short. To impose these constraints during fitting means that we must solve the **quadratic programming** problem:

$$\text{minimize } \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 \text{ subject to } \mathbf{C}\boldsymbol{\beta} \geq \mathbf{b}$$

(where $\|\mathbf{x}\|^2 \equiv \mathbf{x}^T\mathbf{x}$ for all vectors $\mathbf{x}$ — i.e. $\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2$ is just the familiar sum of squares objective). Fortunately this can always be done (for any $\mathbf{C}$, $\mathbf{b}$ pair, not just the given example).

As an example I simulated another set of data from the 4 stage model with sampling error, and then estimated its parameters with and without the constraints. Here are the data:



The following summarise the results, with truth on the left, unconstrained estimates in the middle and constrained estimates on the right:

$$\begin{bmatrix} 0.3 & 0.0 & 0.0 & 5.0 \\ 0.2 & 0.3 & 0.0 & 0.0 \\ 0.0 & 0.2 & 0.3 & 0.0 \\ 0.0 & 0.0 & 0.3 & 0.9 \end{bmatrix} \quad \begin{bmatrix} 0.78 & 0.0 & 0.0 & 1.9 \\ 0.29 & -0.12 & 0.0 & 0.0 \\ 0.0 & 0.10 & 0.16 & 0.0 \\ 0.0 & 0.0 & -0.14 & 0.92 \end{bmatrix} \quad \begin{bmatrix} 0.76 & 0.0 & 0.0 & 2.0 \\ 0.24 & 0.064 & 0.0 & 0.0 \\ 0.0 & 0.10 & 0.16 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.86 \end{bmatrix}$$

It's clear that the unconstrained estimates have several problems: P_1 and G_1 imply survival rates of more than 1 for the first stage, $P_2 < 0$ and $G_3 < 0$. The constrained estimates avoid these problems, but the estimated model suggests that no-one ever reaches adulthood! Clearly we have some way to go yet.

2.2 Quadratic penalties

The linear inequality constraints covered in the previous section can only improve matters by constraining the parameters to be just barely acceptable. i.e. if the constraints do anything it is to set some parameters to meet some of the constraints exactly. Clearly it would be nice to do a bit better than just barely avoiding nonsense in the estimates. This section covers a modification of our general model fitting approach in which parameter estimates that deviate from some expected behaviour are penalized in some way.

Let's start with an example. Suppose that we expected P_1 and P_2 in the simple matrix model to be "close" to each other. One way to achieve this would be to add a penalty term to the fitting objective:

$$\text{minimise } \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda(P_1 - P_2)^2$$

where λ is a parameter controlling the relative weight to give to the conflicting goals of keeping the P_i's close together, relative to fitting the data as closely as possible. Choosing λ will be covered later.

It's not too hard to see how this minimisation can be performed in practice. A row $[\sqrt{\lambda}, 0, 0, -\sqrt{\lambda}]$ can be added to the end of $\mathbf{X}$, to give $\tilde{\mathbf{X}}$, say, and we can also add a corresponding 0 to the end of $\mathbf{y}$, to give $\tilde{\mathbf{y}}$. We then seek to mimimize:

$$\|\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\boldsymbol{\beta}\|^2$$

which is easily seen to be exactly equivalent to the original penalized objective. Basically we've got the model to predict a new quantity, and specified a value that it should be close to, and it's straightforward to incorporate this into least squares fitting.

In the interests of greater generality it's worth writing this fitting problem another way as well. It's quite easy to show that:

$$\|\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\boldsymbol{\beta}\|^2 \equiv \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda\boldsymbol{\beta}^T\mathbf{S}\boldsymbol{\beta}$$

where:

$$\lambda\mathbf{S} = \begin{bmatrix} \sqrt{\lambda} \\ 0 \\ 0 \\ -\sqrt{\lambda} \end{bmatrix} \begin{bmatrix} \sqrt{\lambda} & 0 & 0 & -\sqrt{\lambda} \end{bmatrix} = \lambda \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix}$$

It turns out that for any (symmetric positive semi-definite) penalty coefficient matrix $\mathbf{S}$, we can always find a $\tilde{\mathbf{X}}$[2], and conversely penalties written directly in the $\tilde{\mathbf{X}}$ form can always be re-expressed using a penalty coefficient matrix $\mathbf{S}$. The latter turns out to be more useful when we need to estimate λ, whereas the former makes it very easy to estimate penalized models using standard linear modelling routines.

Quite a number of constraints result in quadratic penalties of this sort. For example consider a 3 stage model, on which we would like to impose the constraints that the rate of change of P_i should not change too rapidly. This means that we would like:

$$(P_1 - 2P_2 + P_3)^2$$

to be close to zero. It should be clear that producing an $\mathbf{S}$ matrix or an $\tilde{\mathbf{X}}$ matrix given this constraint follows in exactly the same way as the simpler case given above. It should also be clear that several such penalties can be introduced without extra difficulties.

