A simple guide to Generalized Additive Models (GAM): theory and applications for R statistical package

Version 8, developed for OC549 course, OSU Corvallis
Spring, 2018

Lorenzo Ciannelli
College of Earth, Ocean and Atmospheric Sciences
Oregon State University, Corvallis, OR, USA
Email: lciannel@coas.oregonstate.edu

Table of contents

Chapter 1: General overview and motivation

A Generalized Additive Model (GAM) is a nonparametric regression technique, used to inspect the relationship between observed (Y or response) and design variables (X or covariates). Ideally, regression techniques should fulfill three goals: 1) *description*, the model should describe how Y change as a function of X; 2) *inference*, it should be possible to assess the effect that a design variable X_j, has on Y; 3) *prediction*, we would like to use the found model to predict Ys at new values of Xs. Linear models excellently fulfill all of the regression requirements. Given a set of Ys and Xs, the linear model can be written as:

$$y_i = \alpha + \beta_1 x_{1,i} + \beta_2 x_{2,i} + ... + \beta_p x_{p,i} + e_i$$

or more compactly:

$$Y_i = \alpha + \sum_{j=1}^{p} (\beta_j X_{ji}) + e_i \qquad 1$$

where α is the intercept, β_j are regression coefficients (slopes) and p is the number of design variables included in the analysis.

Linear models are based on the important assumption that they are linear in the coefficients. Alternative regression techniques that do not assume linearity include parametric nonlinear regression, multidimensional smoothing and GAM (Generalized Additive Models).

- Parametric nonlinear regression can be very useful. However, the underlying functional forms between the Xs and the Ys should be *apriori* specified. Very often we may lack the necessary knowledge to do so.
- Smoothing techniques, like locally weighted regressions (LOESS), are nonparametric in the functional form, in that one does not need to specify the relationships between the predicted and the design variables. However, as we will examine later, all smoothing techniques apply a weighting scheme in the vicinity of the value to be predicted, and in multidimensional smoothing this presents several challenges (i.e., *curse of dimensionality*). Moreover, when more than two covariates are included in the smoother, the resulting model does not easily allow inference, as it is difficult to visualize the response for dimensions higher than two.
- GAM are nonparametric and additive regression models. Thus they enjoy the advantages of nonparametric (e.g., no need to *apriori* specify the functional forms) and additive techniques. The

advantage of additivity is that of easy inference, due to the fact the effect of a certain covariate (e.g., positive, negative, neutral, nolinear, etc.) is the same regardless of the values assumed by the remaining covariates. The con of GAM is that of still being an approximation of the true relationship between the Xs and the Ys, as the condition of additivity need not be satisfied by the data. However, just like for linear models, interactions terms may be included in a GAM to circumvent the problem of nonadditivity (see section on nonadditivity).

GAM shares several similarities with linear regression models. Given a set of covariates (Xs) and response variables (Ys), a GAM model can be formalized as:

$$Y_i = \alpha + \sum_{j=1}^{p} g_j(X_{ji}) + e_i \qquad 2$$

Note that the difference between the linear and the GAM is that in the latter the terms $a_j X_{ji}$ are replaced by 'smooth functions' g_j. This is what makes GAM nonparametric; i.e., the effect of single design variables are modeled by nonparametric smooth functions. 'Smooth' means that the function $g_j(X_j)$ is continuous (no jumps), and it has continuous first and second derivatives (no abrupt change of the slope). Smooth functions play a central role in GAM and will be treated in the next chapter.

Chapter 2: Smoothing

A smoother is a function (g) describing the relationship between a set of design Xs and response variables Ys. Smoothers can be applied to multidimensional data (multiple Xs), however for the time being we limit our discussion to one-dimensional smoothing techniques (a.k.a., scatterplot smoothing), since these are most commonly used within the GAM algorithm[1]. As already mentioned above, an important feature of smoothers is that of being nonparametric. Thus smoothers reverse the typical deductive approach of parametric regression: instead of assuming a functional form between X and Y we let the data tell us the shape of that function. Smooth (i.e., the collection of points estimated by a smoother) can then be used to infer the underlying 'true' parametric functional link between X and Y, or simply to predict Ys. We shall mainly consider nonparametric smoothers such as a moving average, which do not require any specification on the relationship between the Xs and the Ys, aside form their actual values.

An important concept that applies to all smoothers is the tradeoff between *smoothness* and *roughness*. The smoothness is a measure of how straight the resulting prediction is, while the roughness is a measure of the wigliness (or bumpiness). Potentially, a line that join all consecutive $[x_i,y_i]$ observations with straight segments is a smoother. Such line will indeed be very rough. On the other hand, a nearly straight line through a scatter plot has low roughness and high smoothness. Field observations contain noise, so it is important that the smooth does not connect all points in the scatter plot; otherwise it will fit the underlying pattern and noise at the same time. However, it is also important that the smooth is not too coarse; otherwise it may not convey much about a potential nonlineariry among the studied variables. Thus, the tradeoff between smoothness and roughness becomes very important.

For all smoothers, the aforementioned tradeoff is regulated by the *smoothing parameter*. In the case of a running mean, the smoothing parameter is the number of observations included in the averaging that are the vicinity of the point that we are interested in predicting (target point). The resulting output of a running mean will be less jagged if more points are included in the average, while the opposite will hold for fewer points (Fig. 1).

[1] Two-dimensional smoothers are also possible within the GAM

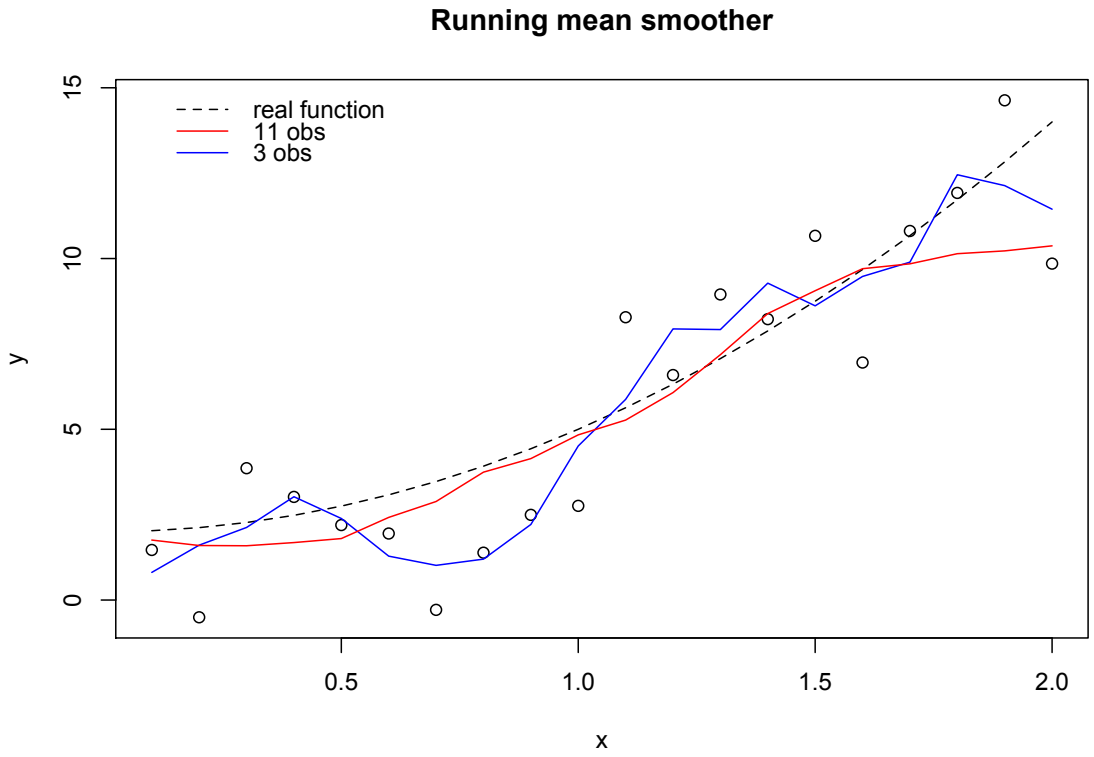


Fig. 1. Example of a running mean smoother with 3 (blue line) and 11 (red line) data observations included in the average. Note that the resulting line is smoother when more observations are included in the running mean. Thus, the *number of observations* is the smoothing parameter of a running mean.

Expectedly, the smoothing parameter is the single most important number characterizing a smoother and its value is critical in establishing how closely the smooth will follow the data. We will get back to the issue of how to objectively determine a smoothing parameter, after brief description of the some popular smoothing techniques.

Smoothing techniques: LOESS and splines

There are many types of smoother, and we focus our attention to those that are most commonly used in the GAM algorithm, namely Locally Weighted Regressions (LOESS) and splines[2].

LOESS are weighted polynomial regressions applied through a set of Xs and Ys. The building polynomials can be of several degrees, but in R only the first (linear) and second (quadratic) degree polynomial are allowed. LOESS adopt a weighting scheme (i.e., the kernel) on the data. The weights are typically a function of the distance between the target point

[2] GAM in S-Plus can take both splines and LOESS, while GAM in R can only take splines

and its neighboring observations. A common kernel is defined with *tri-cube* weight functions (i.e., weights that decrease as a function of the cube of the distance between the neighbor and the target), but other schemes are also possible. The extent of the weighting window, on either side of the target point, is regulated by the *span*, expressed as the proportion of observations that are included in the regression. A span can range from 0 to 1, and the closest it will get to 0 the more locally-weighted and wigglier the smooth will be. A span of 1 signifies that the full data set is included in the analysis, and the smooth will come to resemble a polynomial regression (Fig. 2). Thus the span is the smoothing parameter of the LOESS, as it regulates its degree of roughness vs smoothness (i.e., the famous tradeoff).

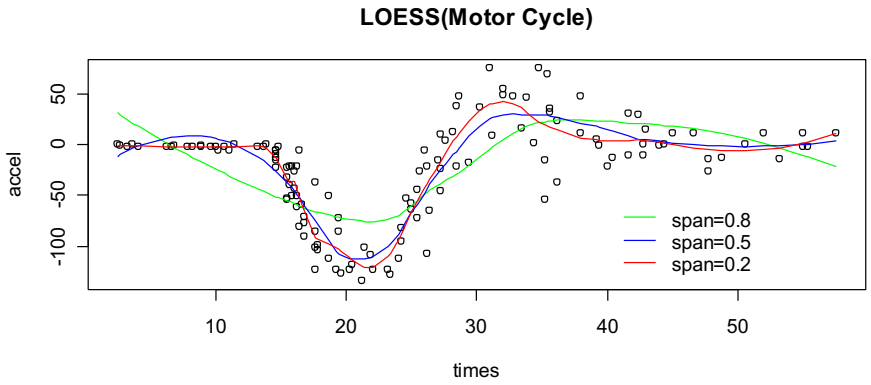


Fig. 2. Motorcycle data fitted with a LOESS. Different colors represent different span and different degree of smoothing.

SPLINES are piecewise polynomial functions, which are joined (i.e., continuous) at the boundaries, with continuous first and second derivative (i.e., smooth). The number of piecewise polynomial regressions (k+1) used to form a spline depends on the number of points or *knots* (*k*) chosen along the dimension of X. A high number of knots will result in a wigglier smooth. Thus *k* represents a smoothing parameter of a spline (Fig. 3).

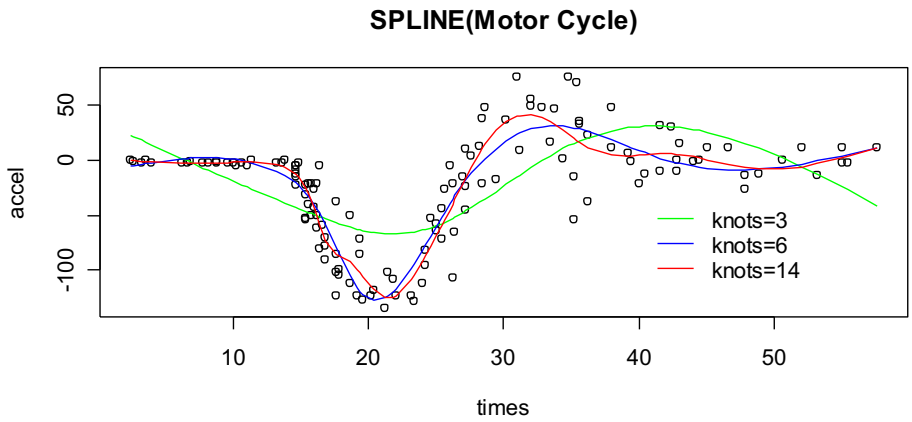


Fig. 3. Motorcycle data fitted with a SPLINE. Different colors represent spline built with different numbers knots.

Among splines, a predominant role is taken by *Natural Cubic Splines* (NCS), typically used in the GAM algorithm of S-plus and R. Given an interval $[a,b]$ within X, and a set of knots $[k_1, k_2, ..., k_n]$ such that $[a < k_1 < k_2, ..., < k_n < b]$, a function g will be defined a NCS if the following criteria are fulfilled:

1) within each interval $[a, k_1]$, $[k_1,k_2]$, ..., $[k_n,b]$, the function g is a cubic polynomial
2) the function g, and its first (g') and second (g'') derivatives are continuous (i.e., no visual discontinuities or bumps) at the knots k_i
3) the smoother is linear at the boundaries, i.e., g' and g''' are zero at the boundary
4) the knots are located at the unique values of X.

NCS are very popular in many different fields. In simple words, NCS are the best compromise between smoothness and roughness, when the smooth is forced to pass along a set of *apriori* specified knots. This notion has practical ramifications. In fact, the shape of a flexible piece of wood (i.e, a spline) joining two points a and b, and forced to pass through a series of knots within the interval $[a,b]$, is that of a NCS that minimizes the roughness penalty function (see next section). Carpenters use NCS all the times!

A valuable way of representing splines is through the use of *basis functions*. We briefly describe this approach here, as it will be relevant to better understand the how GAM are implemented in R. Given a function $f(x)=y$, it is possible to represent it through a set of basis functions b_j in the following fashion:

$$f(x) = \sum_{j=1}^{m} \alpha_j b_j(x) \qquad 3$$

where α_j are unknown coefficients, b_j are the basis functions and m is the maximum number of basis function used to represent $f(x)$. The coefficient m indicates the *dimension of the basis functions*. In general, the higher the m the more jagged the resulting smooth will be. Suitable set of basis functions are, for example, $b_1(x) = 1$, $b_2(x) = x$, $b_3(x) = x^2$, …., $b_m(x)=x^{m-1}$. Note that, with this set of basis, if we choose $m=2$, the eq. 3 will simply amount to a first degree polynomial (linear regression):

$$f(x) = \beta_1(1) + \beta_2(x).$$

When m=3, eq. 3 will be a second degree polynomial, and so on for higher dimensions. Now it is clear why an increase of the bases dimension results in wigglier functions.

Another useful set of basis functions is that of regression cubic spline. Recall that cubic splines are piecewise polynomials, joined at the boundaries, with continuous first and second derivative. Given a set of knots (m), within the range of the X variable, a set of natural cubic spline basis, meeting the criteria of a cubic spline (i.e., continuous and smooth), can be composed as follows: $b_j(x)=|x-x^*_j|^3$, *for* $j = 1, …., m,$ and $b_{m+1}(x)=1$, $b_{m+2}(x)=x$. As an example, for $m=3$ (i.e., 3 knots within the range of X), the cubic spline basis is written as:

$$g(x) = \alpha + bx + \beta_1 |x - x_1^*|^3 + \beta_2 |x - x_2^*|^3 + \beta_3 |x - x_3^*|^3$$

Thus, fitting a cubic spline is equivalent to fitting a multiple liner regression with coefficients (α, b, β_1, β_2, β_3) and design variables (1, x, $|x-x_1|^3$, $|x-x_2|^3$, $|x-x_3|^3$). It is important to realize that the dimension of the cubic spline bases depends on the number of knots used to form the spline. In addition, just like in the polynomial case, the higher is the dimension of the spline basis, and the wigglier (rougher) the resulting smooth can be.

We note that in the above formulation of spline basis there are 5 parameters and 3 knots, resulting in a model with 5 degrees of freedom (df, typically in regression the df are equal to the number of model parameters). However, NCS have the additional constraint of being linear beyond the data range. This is achieved by imposing the coefficients of x^3 and x^2 equal to 0 when $x < x_1^*$ or when $x > x_3^*$ (boundaries). By some algebraic passages it can be seen that at the boundaries the coefficients of x^3 are ($\beta_1+ \beta_2 + \beta_3$), and the coefficients of x^2 are ($x_1^*\beta_1 + x_2^*\beta_2 + x_3^*\beta_3$). By

setting both terms equal to 0 we loose two additional df resulting in a model with as many df as knots. In the next chapter we will introduce an additional constraint on GAM (the identification constraint) which results in the loss of one more df: df = number of knots -1.

Figure 4 shows four curves obtained from the spline basis $|x - x_1^*|^3$ formulation. In each plot it is assumed only one knot within the range of x. As mentioned above, carpentry splines are flexible pieces of wood used to join two points that do not rest on a straight line. Analogously, the curves in Fig. 4 can be viewed as the building blocks of a mathematical spline.

```
> #A function to draw spline basis
> sp.base<-function(a,b,k){
+ x<-seq(a,b,length=100)
+ y<-abs(x-k)^3
+ plot(x,y,main=paste('knot location is',k),xlim=c(a,b),type='l')
+ abline(v=k,lty=2)}
>
> par(mfrow=c(2,2))
> sp.base(0,1,0.1)
> sp.base(0,1,0.3)
> sp.base(0,1,0.5)
> sp.base(0,1,0.8)
```

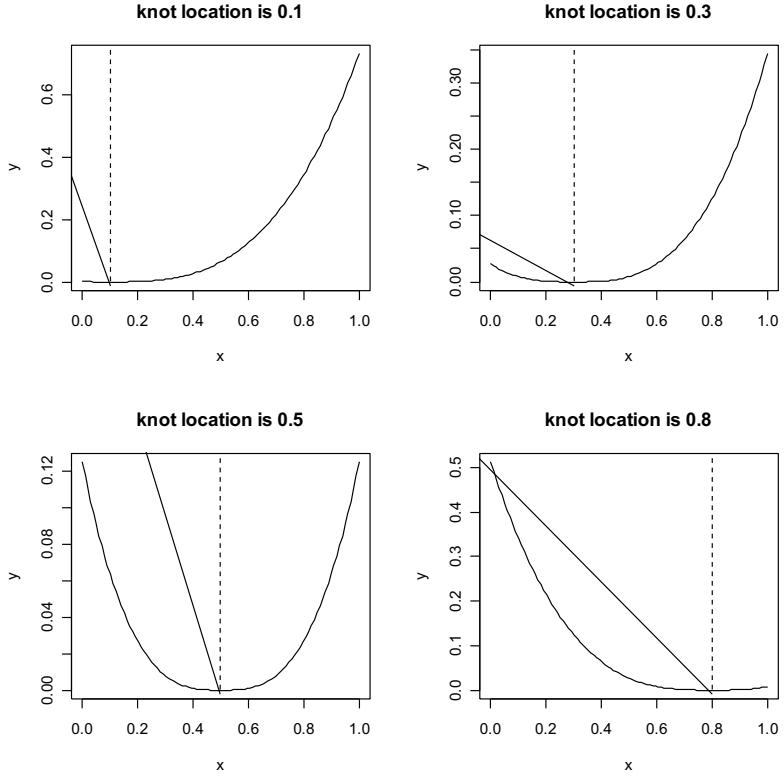


Fig. 4. Splines basis with knots identified by the vertical dotted line.

Smoothing parameter

The smoothing parameter, whether measured as span of the data (LOESS), number of knots (spline), or number of observations (running mean), is the most critical parameter of a smoother. Ideally, one should select a smoothing parameter that takes into account both the goodness (or badness) of the fit and the roughness of the smooth. Excessively rough and excessively poor fits should be penalized. The achievement of an objective balance between these two contrasting needs is the context within which the roughness penalty approach was developed (Green and Silverman 1994). In words, the roughness penalty approach penalizes the goodness of fit of a smoother (i.e., sum of squares) with a measure of its wiggliness (i.e., second derivative).

Given a smoother *g,* defined within the interval [*a,b*], a good measure of its wiggliness is the integral of the squared second derivative:

$$\int_a^b \{g''(x)\}^2 \,. \qquad 4$$

It is intuitive to see why the second derivative is a good measure of function wiggliness: the first derivative is a measure of the rate of change of a function and the second derivative is a measure of the acceleration (the change of the change). Thus, the sum (i.e., integral) of the squared second derivative is a good indicator of how changeable a function is, and also of its wiggliness. It is interesting to know that in physics the second derivative is used to measure the strain of a flexible piece of wood (i.e., spline) when forced to join a set of points that do not lie on a straight line.

Knowing how to quantify the wiggliness of a smooth allow us to take the final step in estimating the roughness penalty function [*P(g)*]. Precisely,

$$P(g) = \sum_{i=1}^{n} \{Y_i - g(x_i)\}^2 + \lambda \int_1^n \{g''(x_i)\}^2 \qquad 5$$

Good compromises between roughness and smoothness result in low penalty function. In 5, the first term measures the sum of squared errors between the predictions of the smooth [g(x)] and the actual observations (Ys). The second term (penalty term) measures the curvature of *g*. The parameter λ plays a central role, since it specifies how much weight the roughness penalty is going to get. If λ is very high, the penalty term also gets a very high weight, and *g* will be forced to have a very small

curvature, approaching a straight line. On the other hand, when λ is very low, the sum of square term gets a very high weight, and g will be forced as close as possible to the actual observations (Ys), ultimately connecting all of them. Hence, very low λ result in highly wiggly smooth, while the opposite holds for very high λ. For this reason λ is also a smoothing parameter, in the sense introduced for smoothers. Unlike other smoothing parameters defined above, whose units can change according to the type of smoother considered (i.e., span of the entire data, number of knots, number of observations), the units of λ are independent from the type of smoothing technique. However, its value will depend upon the variables under scrutiny.

It is reasonable to expect that there is some link between λ and the smoothing parameter of a smoother, since they both regulate the same tradeoff. For example, if the smoother is a running mean, then setting λ is equivalent to specifying the number of data points that are included in the calculation of the mean.

It can be proved mathematically[3] that given a set of observations [X,Y], and a smoothing parameter λ, the the minimizer of Eq. 5 is a NCS with knots at the unique values x_i. This partly explains why NCS are useful smoothing technique.

Automatic selection of the smoothing parameter λ

The present topic is that of finding the λ that minimizes the quantity in Eq. 5. This can be done using the Cross Validation (CV) or the UnBiased Risk Estimator (UBRE).

The CV criterion is based on the assumption that a good model should be able to well predict new observations. If the model is just fitting noise (i.e., too rough), when extrapolated to new observations will perform very poorly. Analogously, a simple model (i.e., smooth line) will also perform poorly upon extrapolation if the relationship among the variables is highly nonlinear. In practice, and regretfully, when we fit a model to a set of data there are no more observations left to be considered as 'new'. To circumvent this problem we can fit the model to all the available observations, save one. Then we assess how well the fitted model predicts the left out observation, for example by measuring the squared predictive error (i.e., squared distance between the model prediction and the new observation). The same routine is repeated for as many data points as there are observations in the original data set, and the model CV is found by taking the average of all the squared predictive errors. The so found CV will be a function of the λ used in the Eq. 5 CV(λ). Formally, what just said in words can be very compactly written as:

[3] Interested readers should refer to Green and Silverman 1994

$$CV(\lambda) = n^{-1}\sum_{i=1}^{n}\left\{Y_i - g^{-i}(x_i,\lambda)\right\}^2 \qquad 6$$

where the term $g^{-i}(x^i, \lambda)$ is the smoother prediction of the *ith* observation, obtained from a model calibrated on all the available data set, save the *i*th observation. The objective is that of minimizing the smoother CV by changing the λ.