As a practical example the two stage model with the given penalty is really rather too un-biological in most cases - there is very seldom any sound reason to expect the juvenile and adult P's to be similar. So let's consider instead a 3 stage model, for which rates in the first two (juvenile) stages are considered likely to be similar.

The basic model is then:

$$\begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix}_{t+1} = \begin{bmatrix} P_1 & 0 & B \\ G_1 & P_2 & 0 \\ 0 & G_2 & P_3 \end{bmatrix} \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix}_t + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{bmatrix}_t$$

with the expectation that P_1 and P_2 will be similar, and that G_1 and G_2 will also be similar. Getting this into the general linear model form, $\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, is straightforward:

$$\begin{bmatrix} n_1(1) \\ n_2(1) \\ n_3(1) \\ n_1(2) \\ . \\ . \\ . \\ . \end{bmatrix} = \begin{bmatrix} n_1(0) & n_3(0) & 0 & 0 & 0 & 0 \\ 0 & 0 & n_1(0) & n_2(0) & 0 & 0 \\ 0 & 0 & 0 & 0 & n_2(0) & n_3(0) \\ n_1(1) & n_3(1) & 0 & 0 & 0 & 0 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \end{bmatrix} \begin{bmatrix} P_1 \\ B \\ G_1 \\ P_2 \\ G_2 \\ P_3 \end{bmatrix} + \begin{bmatrix} \epsilon_1(0) \\ \epsilon_2(0) \\ \epsilon_3(0) \\ \epsilon_1(1) \\ . \\ . \\ . \\ . \end{bmatrix}$$

and the quadratic penalties $\lambda_1(G_1 - G_2)^2$ and $\lambda_2(P_1 - P_2)^2$ are easily incorporated either by adding two zeroes to the data vector and the rows $[0, 0, \sqrt{\lambda_1}, 0, -\sqrt{\lambda_1}, 0]$ and $[\sqrt{\lambda_2}, 0, 0, -\sqrt{\lambda_2}, 0, 0]$ to $\mathbf{X}$, or by working out the corresponding penalty matrices:

$$\lambda_1\mathbf{S}_1 = \lambda_1 \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad \lambda_2\mathbf{S}_2 = \lambda_2 \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

[2]Actually doing this for arbitrary positive semi-definite $\mathbf{S}$ involves finding a square root of $\mathbf{S}$: easily done in standard software using an eigen-decomposition or a singular value decomposition.

We may as well also insist that the parameter estimates obey the constraints ensuring that they are biologically meaningful, namely:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} P_1 \\ B \\ G_1 \\ P_2 \\ G_2 \\ P_3 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \\ -1 \\ -1 \end{bmatrix}$$

($\mathbf{C}\boldsymbol{\beta} \geq \mathbf{b}$, say.) Estimation then proceeds by solving:

$$\text{minimize } \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda_1 \boldsymbol{\beta}^T \mathbf{S}_1 \boldsymbol{\beta} + \lambda_2 \boldsymbol{\beta}^T \mathbf{S}_2 \boldsymbol{\beta} \text{ subject to } \mathbf{C}\boldsymbol{\beta} \geq \mathbf{b}$$

Trying this approach on the simulated 4 stage data used at the end of the last section and applying a penalty on $(P_1 - P_2)^2 + (P_2 - P_3)^2 + (G_1 - G_2)^2 + (G_2 - G_3)^2$, yields (truth on left, estimates on right):

$$\begin{bmatrix} 0.3 & 0.0 & 0.0 & 5.0 \\ 0.2 & 0.3 & 0.0 & 0.0 \\ 0.0 & 0.2 & 0.3 & 0.0 \\ 0.0 & 0.0 & 0.3 & 0.9 \end{bmatrix} \qquad \begin{bmatrix} 0.76 & 0.0 & 0.0 & 2.0 \\ 0.16 & 0.44 & 0.0 & 0.0 \\ 0.0 & 0.07 & 0.36 & 0.0 \\ 0.0 & 0.0 & 0.032 & 0.85 \end{bmatrix}$$

Now, these estimates are clearly rather better than previous methods have managed: none of the estiamtes are completely silly. But we are still not doing very well. Why is this apparently simple problem so difficult?

One reason is quite basic to this regression approach to estimation. The parameters are being chosen to do the best job of predicting the population one timestep ahead given the population now, but the population change over one timestep is the one most sensitive to sampling error. Another problem is that the model is really only strictly correct if we can observe the populations without error — as soon as there is noise in the observation process then the model is no longer quite right. Another issue is that the problem that we've looked at here is actually rather difficult. It's not too hard to show that you can't estimate growth and death rates from structured population data without employing very strong modelling assumptions (simply assuming growth and death rates vary smoothly is not enough, for example). This really means that the problem is not soluble without including a large amount of detailed biology into the population model — but doing so will require that we abandon simple models for which all estimation can be done using linear regression.

That said, the methods developed here are not completely useless. If we can directly observe some of the parameters then quite reasonable estimates may be obtainable for the others. For example, direct information may be available for the G_i's in some cases.