Figure 5 shows an example of why the CV efficiently measures the compromise between roughness and smoothness. A very rough function (one with a very low λ), fitted to all observations can predict very well the within-sample observations. However, when one observation is dropped from the data set, the rough function can considerably change its shape, so to achieve a good fit with the remaining data cases, while the smooth function (one with high λ) may remain largely unchanged. In such circumstances, the prediction error of the rough function is actually larger than that of the smooth function (Fig. 5).

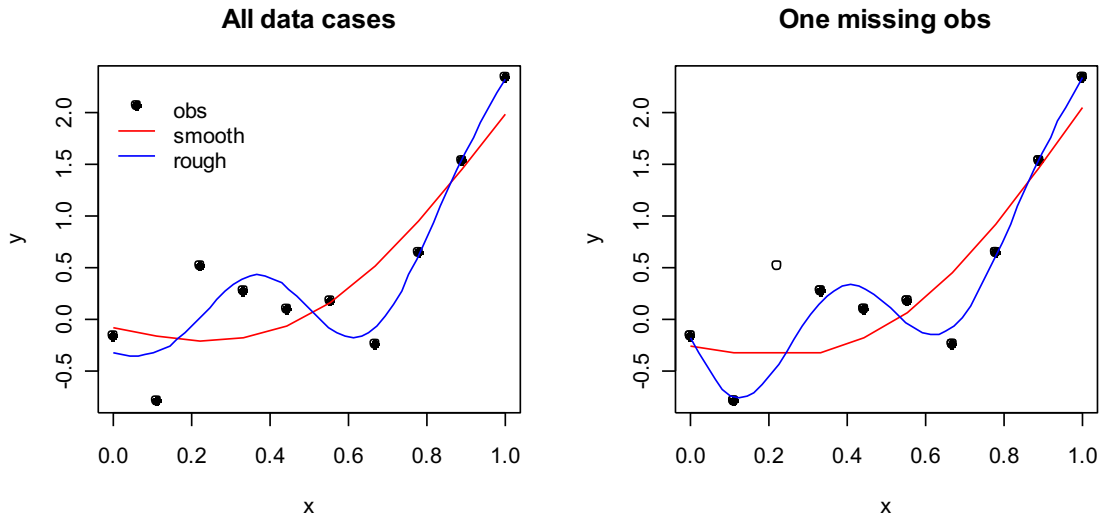


Fig. 5. Example of the prediction error of rough (blue) and smooth (red) functions on within-sample and out-of-sample observations. The entire data set has been used to parameterize the smooths shown on the left plot, while one observations (open circle) has been removed to fit the smooths shown on the right plot.

The automatic selection of a smoothing parameter using CV is very computationally intense. The number of computations is even greater when more than one variable is included in the model formulation (as very often is the case for GAM). Fortunately, it has been proved[4] that

[4] see Green and Silverman (1994) for an in-depth description of the theorem and the proof

there is numerically simpler formula that is equivalent to Eq. 6, which does not require the fitting of n new models every time we select a new λ. By virtue of this new finding, given a λ, the CV can be calculated as:

$$CV(\lambda) = n^{-1}\sum_{i=1}^{n}\left(\frac{Y_i - g(x_i,\lambda)}{1 - S_{ii}(\lambda)}\right)^2 \qquad 7$$

where S is the *smooth matrix*, whose row are the weights applied to each Y observation to project its value into the smooth prediction [g = S(λ)Y]. S is a projection matrix and bears similarities with the *hat matrix* of the linear regression. Because of Eq. 7 the actual number of operations needed to find the smoothing parameter is drastically reduced.

Most computer packages (e.g., S-plus, R) use the generalized cross validation (GCV) as opposed to the CV, to select the λ. CV and GCV are in principle very similar. Their only difference rest on the denominator: the GCV is calculated by dividing the squared errors by the average of the quantity in the denominator of Eq. 7:

$$GCV(\lambda) = \frac{n^{-1}\sum\left(Y_i - g(x_i,\lambda)\right)^2}{\left(1 - trS(\lambda)n^{-1}\right)^2} \qquad 8$$

where *tr(S)* is the trace (sum of diagonal elements) of the matrix S. The primary advantage of using GCV as opposed to CV is computational. In fact in Eq. 8 there is no need to explicitly find all the diagonal elements of S. Only the trace of S is sufficient, which can be readily calculated. Moreover, the GCV has the desirable property of not being highly affected by very influential data cases.

A second approach used to determine the smoothing parameter is that of minimizing the UnBiased Risk Estimator (*UBRE*). The formula for the UBRE is the following:

$$U(\lambda) = n^{-1}\sum_{i=1}^{n}\{g(x_i,\lambda) - y_i\}^2 + \frac{2\sigma^2}{n}df(g,\lambda) + cst$$

where σ^2 is the noise variance, *df(g,λ)* are the degrees of freedom of the smoother given λ, and *cst* is a constant. The important feature derived from the UBRE formulation is that excessive wiggly functions (i.e., functions with high degrees of freedom) are penalized by the *df* term.

While the UBRE is efficient in compromising between roughness and smoothness, it presents the problem of the noise variance (i.e., variance

of the error terms). Typically, the noise variance is hard to estimate because we do not know what is the true function linking x and y. A typical procedure is to approximate the noise variance from the variance of multiple responses at the same or very similar values of covariates. In either case, the application of the UBRE criterion requires multiple measure of the dependent variable for each independent observation, or of many covariate points not too far dispersed from each other. Hence, very often the UBRE criterion is not applicable simply because the available sample size is too small[5].

Equivalent degrees of freedom

It is possible to express the smoothing parameter with a number that is comparable across different type of smoothers, regardless of the units involved in the observation, and regardless of the smoothing technique. The determination of such number is of great advantage because it allows the comparison of different smooth, and it gives a crude appreciation of the degree of nonlinearity required to model a set of X and Y observations. As we will see later on, one of the outputs of a GAM algorithm is the *degree of freedom* of the smooth function, for each of the term included in the model formulation. Just the knowledge of this number can reveal how nonlinear the relationship between X and Y need to be.

The number that is commonly used to compare smoothers is the E*quivalent Degree of Freedom* (EDF). In statistics there are many definition of degrees of freedom, and the one used for smoothers is similar to the one used in general regression techniques, i.e., the degrees of freedom are equal to the number of parameters needed to produce the curve. While this concept is fairly easy in parametric regression, where one can actually count the parameters (e.g., 2 for linear regression), it is not straight forward in nonparametric regression, where by definition there are no parameters involved. However, from linear regression we know that the trace of the *hat matrix* (i.e. the matrix that projects design into response variables) is also equal to the model degrees of freedom. Analogously, a useful approximation to the EDF of a smoother is:

$EDF = tr\ S(\lambda)$.

As already mentioned the maximum df of a spline are also equal to the number of knots -1.

The smoothing parameter (λ) has the greatest influence on the amount of smoothing. Thus, it can be anticipated that under some regularity

[5] In R the GCV criterion is the default for all distribution families except the Poisson and binomial, where the UBRE is used. However, it is possible to force the GCV criterion to all distribution families by setting the `scale` parameter = -1 (or any negative number).

conditions the EDF of a smoother must depend on λ. Hastie and Tibshirani (1990, Appendix B) provide a useful approximation to better understand the connection between EDF from λ:

$$trS(\lambda) = EDF \cong 2 + \sum_{3}^{n} \frac{1}{1 + \lambda p_i}$$

where $p_i = c\pi 4(i-1.5)^4$, $i = 3,\ldots,n$.
details of the above formulation do not matter, however, it is important to note that there is a unique and inverse link between the EDF and the λ:

- *low λ will result in high EDF and wiggly functions*
- *high λ will result in low EDF and less wiggly functions.*

This concept is visually shown in Fig. 6.

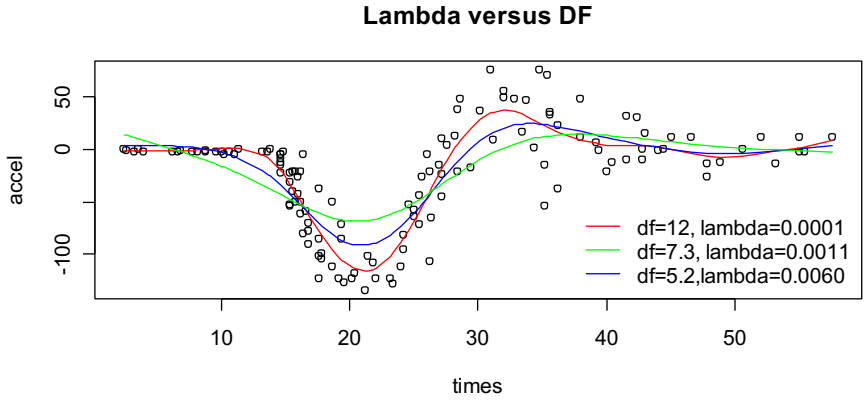


Fig. 6. Motorcycle data fitted with a SPLINE. Note the inverse relation between the degrees of freedom of the smoother (df) and the penalty coefficient λ.

Chapter 3: Additive models

Additive Models (AM) are a special class of GAM in which the error is assumed to be normally distributed and so is the response variable given the design variables being fixed. The theory covered on smoothing and penalized roughness regressions can be readily applied to the estimation of additive models. Following, we cover the case of models applied to conditionally normally distributed data (i.e., AM), to later on expand the theory for different distribution families (GAM).

An AM can be expressed as follows:

$$Y_i = \sum_{j=1}^{p} g_j(X_{ji}) + e_i$$

where g_j are smooth continuous and twice differentiable functions. Here we add another constraint on the gs, precisely:

$$\sum_{i=1}^{n} g_j(X_{ij}) = 0 \text{ for all } j\text{s.} \qquad 9$$

This is equivalent to saying that the sum of all possible predictions from every single smoother, included in a GAM, is equal to zero. This constraint is necessary to guarantee that the model is *identifiable*. In this context identifiable means that there are no two different models that can equally represent the data. If there was no such constraint on the functions g, then it can be easily seen that by adding and subtracting the same constant (c) to two distinct additive terms would result in the same prediction:

$$E(Y_i) = g_1(x_{1i}) + g_2(x_{2i}) = \left[g_1(x_{1i}) + c\right] + \left[g_2(x_{2i}) - c\right]$$

Because of the constraint imposed on the smoother predictions the model identification is guaranteed. However a df is lost. Moreover, to bring the model predictions at the same scale of the measured response, a constant (α) needs to be added to the additive predictors.

$$Y_i = \alpha + \sum_{j=1}^{p} g_j(X_{ji}) + e_i \qquad 9.1$$

Essentially α is equal to the average of the response variables.

Not all functions of an AM need to be nonparametric. It is possible to mix parametric (i.e., linear in the coefficient) and nonparametric terms. This will be shown in later examples.

Fitting a GAM is like estimating a multidimensional smoother (i.e., estimating the coefficients of a very large polynomial), after imposing additivity and the identification constraint of Eq. 9. There are two statistical software that can implement GAM: Splus and R. They differ slightly in the way GAM are estimated, with the GAM in R being slightly more elaborated than those in Splus. In the following sections we first describe the steps involved in the estimation of GAM in Splus, and then describe GAM in R.

AM in Splus

AM in Splus are estimated using the *backfitting algorithm* (BFA). This is very simple in principle, as it amounts to using partial residuals to partition the variance among all the covariates included in the model specification. The bfa is formalized as follows:

$$g_j = G_j(Y - \alpha - \sum_{k \neq j} g_k \mid x_j) \qquad 10$$

where the term *G(y|x)* indicates the smooth function of the response variable *y* given the design variable *x*. The use of the BFA is justified by the additivity imposed in Eq. 2, from which we infer that the conditional expectation of $(Y - \alpha - \sum_{k \neq j} g_k \mid x_j)$ is equal to g_j.