3 Modelling with smooth functions

In the previous sections we saw how simple discrete time models could be re-written in a form suitable for direct estimation using linear regression methods (including penalized constrained regression). In this section, we'll increase the biological generality a little, by considering continuous time models that can be estimated using penalized constrained regression methods while avoiding the problems associated with the one step ahead prediction approach of the previous section. The biological model that I want to deal with involves estimating rather general unknown *functions*. It's most straightforward to introduce the principles of modelling with unknown functions using a very simple model with little biological content, and then to move on to more interesting cases.

To cover most of the key ideas, consider a lab experiment in which a cohort of individuals is followed over time, the cohort constituting a closed population suffering death. At each of several times t_i we have an estimate of the cohort size y_i, and wish to model these data by:

$$y_i = f(t_i) + \epsilon_i$$

where $f()$ is a smooth function, and ϵ_i a zero mean random error term (independent for different i). Given the biological context here, we should really insist that $\mathrm{d}f/\mathrm{d}t < 0$, but ignore this restriction for the moment. We need a way of estimating f. There's some very elegant theory relating to the estimation of f and generalizations of this problem, which we'll only touch on here (see Green and Silverman 1994, and Wahba 1990).

The key to estimating f is to turn the estimation problem into one involving only unknown parameters rather than an unknown function. To do this we write f as a weighted sum of known "basis" functions, $b_i(t)$ as follows:

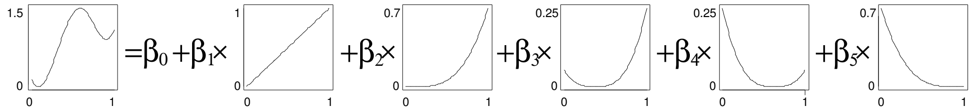
$$f(t) = \sum_{i=1}^{k} \beta_i b_i(t)$$

only the β_i are unknown in this expression, the $b_i(t)$ have no unknown parameters. The $b_i(t)$ define a basis for a space of functions (hence their name) and by varying the β_i we obtain different functions within the space. A simple example of a set of basis functions might be $b_1(t) = 1$, $b_2(t) = t$, $b_3(t) = t^2$, etc. although this "polynomial basis" is actually rather poor in practice.

A rather good set of basis functions is the "spline" basis:

$$b_1(t) = 1 \quad b_2(t) = t \quad b_3(t) = |t - t_1^*|^3 \quad b_4(t) = |t - t_2^*|^3 \quad b_5(t) = |t - t_3^*|^3 \quad \text{etc.}$$

where the t_i^* are a set of points known as "knots" spread "nicely" through the t_i data. The following is a picture[3] of how f is represented using this basis:



Given a basis we're in a position to estimate $f()$ using the linear model theory already met. Specifically, the model becomes:

$$\begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} = \begin{bmatrix} 1 & t_1 & |t_1 - t_1^*|^3 & |t_1 - t_2^*|^3 & \cdot & \cdot & \cdot \\ 1 & t_2 & |t_2 - t_1^*|^3 & |t_2 - t_2^*|^3 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ \cdot \\ \cdot \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}$$

and is easily recognisable as having the general form $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$ that we know how to use.

We now have an additional issue to deal with, however. How large should the number, k, of basis functions be? There are two points to consider here:

1. If k is too high then the model will be very flexible and should be able to reproduce the underlying true f quite well, but unfortunately, when used with noisy data it will tend to fit the noise!

2. If k is too low then we should avoid the problem of fitting the noise (over fitting), but at the cost of not necessarily having the model flexibility required to represent the true f very well.

So we really need to be quite careful in choosing some intermediate k if we are going to avoid problems. Unfortunately, however, it's often the case that a reasonable choise of k for avoiding overfitting is rather smaller than a reasonable choise for representing the true f well!

[3]This picture was lifted from a course in which labels started at 0! Sorry.

The solution is to use a fairly high k but to avoid overfitting by applying a fitting penalty on model "wiggliness": this penalty will turn out to be the sort of quadratic penalty that we've already met. In detail we'll fit by minimising:

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda J(f)$$

where

$$J(f) \equiv \int [f''(x)]^2 dx$$

i.e. the integral of the squared second derivative of f — the wiggliness of f.