When estimating a multidimensional GAM, the BFA is used in an iterative fashion. Hastie and Tibshirani (1990) provide the following 3 steps for the iteration process:

1) Initialize: α=average(Y); $g_j=g_{j,0}$ j=1,….,p
2) Cycle: for j=1 to p $\{g_j = G_j(Y - \alpha - \sum_{k \neq j} g_k \mid x_j)\}$ (the smoother is estimated using the roughness penalty approach (Eq. 5)
3) Continue step 2, until the individual smoothers do not change

The constant α is specified at the very start of the BFA, typically as the mean of the predicted variables (Y). Then, all the function g_j, are set to an initial function, $g_{j,0}$. A good place to start for initial functions is the linear regression of Y on the design variables x_j. After these initial steps, the smooth functions are fitted to the partial residuals (by minimizing the roughness penalty function of Eq. 5), and the process is repeated until all the estimated smooth do not change much from one iteration to the next. In Splus (and in R) there is a degree of tolerance to the amount of change that smooth are allowed to achieve in two successive iterations. Once the target tolerance is reached, the BFA is stopped. Typically, all the smoothers converge to the desired tolerance level within less than 20 iterations. However, this may not always be the case, and in such

circumstances one can either increase the number of maximum iterations or increase the tolerance level.

The BFA, applied to multidimensional AM, would result in a computational nightmare if, along with the smoothers, also the smoothing parameter (λ) had to be estimated. Nested within every cycle of the BFA there would another loop needed to make a search grid for smoothing parameter λ. The required number of iterations would grow enormously, and we would wait hours before getting a result. Thus, in Splus GAM are fitted assuming a fixed number of degrees of freedom for each smoother. Default values are 4 for spline or a span of 0.5 for loess. In case of spline, setting the degrees of freedom to 1 would be equivalent to fitting a straight line through the data. Note that having a high number of degrees of freedom does not necessarily imply that the resulting smooth functions will be very rough, as the penalty term of Eq. 5 would downweight excessively wiggly output. However, if the data do not support a strong curvature, using an excessive number of degrees of freedom would simply be a waste of power.

In Splus it is possible to compare different models against each other (Venables and Ripley 2002). For example we could compare a linear model against several AM with degrees of freedom > 1. This operation would amount to asking weather the data support a nonlinear relationship between the design and predicted variables, and if so, in what form (quadratic, cubic, etc.).

AM in R

The automatic selection of the smoothing parameter for all model components included in the AM also necessitates the determination of a global GCV (the GCV for entire model). In fact, minimizing the GCV of each model term is not necessarily equivalent to minimizing the entire model GCV. Therefore the selection of the smoothing parameters (i.e., degrees of freedom) of each single smoother has to be done on a global measure of balance between fitting and smoothing the data. Once a suitable analytical representation of the GCV for the entire model has been found, one has to ensure that it also is efficient, so that it can be actually determined and minimized in few operations, and that the computational requirements are not excessive. The new implementation of GAM in R has overcome these problems (Wood 2000, Wood and Augustin 2002, Wood 2004). Here we briefly illustrate how the estimation problem of the multiple smoothing parameters is set up. Readers interested in getting a more in-depth exposure to this topic can read Chan (MS, pag. 22) and the cited references.

Consider an AM with two covariates (X_1 and X_2):

$$y_i = \alpha + g_1(x_{1,i}) + g_2(x_{2,i}) + \varepsilon_i .$$

We are interested in finding the shape of the two smoothers (g1 and g2) that minimize the following roughness penalty equation:

$$P(y,\lambda_1,\lambda_2) = n^{-1}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda_1 \int g_1(x_1)^2 + \lambda_2 \int g_2(x_2)^2$$

By analogy with the univariate case, given a set of smoothing coefficients (λ_1,λ_2), the solution of the penalized problem is a NCS, of the form:

$$\hat{y}_i = \alpha + a_1 + b_1 x_{1,i} + \beta_1 \mid x_{1,i} - x_1^* \mid^3 + \ldots\ldots + \beta_n \mid x_{n,i} - x_n^* \mid^3 +$$
$$+ a_2 + b_2 x_{2,i} + \gamma_1 \mid x_{2,i} - x_2^* \mid^3 + \ldots. + \gamma_n \mid x_{n,i} - x_2^* \mid^3$$

Thus, the problem of finding the smoothers of a bivariate additive model is equivalent to that of finding the linear coefficients $(a_1, b_1, \beta_1, \ldots, \beta_n, a_2, b_2, \gamma_1, \ldots, \gamma_n)$, subject to similar constraints as in the univariate case (linearity outside the data range).

The search of the smoothing parameters is also based on the optimization of the model GCV, which takes the following form:

$$GCV(\lambda_1, \lambda_2) = \frac{n^{-1}\sum\left(y_i - \hat{y}_i(\lambda_1, \lambda_2)\right)^2}{\left(1 - trS(\lambda_1, \lambda_2)n^{-1}\right)^2}$$

Thus, the main difference between AM in R and Splus is that in the former the smoothing parameter of each single smoother is automatically selected, and it is part of the model output. Moreover, GAM in R can only accept spline smoothers, while in Splus it is possible to use spline and loess. Finally, in R the maximum degrees of freedom of a spline are expressed as the dimension of the basis function used to build the spline. However, recall that for a spline its maximum df is equal to the number of knots (k) used to form the spline minus 1 (the identification constraint). In turn, the number of knots is equal to the dimension of the basis function (m). Thus, for all practical purposes, we can control the maximum wiggliness of the smooth by setting a small k.

Chapter 4: Practical GAM fitting in R

This chapter illustrates how to code a simple gam model in R. As a general convention throughout the remaining part of the notes, the code which is executable in R is preceded by the sign: >. Both, executable code and model output are in the `courier` font. Codes which are followed by the pound sign '#' are comments, and are not processed in R. Finally, a `'+'` at the start of the line is simply a carriage return character.

Note that to access the gam function it is necessary to attach the mgcv library. This can be readily done either by typing `library(mgcv)`*, or from the menu bar under* `packages, load package, mgcv`*.*

Data and functions included in these notes are also available in the accompanying script file for R, which is also provided to you.

Example 1: additive effect with two significant variables

In the following example a GAM model is fit to a simulated data set composed by two covariates.

```
> set.seed(999)
> #Setting the seed guarantees the reproducibility of the example.

> x1<-runif(100)
> x2<-runif(100)
> e<-0.5*rnorm(100)#normally distributed error, with mean=0 and sd=0.5
> y<-5*x1^3+sin(3*pi*x2)+e #real additive function plus error
> dat<-data.frame(x1=x1,x2=x2,y=y)
> pairs(dat)#pairwise scatterplot matrix of the data (Fig. 1)
```

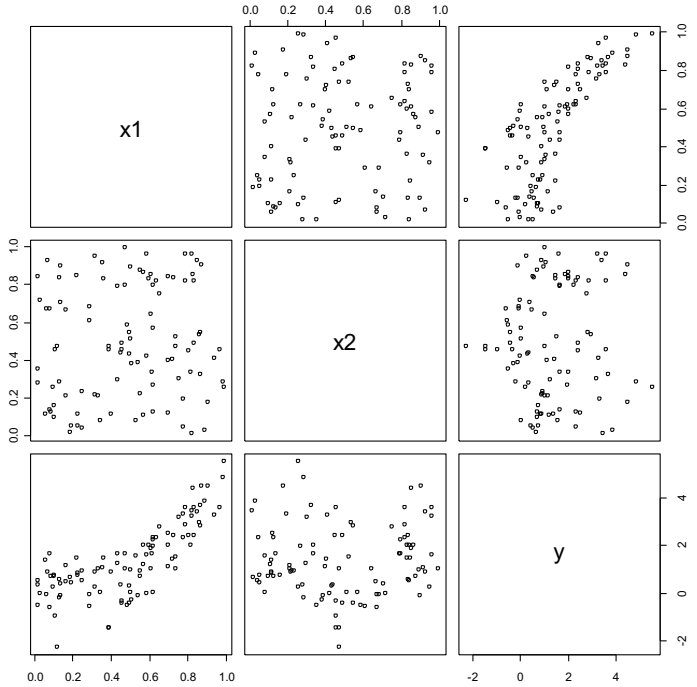


Fig. 1. Pairwise scatterplot matrix of the fictitious data set in Example 1

```
> model1<-gam(y~s(x1)+s(x2),data=dat)
```

The above line specifies the GAM model. Type `?gam` in R to access an extensive help on the function. The required arguments of the gam function include the *model formula* and the *data frame* where the variables are stored.

- The *model formula* is of the type: `y~…..`, where the left hand side identifies the response variable and the right hand side identifies the model covariates. The effect of the covariate is modelled with splines, so the `s` operator produces the spline base expansion. Besides the covariate, the `s` operator also has its own additional arguments (type `?s` in R to access the help file). An important argument is '`k`' identifying the dimension of the bases used to build the smooth term. By default this is no greater than 10, corresponding to 10 knots and *k-1* = 9 d.f. However, the maximum number can be either increased (e.g., `s(x1, k=20)`), decreased (e.g., `s(x1, k=5)`) or fixed (e.g., `s(x1, k=3, fx=T)`). Model covariates can also be entered parametrically (i.e., linearly) in which case the covariate is not preceded by the s operator (e.g., `y~s(x1)+ x2` where `x2` is the linear term).
- The *data frame* should contain all the named variables included in the model formula.

```
> summary(model1)

Family: gaussian
Link function: identity

Formula:
y ~ s(x1) + s(x2)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.36876    0.04454   30.73   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
        edf Ref.df      F p-value
s(x1) 3.136  3.907 212.26  <2e-16 ***
s(x2) 6.234  7.371  29.46  <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.912   Deviance explained =   92%
GCV = 0.22137  Scale est. = 0.19842   n = 100
```

The summary output of the GAM contains most of what we need to know about the model fit. It is divided in three sections, the *parametric*, the *nonparametric*, and the *goodness of fit* components of the model.

- The parametric section, in the case of example 1, only includes the intercept term, typically equal to the mean of the response variable (y). A standard error and a p-value testing the null hypothesis of identity to zero, is also reported. If we had included a linear term, its attributes (i.e., slope and significance of the slope term) would also be listed here.
- The nonparametric portion identifies the covariate, the estimated degrees of freedom (`edf`), and the significance (p-value) of the smooth terms. The significance is based on the null hypothesis that the smooth effect is equal to zero throughout its range.
- The last portion of the summary contains information about the goodness of fit of the model, including the R^2, the percentage of deviance explained, and the final model GCV.

```
> plot(model1,pages=1,se=T,res=T,pch=1)#covariate effect: (Fig. 2)
```

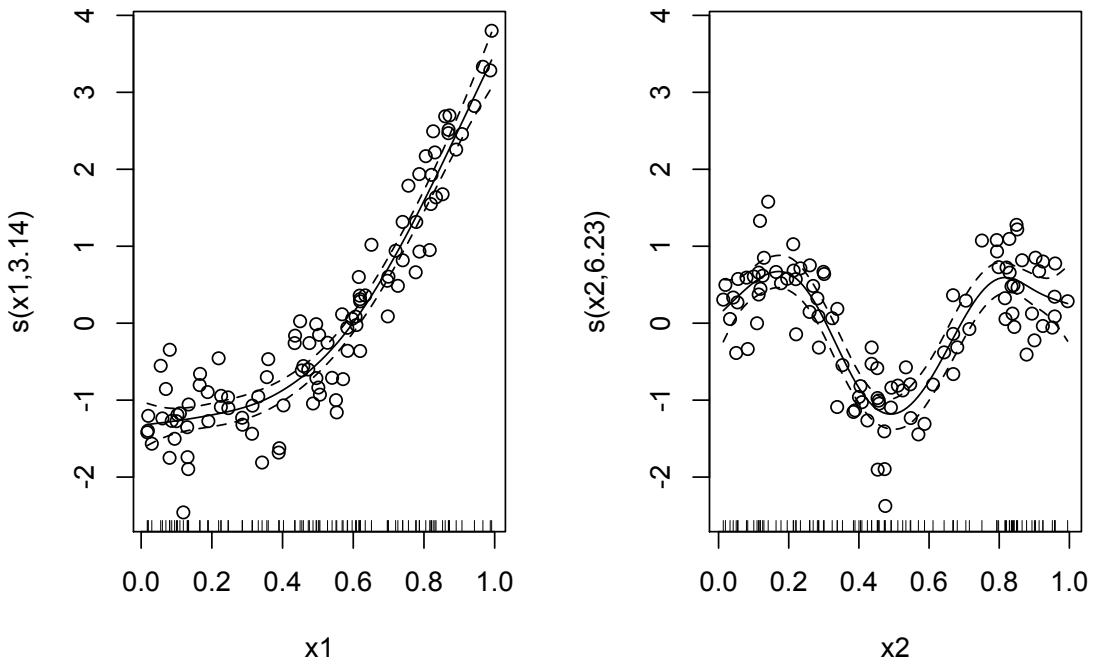


Fig. 2. Smooth term effect of the covariate (x1 and x2) from Example 1.