Now it's not immediately obvious that this expression is anything like the quadratic penalties that we met before, so it's worth seeing why it is in fact a quadratic penalty (although I will not slog through the algebra required to actually calculate the coefficients of the penalty). First note that the second derivative of f is just a weighted sum of the second derivatives of its basis:

$$f''(t) = \sum \beta_i b_i''(t) \equiv \begin{bmatrix} b_1''(t) & b_2''(t) & . & . \end{bmatrix} \boldsymbol{\beta}$$

Now $f(t)$ is a scalar, and therefore equal to its own transpose, so:

$$\begin{aligned} [f''(t)]^2 = [f''(t)]^T[f''(t)] &= \boldsymbol{\beta}^T \begin{bmatrix} b_1''(t) \\ b_2''(t) \\ . \\ . \end{bmatrix} \begin{bmatrix} b_1''(t) & b_2''(t) & . & . \end{bmatrix} \boldsymbol{\beta} \\ &= \boldsymbol{\beta}^T \begin{bmatrix} [b_1''(t)]^2 & b_1''(t)b_2''(t) & . & . \\ b_1''(t)b_2''(t) & [b_2''(t)]^2 & . & . \\ . & . & . & . \\ . & . & . & . \end{bmatrix} \boldsymbol{\beta} \end{aligned}$$

Hence:

$$J(f) = \boldsymbol{\beta}^T \begin{bmatrix} \int [b_1''(x)]^2 dx & \int b_1''(x)b_2''(x)dx & . & . \\ \int b_1''(x)b_2''(x)dx & \int [b_2''(x)]^2 dx & . & . \\ . & . & . & . \\ . & . & . & . \end{bmatrix} \boldsymbol{\beta} \equiv \boldsymbol{\beta}^T \mathbf{H} \boldsymbol{\beta}$$

by definition of $\mathbf{H}$. So provided we can do the required differentiation and integration of the basis functions, it's possible to re-express $J(f)$ as $\boldsymbol{\beta}^T\mathbf{H}\boldsymbol{\beta}$. For the spline basis given above $\mathbf{H}$ can simply be looked up in a textbook (e.g. Green and Silverman, 1994).

So the model fitting exercise has the general form:

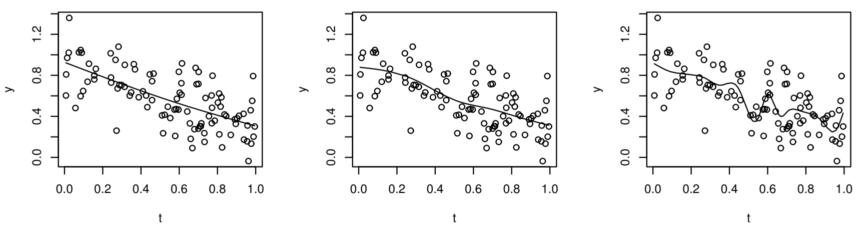
$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \boldsymbol{\beta}^T \mathbf{H} \boldsymbol{\beta}$$

...and we know how to solve this, given λ. It probably won't surprise you to learn that we can also find linear inequality constraints that impose on f the requirement that it should be non-increasing with time. It's possible to find sufficient constraints to make sure that f is strictly non-increasing, but their derivation is somewhat involved. It's much easier to follow the derivation of approximate constraints. Just choose a set of t values closely but evenly spaced over the interval $[t_1, t_n]$: $t_1^\circ, t_2^\circ, \ldots$. Now we can approximately ensure non-increase and positiveness of f with the constraints:

$$\begin{bmatrix} -b_1'(t_1^\circ) & -b_2'(t_1^\circ) & -b_3'(t_1^\circ) & . & . \\ -b_1'(t_2^\circ) & -b_2'(t_2^\circ) & -b_3'(t_2^\circ) & . & . \\ -b_1'(t_3^\circ) & -b_2'(t_3^\circ) & -b_3'(t_3^\circ) & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ b_1(t_n) & b_2(t_n) & b_3(t_n) & . & . \end{bmatrix} \boldsymbol{\beta} \geq \mathbf{0}$$

and again we can fit the model subject to these constraints by quadratic programming.

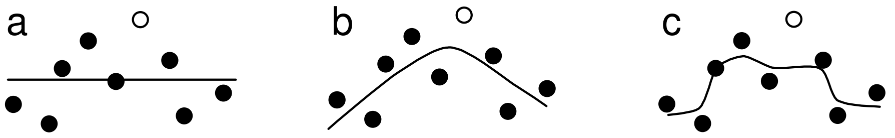
Here's an example with no constraints but different λ values (high to low from left to right):



which begs the question: how should λ be chosen?

3.1 Choosing λ

The way to choose λ objectively is based on the following idea. Suppose you left a datapoint, $\circ$,out of the model fit and just fitted the model to the remaining data, $\bullet$. Now consider how well the model would fit the missing data point, $\circ$, under different scenarios:



In scenario **a**, we fit a very smooth model with very high λ — it doesn't fit any of the data very closely and doesn't do any better on the missing point. In scenario **c**, a very wiggly model with low λ is fitted — this seems to be fitting the "noise" in the data as well as the systematic trend: as a result it wiggles about alot and is a long way from the point $\circ$. The intermediate scenario **b** is better: here a moderately smooth curve has been fitted that seems to capture the trend quite well, while smoothing out the noise — with this optimum amount of wiggliness the model does the best job of getting close to the point $\circ$.