The y-axis in each plot is in the additive scale and it measures the effect or contribution that the covariate has on the response variable. The cumulative effect at any given point (i.e., the GAM prediction) is given by the sum of all partial effects plus a constant. The constant is the average of the response variable, given as intercept in the summary output (α). The dashed lines identify the Bayesian 95% confidence interval around the mean prediction. The dots are partial residuals calculated by adding to the effect of the concerned covariate to the residuals (see section on 'Model diagnostic'). The rug in the bottom of each plot indicates the location where the covariate measurements occur. Note how the effect of each covariate is centered around 0, to satisfy the identifiably constraint.

It is possible to access the additive predictors for each covariate. This is given in the form of a matrix with as many columns as covariates (2) and as many rows as data points (100). Here we only print the first 10.

```
> predict.gam(model1,type='terms')[1:10,]
        s(x1)       s(x2)
1  -0.9021397 -1.1241367
2  -0.0957518  0.3078121
3  -1.2722820  0.3764754
4   2.0812580  0.3693679
5   1.4666657  0.3053425
6  -1.2557740 -1.1721507
```

```
7   0.0487701 -0.4160000
8  -1.2808174  0.6430567
9  -0.8976550 -1.1694947
10  0.1339144 -0.9497385
```

The two columns give the additive effect of the first and second covariate, respectively. The sum of each i^{th} additive effect (row) with the model intercept gives the i^{th} gam predictions, as indicated in Eq. 9.1. The sum of each j^{th} (column) additive effect should instead be equal to 0, as indicated in equation 9. The additive predictors and the standard errors (access by setting `se.fit=T`) are used to draw the covariate plots shown in Fig. 2. The following is an example of how to draw the covariate plots using the additive predictors and the estimated standard errors (Fig. 3):

```
> predX1<-predict.gam(model1,type='terms')[,1]
> predX2<-predict.gam(model1,type='terms')[,2]
> seX1<-predict.gam(model1,type='terms',se.fit=T)[[2]][,1]
> seX2<-predict.gam(model1,type='terms',se.fit=T)[[2]][,2]
> par(mfrow=c(2,1))
> plot(sort(x1),predX1[order(x1)],xlab='x1',ylab='additive
+effect',type='l',ylim=c(-2,4))
> lines(sort(x1),predX1[order(x1)]+1.96*seX1[order(x1)],lty=2)
> lines(sort(x1),predX1[order(x1)]-1.96*seX1[order(x1)],lty=2)
> plot(sort(x2),predX2[order(x2)],xlab='x2',ylab='additive
+effect',type='l',ylim=c(-2,4))
> lines(sort(x2),predX2[order(x2)]+1.96*seX2[order(x2)],lty=2)
> lines(sort(x2),predX2[order(x2)]-1.96*seX2[order(x2)],lty=2)
#These 13 lines of code could be spared by simply using the plot
#function directly on the gam object…
```

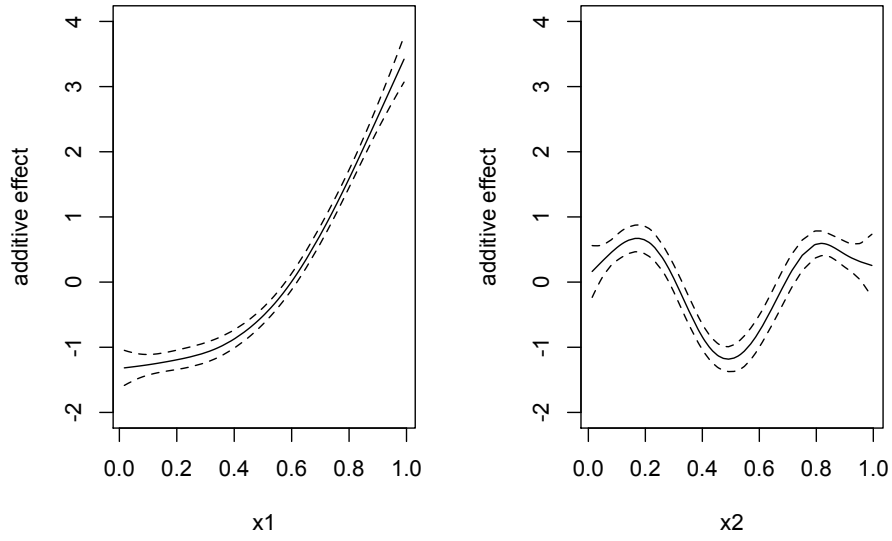


Fig. 3. Same plot as in Fig. 2, drawn from the additive terms.

The next script makes a nice 3-D graph of both the real and the GAM rebuilt functions and illustrate the use of `vis.gam` (Fig. 4)

```
> newdata<-expand.grid(sort(x1),sort(x2))
> names(newdata)<-c('x1','x2')
> realy<-(5*newdata$x1^3+sin(3*pi*newdata$x2)) #real function
> realy<-matrix(realy,100,100)
> par(mfrow=c(1,2))#Partition the graph device in two areas
> persp(sort(x1),sort(x2),realy,xlab='x1',ylab="x2",zlab='real y',
+ main='Real function',theta=30,phi=30,ticktype = "detailed",cex=1.4)
> vis.gam(model1,main='Rebuilt function',theta=30,phi=30,
+ticktype="detailed",color='bw',zlab='predicted y',cex=1.4)
```

The function `vis.gam` automatically provide the 3-D plot of a GAM. If more than 2 covariates are included in the model than the `vis.gam` function can be conditioned on one or more covariate (see `?vis.gam`)

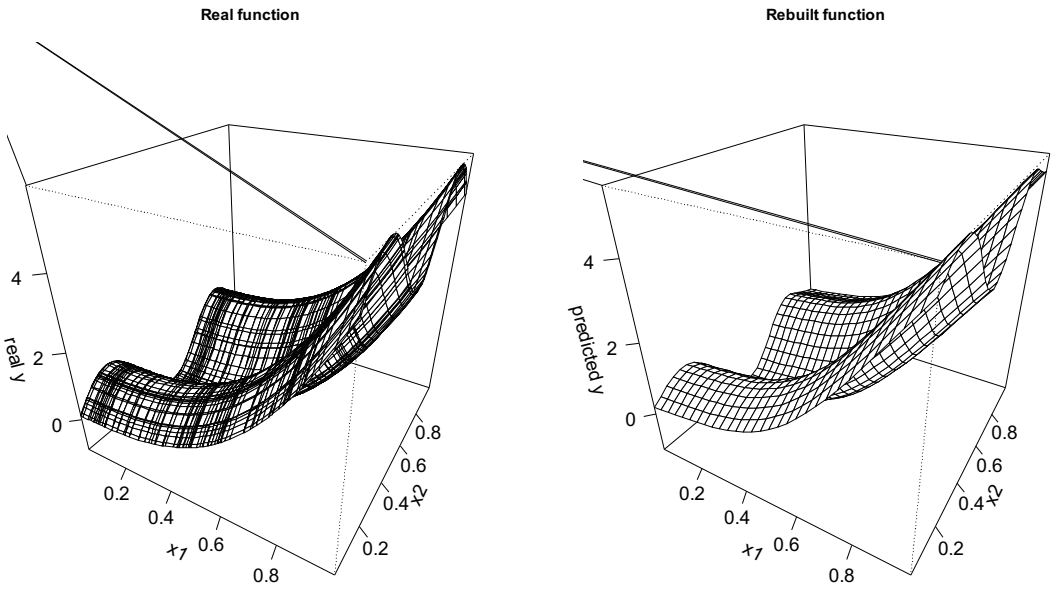


Fig. 4. 3-D representation of the real (left) and rebuilt (right) function of example 1.

Example 2: additive effect with a nonsignificant variable
The following example is similar to the one shown above, with the exception that a nonsignificant covariate (x3) is added to the model forlumation.

```
> set.seed(999)
> x1<-runif(100)
> x2<-runif(100)
> x3<-runif(100)
> e<-0.5*rnorm(100)
```

```
> y<-5*x1^3+sin(3*pi*x2)+e #note that x3 is not included in the true
#function
> dat<-data.frame(x1=x1,x2=x2,x3=x3,y=y)
> model2<-gam(y~s(x1)+s(x2)+s(x3),data=dat)#note the inclusion of x3
#as a model covariate
> summary(model2)

Family: gaussian
Link function: identity

Formula:
y ~ s(x1) + s(x2) + s(x3)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.39998    0.04909   28.52   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
        edf Ref.df      F p-value
s(x1) 6.188  7.326 74.452  <2e-16 ***
s(x2) 6.273  7.401 26.428  <2e-16 ***
s(x3) 1.335  1.589  0.265   0.775
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.886   Deviance explained = 90.2%
GCV = 0.28287  Scale est. = 0.24102   n = 100
```

From the model output it is clear that the covariate x3 does not enter significantly (p>>0.05). In addition, the GAM plot shows that the effect of x3 is essentially linear (edf=1.33), and that it is close to zero throughout its domain, thus giving another indication of non-significance.

```
> plot(model2,pages=1,se=T, residuals=T,pch=1)#Fig. 5
```

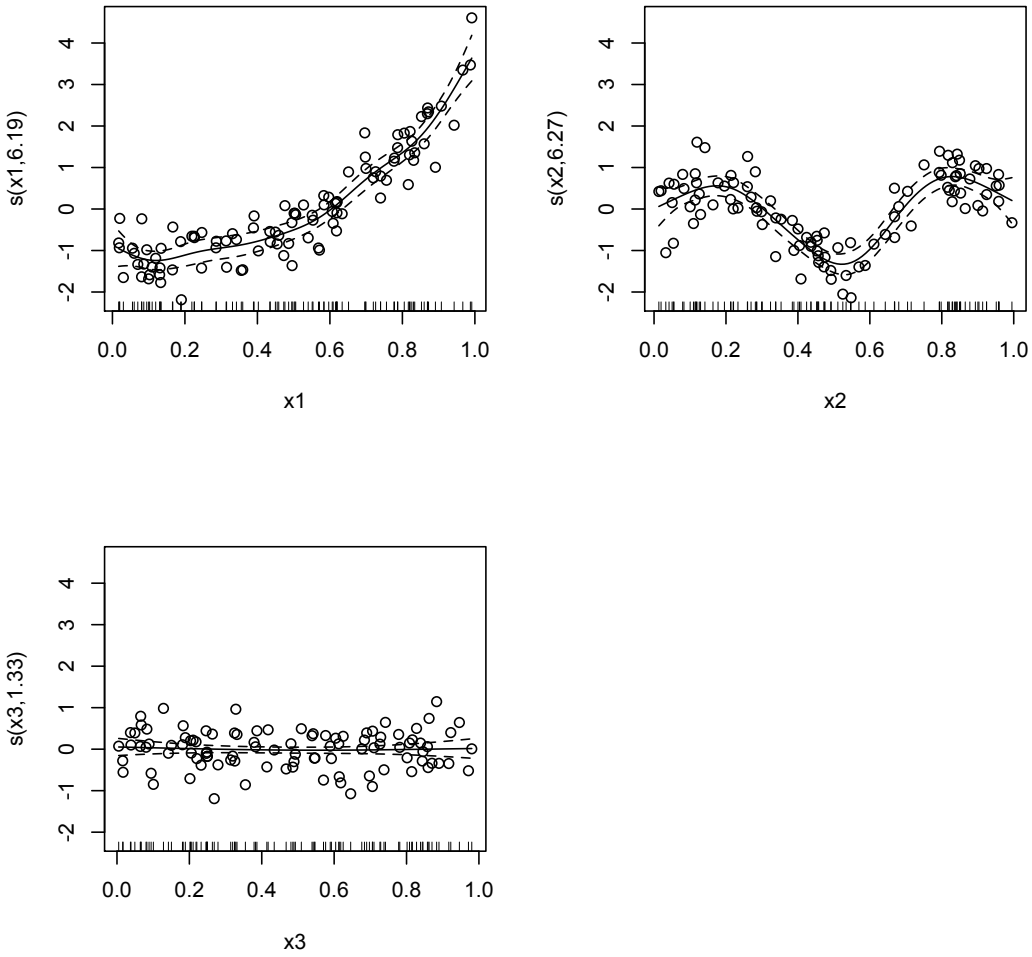


Fig. 5. Additive covariate effect of the data set in Example 2. Note that x3 has a linear effect (EDF≈1) of zero value throughout the domain, indicating a nonsignificant effect.

Chapter 5: Model selection criteria

In the example 2 of the previous chapter it was shown that the covariate `x3` was not contributing to the overall model prediction, as the confidence interval of its additive effect was 0 throughout the domain. Thus, `x3` should not be included in the final model formulation. Example 2 introduces the topic of the present chapter, precisely: Are there objective criteria for including or excluding terms from a model formulation?

The 95% confidence interval (CI) of each covariate effect (plotted by setting `se=T` within the `plot.gam` function) and the p-value play an important part in deciding when to drop a covariates. The p-value refers to the null hypothesis that the covariate has no effect (0 throughout the range), and the CI is a point-wise estimate (rather than global). A Bayesian approach is applied to obtain the covariates p-values and CI. Details of their derivation can be found in Wood 2006. The so derived CI are reliable with respect to the overall mean response (the model prediction), but are unreliable with respect to the single covariate effect. Unreliable means that the probability of the true curve to fall outside of the 95%CI is >5%. The main problem with the current implementation of the Bayesian-derived CI is that they do not properly account for the variability in the smoothing parameter (λ) (Wood 2004).

Typically, we start with a model that contains all the covariates under scrutiny, and then perform backward selection from the full model. Wood (2000) and Wood and Augustin (2002) proposed that the following questions be asked to decide when to drop a term:

1) Are the EDF for the questioned term approaching 1 (i.e., linear effect)?
2) Is the confidence interval of the term including zero throughout its domain? (i.e., high p-value)
3) Does the GCV score for the entire model drop when the term is removed?

If the answer to ALL three questions is YES, then the term should definitely be dropped. Cases in which only two out of the three criteria apply may require further judgment (see examples). Because covariate may be correlated, only one term at a time should be removed from the initial model formulation. The following example illustrates the concept of variable selection.

Example 3: selecting variables

Starting from the same model introduced in the example 2, we add a new nonsignificant variable to the GAM formulation:

```
> set.seed(999)
> x1<-runif(100)
> x2<-runif(100)
> x3<-runif(100)
> x4<-runif(100)
> e<-0.5*rnorm(100)
> y<-5*x1^3+sin(3*pi*x2)+e
> dat<-data.frame(x1=x1,x2=x2,x3=x3,x4=x4,y=y)
> model3<-gam(y~s(x1)+s(x2)+s(x3)+s(x4),data=dat)
> summary(model3)

Family: gaussian
Link function: identity

Formula:
y ~ s(x1) + s(x2) + s(x3) + s(x4)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.34020    0.05195    25.8   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
        edf Ref.df       F p-value
s(x1) 2.812  3.497 131.331  <2e-16 ***
s(x2) 5.724  6.855  23.588  <2e-16 ***
s(x3) 1.065  1.126   0.100   0.746
s(x4) 1.790  2.222   1.556   0.199
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.866   Deviance explained = 88.2%
GCV = 0.30799  Scale est. = 0.26983   n = 100
```

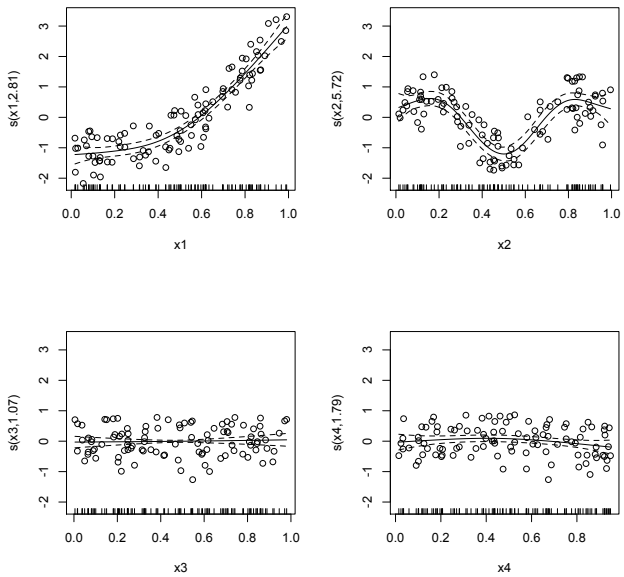


Fig. 1. Covariate effect of model 3 from Example 3.

From the model summary it is clear that the covariates x3 and x4 have EDF approaching 1 and a nonsignifcant p-value. The covariate plot (Fig. 1) shows that the confidence interval of x3 and x4 include 0 throughout their range, thus both terms could be dropped. However, because of potential correlation among covariates, we remove only one variable at-a-time, starting with the one having the least significant effect (i.e., highest p-value); in this case x3.

```
> model3.1<-gam(y~s(x1)+s(x2)+s(x4),data=dat)
> summary(model3.1)
Family: gaussian
Link function: identity

Formula:
y ~ s(x1) + s(x2) + s(x4)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.3402     0.0518   25.87   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
        edf Ref.df       F p-value
s(x1) 2.800  3.484 140.621  <2e-16 ***
s(x2) 5.714  6.845  23.831  <2e-16 ***
s(x4) 1.751  2.170   1.485   0.216
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.867   Deviance explained = 88.1%
GCV = 0.30236  Scale est. = 0.2683    n = 100
```

The new model (3.1) has slightly lower GCV than its predecessor and still one more term which is potentially removable. (The reader is invited to make a plot of the new model to verify that the effect of x4 is 0). We proceed by removing the fourth variable.

```
> model3.2<-gam(y~s(x1)+s(x2),data=dat)
> summary(model3.2)
Family: gaussian
Link function: identity

Formula:
y ~ s(x1) + s(x2)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.34020    0.05245   25.55   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
Approximate significance of smooth terms:
        edf Ref.df      F p-value
s(x1) 2.783  3.466 138.06  <2e-16 ***
s(x2) 5.771  6.911  24.21  <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.864   Deviance explained = 87.5%
GCV = 0.30419  Scale est. = 0.27513   n = 100
```

The final model (3.2) has all terms with very significant p-values (<< 0.05), however it has a slightly higher GCV. Given, two out of the three selection criteria are met, the removal of X4 may still be justified.

F-ratio test

The two last models can also be compared against each other using an F-ratio test (see sections 2.1.6 and 4.10 in Wood 2006), testing the null hypothesis (Ho) that the simpler between the two models (3.2) is true. The F-ratio test, under Ho yields the following result:

$$F = \frac{(D_0 - D_1)/(p_1 - p_0)}{D_1/(n - p_1)} \sim F_{p_1 - p_o, n - p_1}$$

where D_0 and D_1 are the deviance of the simpler and more complex model, respectively, p_1 and p_0 are the number of parameters (effective degrees of freedom) for the most complex and simpler model, respectively, and n is the sample size. Gven a model, the deviance (D) is defined as the difference between the log-likelihood (l) of a saturated model (with as many coefficient as there are observations) and the model in question:

$D = 2(l(\beta max) - l(\beta))$

Note that for a model with normally distributed errors the deviance is equal to the residual sum of squares.

It is important to recognize the limitation of such test when applied to GAM. First, it is necessary that the two models being compared are nested within each other: the more complex model must contain the simpler one. Second, and most importantly, the F-ratio test relies on the assumptions that the models degrees of freedom (number of parameters) are known a-priori. We know that in the case of penalized regression splines this is not true, since the effective degrees of freedom are estimated through λ using the GCV. If we had a-priori fixed the number of knots using the `k=...` and `fx=T` within the smooth term (`s`), then the F-ratio approximation would hold true. Thus when using such test one must realize that it is not a clear cut situation, and thus not solely rely

on its use. Usually, the problem arises when the p-value of the F-test is near some pre-determined significance level (e.g., 0.05).

Here is the F-ratio applied to models 3.1 (complex) and 3.2 (simple):

```
#Anove table: F ratio test (Ho: simpler of two GAM model, 3.2, is
#correct)
> anova(model3.2,model3.1,test='F')
Analysis of Deviance Table

Model 1: y ~ s(x1) + s(x2)
Model 2: y ~ s(x1) + s(x2) + s(x4)
  Resid. Df Resid. Dev     Df Deviance      F Pr(>F)
1    88.623     24.884
2    86.500     23.808 2.1225   1.0766 1.8905 0.1547
```

The test suggests that the null hypothesis cannot be rejected (p=0.1547), and the simpler of the two models (3.2) is therefore to be retained. Note that as mentioned above in the F-ratio test the simpler model has to be nested within the more comlex one. In our case this is true, as model 3.1 contains model 3.2.

If one is interested in manually deriving the F-ratio from the two models, the estimated total degress of freedom for each model needs to be known. These can readily be obtained as follows:

```
> #Manually deriving the F-ratio
> #First get the total estimated DF
> model3.2
Family: gaussian
Link function: identity
Formula:
y ~ s(x1) + s(x2)
Estimated degrees of freedom:
 2.783311 5.770936   total =  9.554247
GCV score:  0.3041916
> model3.1
Family: gaussian
Link function: identity
Formula:
y ~ s(x1) + s(x2) + s(x4)
Estimated degrees of freedom:
2.799695 5.71402 1.750989   total =  11.26470
GCV score:  0.3023591
#Note that the Deviance of a model under Normally distributed error is
the sum of residuals squares:
sum(model3.2$residuals^2)#24.88419
sum(model3.1$residuals^2)#23.80762
```

All the ingridients to compute the F-ratio are now available:

```
> ((24.8842-23.8076)/(11.26470-9.554247))/((23.8076)/(100-11.26470))
[1] 2.345979
```

A real life example: Barents Sea cod survival (data `[BScod]` provided)
The objective of this analysis is to study the factors affecting the abundance (and thus survival) of age-1 cod in the Barents Sea. We use the BScod data frame composed by the following variables:

- year: the data set goes from 1980 to 2002 (n=23).
- `age1`: natural logarithm of age1 cod abundance at the year *t*, estimated during the winter trawl survey. This term is the response variable.
- `age0`: natural logarithm of age-0 abundance, at the year *t-1*, estimated during the fall juvenile survey. This term captures the density-dependent effect between the abundance of age-0 and the ensuing age-1 class
- `age4.6`: natural logarithm of age-4 to age-6 cod abundance, at the year *t-1*, estimated from catch-at-age data. This term captures the cannibalistic effect of subadult cod (age 4 to age 6) on age-0 cod. In the Barents Sea only subadult cod prey on younger stages due to greater overlap between the two groups. During winter, adult cod (> age-6) migrate out of the Barents Sea, to the spawning areas of the Norwegian Sea (Lofoten Islands)
- `capelin`: natural logarithm of capelin abundance at the year *t-1*, estimated from the hydroacustic survey. Capelin are supposed to affect juvenile cod mortality either directly (via competition for food) or indirectly (subadult cod may switch their diet from energy-poor juvenile cod to energy-rich capelin when the latter are more abundant)
- `temp`: average summer water temperature during the year *t-1*
- `trawl.type`: categorical variable distinguishing between the two trawl types used in the age-1 cod survey. This term is an indicator variable (1,2) and captures a potential difference of catchability between the two trawl types. Indicator variables will be dealt with in details in Chapter 8.