Choosing the model that does best at matching points in the data set that it was not actually fitted to, is called "cross validation". In practice each datapoint is left out in turn and the average squared difference between missing data and model is calculated for each candidate model (i.e. each λ value). The model that minimises this quantity if doing best at predicting the data it was not fitted to — and is therefore selected. The different candidate models will differ in their λ value, and hence in how wiggly they can be.

In practice with penalized linear models we can actually obtain the cross validation score rather efficiently for each different λ, and it turns out to be a weighted sum of the model residuals. However, ordinary cross validation has some peculiar properties that suggest that a modified version of it might actually be better in practice: this modification consists of simply setting all the weights in cross validation score equal to the mean weight. The resulting Generalized Cross Validation score is:

$$V(\lambda) = \frac{\|\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}_\lambda\|^2}{[\text{tr}(\mathbf{I} - \mathbf{A}_\lambda)]^2}$$

where $\hat{\boldsymbol{\beta}}_\lambda$ is the estimated parameter vector for a given λ and $\mathbf{A}_\lambda \equiv \mathbf{X}(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{H})^{-1}\mathbf{X}^T$. The trace term in the denominator of the expression is the estimated effective residual degrees of freedom, given λ. λ is chosen to minimise $V(\lambda)$. It's interesting to note that the GCV score is actually the model estimate of residual variance, divided by the estimated residual degrees of freedom. So GCV amounts to minimising the residual variance per residual degree of freedom.

If a model has several penalties with several smoothing parameters then it is possible to estimate all of them simultaneously by GCV, but quite challenging numerically.

3.2 Continuous model for an age structured population

Consider a closed population in which the population per unit age interval at time t and age a is given by $\eta(a,t)$, a smooth function of age and time. $\eta(a,t)$ is related to the per capita death rate in the population, $\mu(a,t)$ via the p.d.e.

$$\frac{\partial \eta}{\partial a} + \frac{\partial \eta}{\partial t} + \mu\eta = 0$$

Typically we might be able to observe the number of individuals within population age classes, so that a typical datum for this population might be:

$$y_i = \int_{a_i^-}^{a_i^+} \eta(x,t_i)dx + \epsilon_i$$

where the age interval for this datum is $[a_i^-, a_i^+)$ and the observation is made at time t_i. ϵ_i is a zero mean error term, as usual. The idea is that we estimate $\eta(a,t)$ (and hence $\mu(a,t)$) from data y_i (given that a_i^+, a_i^- and t_i are known).

There are actually a number of ways of approaching this problem: let's use a fairly straightforward one here. We can represent $\eta(a,t)$ using a "thin-plate" spline basis, which is the natural generalization to functions of several variables of the spline basis introduced earlier. There's a whole family of such bases, depending partly on the exact nature of the wiggliness penalty to be used. I'll use:

$$J(\eta) = \int\int \left(\frac{\partial^3 \eta}{\partial t^3}\right)^2 + 3\left(\frac{\partial^3 \eta}{\partial a^2 \partial t}\right)^2 + 3\left(\frac{\partial^3 \eta}{\partial a \partial t^2}\right)^2 + \left(\frac{\partial^3 \eta}{\partial t^3}\right)^2 dadt$$

to measure wiggliness - it turns out that this is quadratic in the parameters defined below. This penalty gives rise to the following set of basis functions. $b_1(a,t)$ to $b_6(a,t)$ are $1, a, t, at, t^2, a^2$ (basically a basis for the space of functions of zero wiggliness according to $J(\eta)$). Defining a set of "knots" (a_i^*, t_i^*) and $r_i \equiv \sqrt{(a-a_i^*)^2 + (t-t_i^*)^2}$, the remaining basis functions are $b_{i+6}(a,t) = r_i^4 log(r_i)$ (defined to be zero at $r_i = 0$).

So

$$\eta(a,t) = \sum_{i=1}^{k} \beta_i b_i(a,t)$$

The choice of basis functions and penalty is partly guided by the fact that in this application we would often like to estimate the population death rate from the fitted $\eta(a,t)$: to ensure smoothness of this estimate requires that we use 3rd order derivatives in the penalty, for reasons that are purely technical.

As in the simple one dimensional case, the penalty can be written $J(\eta) = \boldsymbol{\beta}^T\mathbf{H}\boldsymbol{\beta}$ for some known matrix of coefficients $\mathbf{H}$ (see Green and Silverman 1994, if you really want to know the elements of $\mathbf{H}$). It's clear then that the model is simply a penalized linear model that can be fitted by minimization of some expression like:

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda\boldsymbol{\beta}^T\mathbf{H}\boldsymbol{\beta}$$