We start the analysis from a model containing all the covariates described above. This is the most complex model, against which all other nested formulations will be compared. As there are not many observations in the data set (n=23) we force the smooth terms to have d.f ≤4. This is done by setting the `k` argument (i.e., dimension of the basis functions) to 5. The d.f. = k-1.

```
> BScod<-read.table('/Users/lciannel/Documents/MyDocuments/GAM
class/Flodevigen2/data/Barents Sea cod/cod01.csv',header=T,sep=';')
> BScod[1:3,]
  year       age1     age0  capelin temp age4.6 trawl.type
1 1980  1.5260563 4.110874 6.492240  7.4  15.32          1
2 1981 -0.2231436 4.174387 6.599870  6.6  14.85          1
3 1982  5.0297841 4.912655 6.625392  7.1  14.66          1
```

```
> age1.0<-gam(age1~s(age0,k=5)+s(age4.6,k=5)+s(temp,k=5)+
+s(capelin,k=5)+factor(trawl.type),data=BScod)
>
```

Note that there is one parametric component included in the model formulation: `trawl.type`. This term is included as a factor, composed by two levels: 1 and 2.

```
> summary(age1.0)
Family: gaussian
Link function: identity

Formula:
age1 ~ s(age0, k = 5) + s(age4.6, k = 5) + s(temp, k = 5) + s(capelin,
    k = 5) + factor(trawl.type)

Parametric coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)           4.5125     0.3489  12.935 4.22e-09 ***
factor(trawl.type)2   1.9687     0.6666   2.953   0.0106 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
             edf Ref.df     F  p-value
s(age0)    2.962  3.409 9.400 0.000669 ***
s(age4.6)  1.445  1.740 5.684 0.042920 *
s(temp)    1.000  1.000 4.546 0.051167 .
s(capelin) 1.795  2.172 1.943 0.160614
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.865   Deviance explained = 91.5%
GCV = 1.4459  Scale est. = 0.86742   n = 23
>
```

The result of the base model indicates that with the exception of `capelin` and possibly `temp`, all factors have a significant contribution on the survival of age-0 cod. The parametric term of the model, `trawl.type`, also has a significant effect. In particular, it appears that the second gear (trawl.type=2) is more efficient in catching age-1 cod, as the average abundance increased about 45% (1.9687/4.5125*100) when this gear was used. The trawl.type term affects the model intercept and indicates the difference in the average catch between the second and the first gear.

A plot of the model effect (Fig. 2) further illustrates that `capelin` and probably `temp` can be dropped from the model formulation, as their effect is including zero throughout their domain.

```
> par(mfrow=c(2,2))
> plot(age1,se=T,residuals=T,pch=1)
```

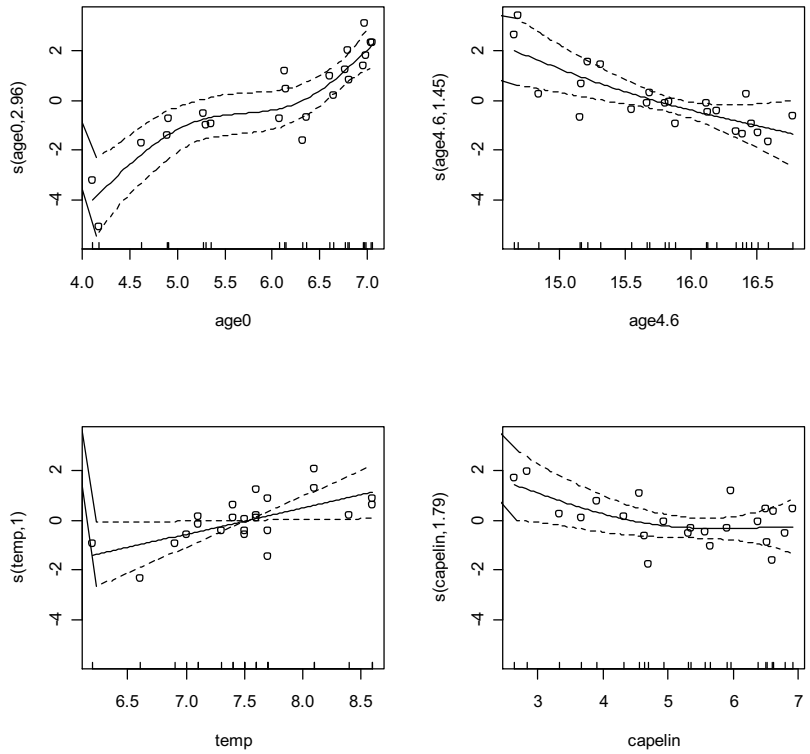


Fig. 2. Additive covariate effects of the `age1.0` model.

The inspection of the covariate effect also gives an assessment of which term has the largest contribution in the predictions of the response variable. For example, among the variable plotted in Fig. 2, the abundance of the previous year age-0 spans the widest range, indicating that this covariate has the largest contribution in the predictions of age-1 abundance. Note that by default all the plots have similar y-axis scale, so they are directly comparable. You can change the default, and have a different scale for each plot, by including the argument: `scale=0` in the `plot.gam` function.

Because of potential correlation between the covariates, we proceed the analysis by removing one term at-a-time, starting with the least significant one, `capelin`.

```
> age1.1<-gam(age1~s(age0,k=5)+s(age4.6,k=5)+s(temp,k=5)+
+factor(trawl.type),data=BScod)

> summary(age1.1)

Family: gaussian
Link function: identity
```

```
Formula:
age1 ~ s(age0, k = 5) + s(age4.6, k = 5) + s(temp, k = 5) +
factor(trawl.type)

Parametric coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)           4.6599     0.3609  12.913 2.73e-10 ***
factor(trawl.type)2   1.6297     0.6510   2.504   0.0226 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
            edf Ref.df      F  p-value
s(age0)   1.000   1.00 35.128 7.27e-06 ***
s(age4.6) 1.768   2.18  2.223    0.123
s(temp)   1.000   1.00  2.368    0.142
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.821   Deviance explained = 85.9%
GCV = 1.5386  Scale est. = 1.1528    n = 23
>
```

The model GCV has increased, but two other criteria of the selection strategy were met (CI including 0 and EDF ~ 1). The F-ratio test confirms the validity of the simpler model:

```
> anova(age1.1,age1.0,test='F')
Analysis of Deviance Table

Model 1: age1 ~ s(age0, k = 5) + s(age4.6, k = 5) + s(temp, k = 5) +
factor(trawl.type)
Model 2: age1 ~ s(age0, k = 5) + s(age4.6, k = 5) + s(temp, k = 5) +
s(capelin,
    k = 5) + factor(trawl.type)
  Resid. Df Resid. Dev     Df Deviance      F Pr(>F)
1     16.82     19.865
2     12.68     11.969 4.1403   7.8959 2.1986 0.1261
```

However, the narrow margin against which rejection would apply brings us in that grey area. Recall that the test is an approximation, as we assume that the model's degrees of freedom are a-priori known.

The removal of capelin resulted in an increased p-value for temperature, suggesting that even this term may be dropped, as also indicated from the inspection of the covariates effect (Fig. 3).

```
> plot(age1.1,se=T,residuals=T,pch=1)
Press return for next page....
Press return for next page....
```

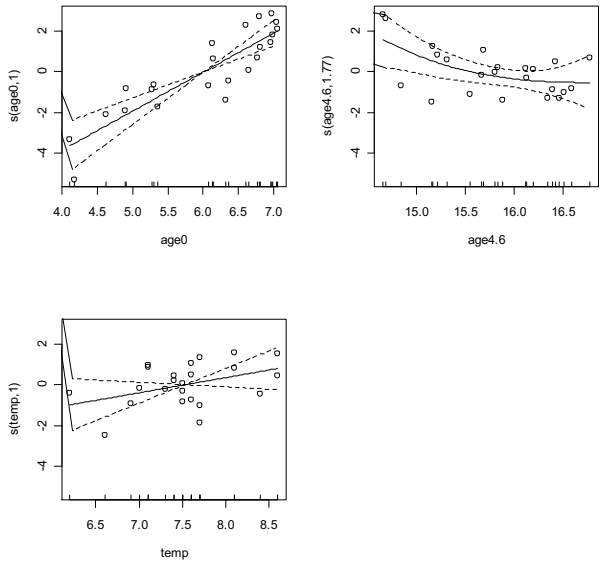


Fig. 3. Covariates effect of the model age1.1. Note that the effect of temp includes zero throughout the range, indicating a nonsignificant contribution.

We proceed by removing temperature from the model formulation and with an F-ratio test:

```
> age1.2<-gam(age1~s(age0,k=5)+s(age4.6,k=5)+factor(trawl.type),
+data=BScod)
> summary(age1.2)

Family: gaussian
Link function: identity

Formula:
age1 ~ s(age0, k = 5) + s(age4.6, k = 5) + factor(trawl.type)

Parametric coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)           4.5690     0.3721   12.28 2.61e-10 ***
factor(trawl.type)2   1.8388     0.6662    2.76   0.0127 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
            edf Ref.df     F  p-value
s(age0)   1.000  1.000 56.37 5.36e-08 ***
s(age4.6) 1.583  1.952  4.27   0.0448 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.805   Deviance explained = 83.7%
GCV =  1.567  Scale est. = 1.2548    n = 23

> anova(age1.2,age1.1,test='F')
Analysis of Deviance Table
```

```
Analysis of Deviance Table

Model 1: age1 ~ s(age0, k = 5) + s(age4.6, k = 5) + factor(trawl.type)
Model 2: age1 ~ s(age0, k = 5) + s(age4.6, k = 5) + s(temp, k = 5) +
factor(trawl.type)
  Resid. Df Resid. Dev     Df Deviance      F Pr(>F)
1    18.048     23.109
2    16.820     19.865 1.2281   3.2442 2.2916 0.1456
```

Now all the terms have p-value < 0.1, but the GCV has slightly increased compared to that of the previous model. The removal of temperature was still justified by the low EDF (1.00), by the approximately flat profile of the term's effect and by the F-ratio test. However it is possible that temperature and capelin still have an effect, perhaps nonadditive, given the narrow margin over which they were dropped from the model formulation. In a later chapter (10, Modelling nonadditivity), we consider again temperature and capelin as nonadditive covariates.

The last model explains 80.5% of the total variance in age-1 abundance. The abundance of age-0 and subadult cod affect the abundance of the ensuing age-1 cod in a linear (`edf=1`) and in a nonlinear (`edf>1`) fashion, respectively. Note that from the summary output it is not possible to understand weather these effects are positive or negative, although it is rather intuitive to guess the outcome. In order to assess the terms effect it is necessary to plot the model results (Fig. 4).

```
> plot(age1.2)
```

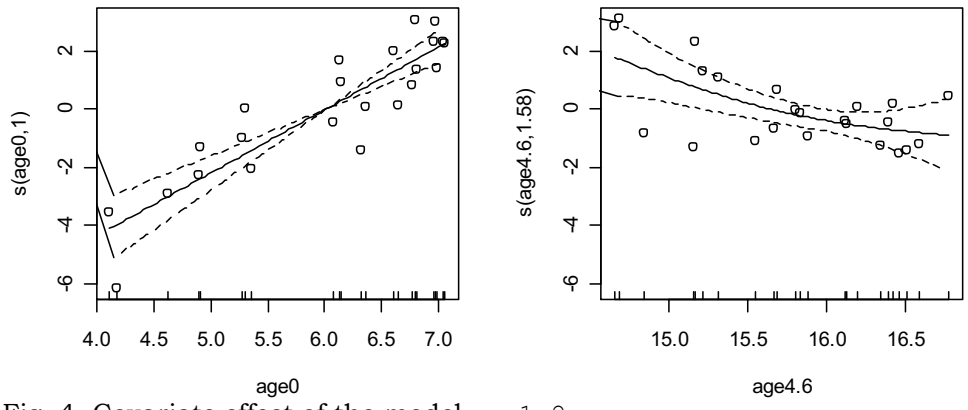


Fig. 4. Covariate effect of the model `age1.2`.

The removal of any other term would result in a further increase of the model GCV, and it is not justified by the covariate effect (EDF>>1 and CI not always including 0). In the next section, on model diagnostic, we examine the residuals of this final formulation.

Chapter 6: Model diagnostic and remedial measure

In the present section we examine different ways to assess the quality of a model fit, i.e., the model diagnostic. As for linear models, an important component of the model fit is given by residual pattern. By inspecting the residuals it is possible to assess if there are outliers that may force the fit toward unreasonable stretches, or if there is any left over pattern, indicating that the model is not adequate, or that some important variables have been omitted from the formulation.