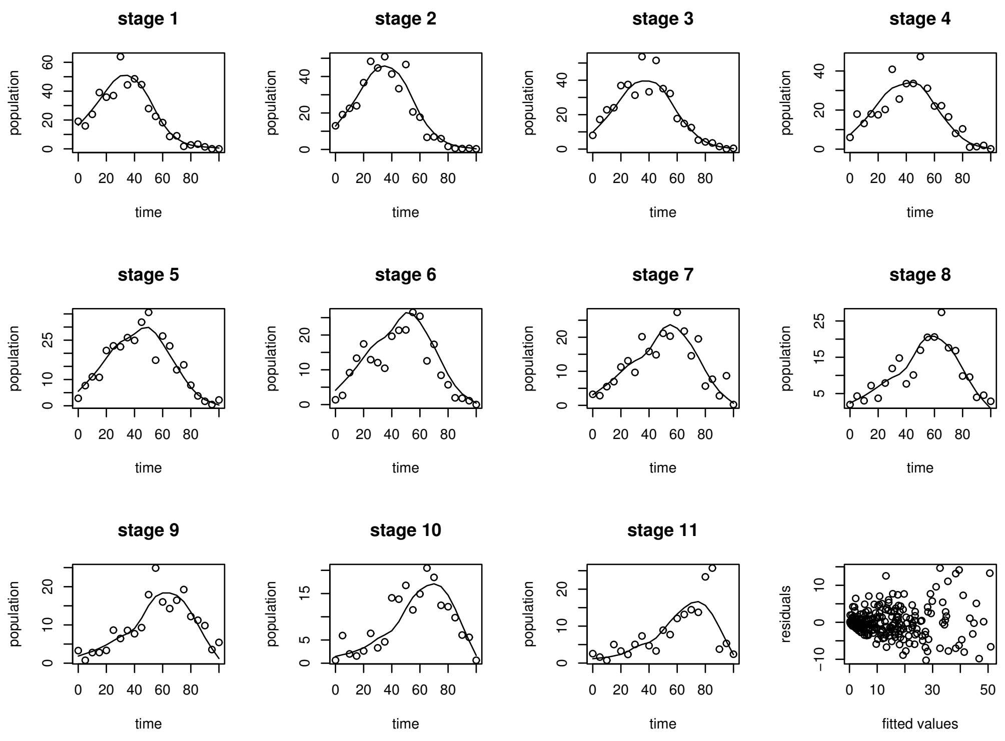
and of course we can choose λ by GCV.

As with previous models there are some basic biological constraints that should be applied to $\eta(a,t)$. Firstly, it's pretty clear that $\eta(a,t)$ can not be negative, and secondly it should not imply negative death rates! The latter condition is equivalent to requiring that:

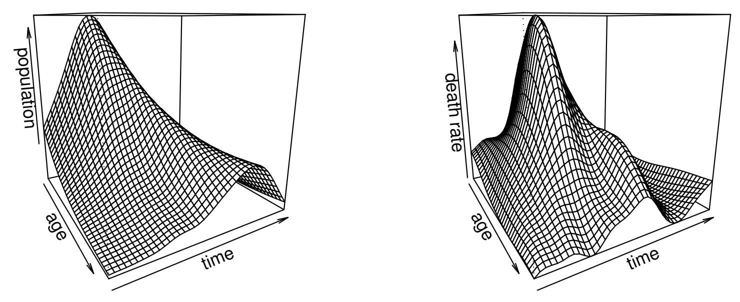
$$\frac{\partial \eta}{\partial a} + \frac{\partial \eta}{\partial t} \leq 0$$

As in the simple one dimensional example it's easy to ensure that these conditions hold approximately, by ensuring that they hold at a finite set of points spread evenly over the relevant portion of the age time plane. If we do this we'll get a set of constraints $\mathbf{C}\boldsymbol{\beta} \geq \mathbf{b}$.

Here's an example of the fit that results from applying exactly this method to age structured data simulated for an 11 stage age structured population:



While this figure shows the corresponding population and death rate surface estiamtes ($\hat{\eta}(a,t)$ and $\hat{\mu\eta}$):



In this example the smoothing parameter was estimated for the unconstrained problem (i.e. populaiton and death rate allowed to be negative!) and then this smoothing parameter estimate was used to refit the model subject to constraints, by quadratic programming.

4 Non-linear models

So far the biological models used have not contained much mechanism beyond basic demography, and it is largely this simplicity that has enabled them to be fitted to data using linear modelling methods. In this section we consider how to deal with non-linear models using approximating linear models, and hence how we can begin to fit quite mechanistic ecological dynamic models to data. As in previous sections, it's easiest to start with the general statistical problem first, and then to examine some ecological modelling examples.

4.1 Fitting non-linear models using linear approximations

Recall the geometric picture of regression from the first lecture. We can consider an n dimensional vector of data $\mathbf{y}$ as a single point in an n dimensional space. The linear model defines a flat surface within this space, and least squares estimation amounts to finding the orthogonal projection of $\mathbf{y}$ onto that surface. In this way of looking at model fitting a non-linear model is a model defining a curved surface within the data space, and least squares model fitting amounts to finding the model parameters corresponding to the point on the model surface that is as close as possible to the data. One way to approach this problem is to iteratively approximate the non-linear model in the vicinity of the current parameter estimates, by a linear model. The best fit parameter estimates for this linear model give updated estimates of the parameters of the non-linear model. To facilitate the following discussion I'll assume that the model being fitted has the general form:

$$\mathbf{Y} = \boldsymbol{\mu}(\boldsymbol{\beta}) + \boldsymbol{\epsilon}$$

where $\boldsymbol{\mu}$ is a vector valued non-linear function of the parameters $\boldsymbol{\beta}$. The following figure attempts to illustrate how this non-linear model fitting works geometrically, in the case of a one parameter model fitted to 2 data.

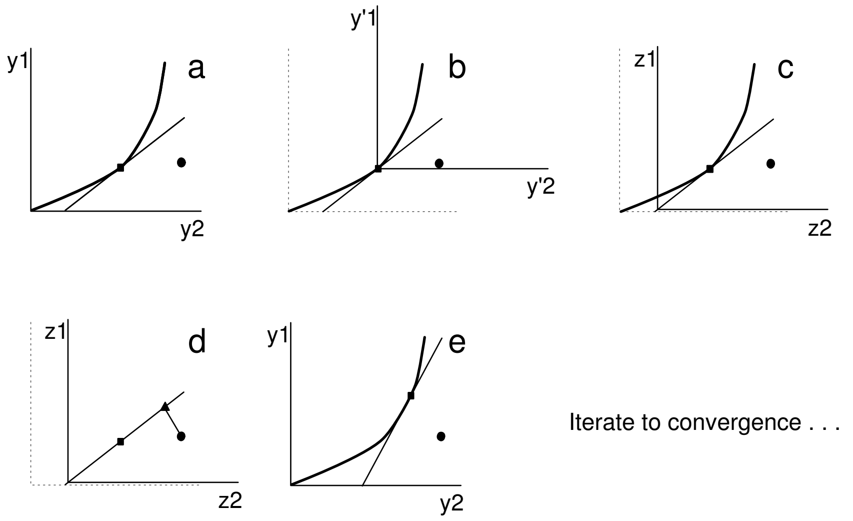


Figure a shows the basic setup: the solid circle represents the location of the data y_1, y_2, within the "data space". The thick line represents the model manifold within that space: as the model parameter β is changed the fitted values (μ_1, μ_2) trace out this curve. The solid square gives the location $(\mu_1^{[k]}, \mu_2^{[k]})$ on the manifold according to the current parameter estimate, $\beta^{[k]}$. The thin straight line is the tangent to the model manifold at $(\mu_1^{[k]}, \mu_2^{[k]})$. We would like to find the point on the model manifold (thick curve) that is closest to the data (solid circle). The remaining figures illustrate how we can improve the estimate of the model parameter, β. In figure b the axes are transformed linearly so that the origin is at $(\mu_1^{[k]}, \mu_2^{[k]})$. In figure c a further linear transformation is made so that $(\mu_1^{[k]}, \mu_2^{[k]})$ is given by $\beta^{[k]}$ multiplied by the tangent vector. The tangent is now just like a linear model, and we can improve our β estimate by taking the least squares projection from y_1, y_2 onto the tangent (shown as a triangle in figure d), which will immediately yield a new β estimate: $\beta^{[k+1]}$. Returning to the original data space and feeding $\beta^{[k+1]}$ into the non-linear model yields a new estimated $(\mu_1^{[k+1]}, \mu_2^{[k+1]})$, as shown in panel e, and the whole process can then be repeated again until the estimates converge.

From the geometry of the situation three things should be clear:

- The method will always converge if the model manifold is smooth and the initial parameter estimate is close enough to the optimum estimate, but will not necessarily converge from all possible initial estimates.
- Convergence should be straightforward if the model manifold is relatively flat.
- Convergence is likely to be quicker if the datapoint is close to the model manifold.

It should also be clear that the method generalizes in a straightforward way to higher dimensional data spaces and models. In particular if we have n data then the data space is simply n dimensional. More than 1 model parameter simply implies that the model manifold is now some multi-dimensional surface within the dataspace, while the tangent is a linear subspace of the model space.

Given data $\mathbf{y}$ and a model that predicts the expected values, $\boldsymbol{\mu}$, of $\mathbf{y}$, given some parameter $\boldsymbol{\beta}$, here is what is done:

1. Obtain an initial parameter guess $\boldsymbol{\beta}^{[1]}$, set $k = 1$ and iterate the following steps:
2. Use the non-linear model to obtain $\boldsymbol{\mu}^{[k]}$ using $\boldsymbol{\beta}^{[k]}$. Also obtain the matrix $\mathbf{J}^{[k]}$ where:
$$J_{i,j}^{[k]} = \left.\frac{\partial \mu_i}{\partial \beta_j}\right|_{\boldsymbol{\beta}^{[k]}}$$
(the columns of $\mathbf{J}^{[k]}$ are the basis vectors of the tangent space).
3. Form "pseudodata" $\mathbf{z}^{[k]} = \mathbf{y} - \boldsymbol{\mu}^{[k]} + \mathbf{J}^{[k]}\boldsymbol{\beta}^{[k]}$ (this is the location of the data vector relative to the transformed axes in figure c).
4. Find the $\boldsymbol{\beta}$ minimising the linear least squares objective: $\|\mathbf{z}^{[k]} - \mathbf{J}^{[k]}\boldsymbol{\beta}\|^2$: this is $\boldsymbol{\beta}^{[k+1]}$.
5. If $\boldsymbol{\beta}^{[k+1]}$ differs from $\boldsymbol{\beta}^{[k]}$ by more than some tolerance, increment k by one and return to 2. Otherwise stop iterating.

This approximating linear model approach is appealing for several reasons:

- It's completely straightforward to substitute a penalized and/or constrained least squares problem at step 4 if we are working with a penalized or constrained non-linear model fitting problem.
- Any smoothing parameters can be selected at each step by applying the methods for penalized linear models to the approximating penalized linear model.
- At convergence we have an approximating linear model that can be used for approximate inference. For example, in the absence of penalties or constraints: $\mathbf{V}_{\hat{\beta}} \approx \left(\mathbf{J}^T\mathbf{J}\right)^{-1}\sigma^2$.

Its main problem is that we must calculate $\mathbf{J}$ at each step of the iteration. In the context of dynamic model fitting there are two choices for calculating $\mathbf{J}$. The first is that $\mathbf{J}$ can be approximated numerically by finite differencing. This requires careful numerical analysis, but has the advantage of minimising the amount of work that the modeller has to do analytically with the model. Details of this approach are given in Wood (2001). The second approach is to derive a system of equations which can be solved to obtain the required derivatives: these equations are sometimes referred to as "adjunct equations" or the "adjunct model", and sometimes as the "sensitivity equations". The advantage of the latter approach is that the numerical analysis is no more difficult than the numerical analysis required to solve the original model (and usually results in better **J** estimates). The disadvantage is the amount of work required to derive and check the extra equations, particularly since checking will require the use of finite difference schemes!

In practice for a lengthy project, in which the major focus is on model fitting with one model, the sensitivity approach is probably preferable. For smaller projects, possibly involving many models, the finite differencing approach is probably more sensible. In these notes I'll give an example of the sensitivity equation approach.

4.2 A non-linear population dynamic modelling example

To see how the **J** calculation works in practice, consider the example of a very simple structured population model, in which the population of stage i is given by:

$$\frac{\mathrm{d}n_i}{\mathrm{d}t} = r_i - g_i n_i - \mu_i n_i \tag{2}$$

where the g_i are assumed known,

$$r_i = \begin{cases} \beta_1 + \beta_2 \exp[-(t-\beta_3)^2\beta_4] & i = 1 \quad (say) \\ g_{i-1}n_{i-1} & i > 1 \end{cases}$$

and, for example,

$$\mu_i = \begin{cases} \beta_5 & i \le 6 \\ \beta_6 & i > 6 \end{cases}$$

The derivation of the sensitivity equations is straightforward, but tedious. Both sides of (2) are differentiated with respect to β_j, for each i, and since in general:

$$\frac{\partial}{\partial x}\frac{\partial y}{\partial z} = \frac{\partial}{\partial z}\frac{\partial y}{\partial x}$$

we obtain differential equations for the way in which $\mathrm{d}n_i/\mathrm{d}\beta_j$ changes through time. For example:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial n_1}{\partial \beta_5} = -g_1\frac{\partial n_1}{\partial \beta_5} - n_1 - \beta_5\frac{\partial n_1}{\partial \beta_5} \quad \text{and} \quad \left.\frac{\partial n_1}{\partial \beta_5}\right|_{t=0} = 0$$

while for $1 < i \le 6$

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial n_i}{\partial \beta_5} = g_{i-1}\frac{\partial n_{i-1}}{\partial \beta_5} - g_i\frac{\partial n_i}{\partial \beta_5} - n_i - \beta_5\frac{\partial n_i}{\partial \beta_5} \quad \text{and} \quad \left.\frac{\partial n_i}{\partial \beta_5}\right|_{t=0} = 0$$

and for $i > 6$:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial n_i}{\partial \beta_5} = g_{i-1}\frac{\partial n_{i-1}}{\partial \beta_5} - g_i\frac{\partial n_i}{\partial \beta_5} - \beta_6\frac{\partial n_i}{\partial \beta_5} \quad \text{and} \quad \left.\frac{\partial n_1}{\partial \beta_5}\right|_{t=0} = 0$$

The key point here is that each derivative $\partial n_i/\partial\beta_5$ is just the state variable in an ordinary differential equation, and by solving the system of equations (numerically) we can obtain $\partial n_i/\partial\beta_5$ at any time, and hence one column of **J**.

Notice that for every parameter to be fitted we obtain a system of equations of about the same size and complexity as the original model: this is typical. I won't derive the rest of the equations here!

Wood (2001,1999) gives several non-linear modelling examples (some of which also involve constraints and penalties).

Simon N. Wood
16 December 2001