A model that does not have any major problem in diagnostic, may produce results that are unreasonable from a biological and ecological perspective. Thus, the judgment of the analyst is the principal diagnostic measure. Statistical models are used to inspect patterns in the data. If there is no possible 'practical' explanation to a found statistical relationship (where practical does not refer only to our desired expectation), than the validity of such relationship should be questioned.

There are two types of residuals in a GAM model: *full* and *partials*. Given the following additive model, composed by two variables:

$$y_i = \alpha + s_1(x_{1i}) + s_2(x_{2i}) + e_i$$

the *full* model residuals are simply the difference between the observations and the predictions:

$$e_i = y_i - \alpha + s_1(x_{1i}) + s_2(x_{2i})$$

Partial residuals refer to single terms, and they are derived as the difference between the observations and the predictions from a model that does not include the effect of the term under scrutiny. For example, the partial residuals of the first term (x1) are given by:

$$e_{1i} = y_i - \alpha + s(x_{2i}) = s_1(x_{1,i}) + e_i$$

A plot of the partial residual of a term (x) against the value of the same term is useful in recognizing whether there are data points that force the smooth the term toward unreasonable stretches (leverage). Such plots are readily obtained by setting the `residuals=T` within the `plot` function. The following example illustrates the use of partial residuals in the detection of extreme data cases.

Example 1: partial residuals

Start from an additive model with two significant variables, but with an outlier effect:

```
> set.seed(999)
> x1<-runif(100)
> x2<-runif(100)
> e<-0.5*rnorm(100)
> y<-5*x1^3+sin(3*pi*x2)+e+c(rep(0,49),3,rep(0,50))#outlier
#effect
> dat<-data.frame(x1=x1,x2=x2,y=y)
> model1<-gam(y~s(x1)+s(x2),data=dat)
```

Note the artificial outlier effect of the 50th data point, obtained simply by adding a relatively large number (say 3) to the generated responses (y).

```
> model1<-gam(y~s(x1)+s(x2),data=dat)
> summary(model1)

Family: gaussian
Link function: identity

Formula:
y ~ s(x1) + s(x2)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.39876    0.05798   24.12   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
        edf Ref.df     F p-value
s(x1) 2.929  3.649 130.4  <2e-16 ***
s(x2) 5.771  6.910  19.4  <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.857   Deviance explained = 86.9%
GCV = 0.37227  Scale est. = 0.33616   n = 100

> plot(model1,residuals=T,pch=1,pages=1)#Fig. 1
```

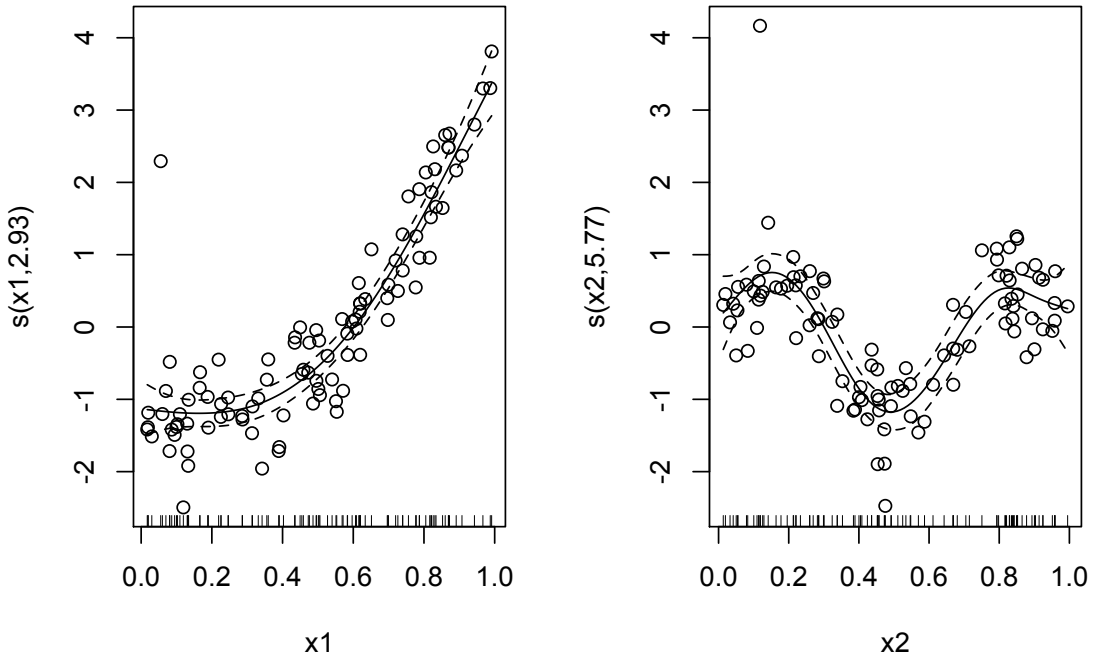


Fig. 1. Covariate effect of model 1. Note the presence of an outlier from the plot of the partial residuals.

The effect of the outlier can be modeled by including in the model an 'outlier' variable:

```
> outlier<-c(rep(0,49),1,rep(0,50))
> model1.1<-gam(y~s(x1)+s(x2)+factor(outlier),data=dat)
> summary(model1.1)

Family: gaussian
Link function: identity

Formula:
y ~ s(x1) + s(x2) + factor(outlier)

Parametric coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)       1.36048    0.04427  30.732  < 2e-16 ***
factor(outlier)1  3.82837    0.46542   8.226 1.54e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
        edf Ref.df      F p-value
s(x1) 3.162  3.942 216.19  <2e-16 ***
s(x2) 6.276  7.412  29.57  <2e-16 ***
---
```

```
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.917   Deviance explained = 92.6%
GCV = 0.21884  Scale est. = 0.19381   n = 100

> plot(model1.1,residuals=T,pch=1,pages=1)# Fig. 2
```

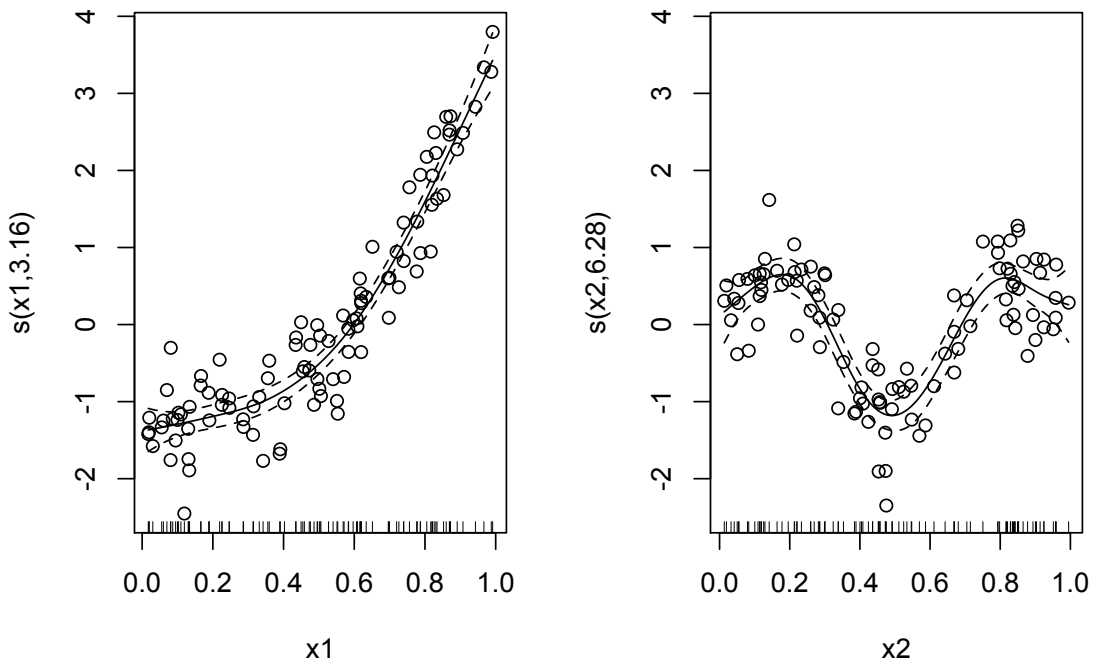


Fig. 2. Covariate effect of model 1.1.

The following code shows how to get the partial residuals, term predictions and confidence interval to draw the same plot shown in Fig. 1.

```
> predX1<-predict.gam(model1,type='terms')[,1]
> predX2<-predict.gam(model1,type='terms')[,2]
> seX1<-predict.gam(model1,type='terms',se.fit=T)[[2]][,1]
> seX2<-predict.gam(model1,type='terms',se.fit=T)[[2]][,2]
> res<-residuals(model1)
>
> resX1<-predX1+res
> resX2<-predX2+res
>
> par(mfrow=c(2,1))
> plot(sort(x1),predX1[order(x1)],xlab='x1',ylab='additive
+effect',type='l',ylim=c(-2,4.2))
> lines(sort(x1),predX1[order(x1)]+1.96*seX1[order(x1)],lty=2)
> lines(sort(x1),predX1[order(x1)]-1.96*seX1[order(x1)],lty=2)
> points(x1,resX1,pch=1)
```

```
> plot(sort(x2),predX2[order(x2)],xlab='x2',ylab='additive
+effect',type='l',ylim=c(-2,4.2))
> lines(sort(x2),predX2[order(x2)]+1.96*seX2[order(x2)],lty=2)
> lines(sort(x2),predX2[order(x2)]-1.96*seX2[order(x2)],lty=2)
> points(x2,resX2,pch=1)
```

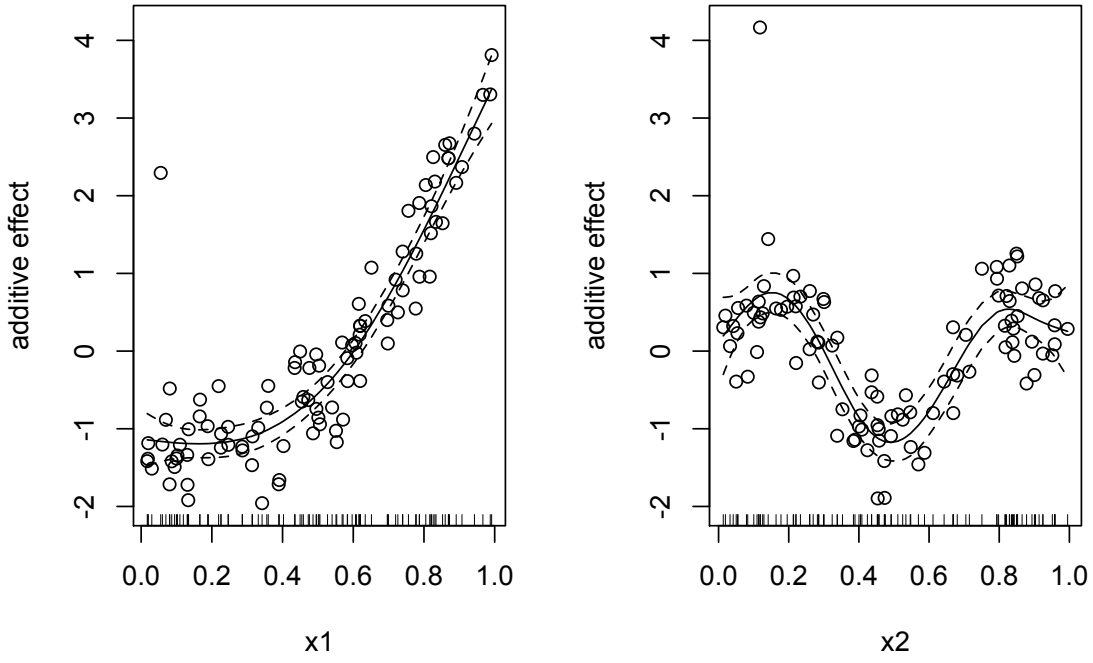


Fig. 3. Same plot of Fig. 1, drawn from the model terms.

Example 2: full residuals

When inspecting the full residuals we are mainly concerned about the following features:

- normality
- constant variance
- autocorrlation
- outliers

Very similar objectives of the residual patterns also apply to linear regression models, as collectively they fulfill the *iid* (independent and identically distributed) assumption of the error terms.

The normality of the residuals can be visually inspected through histograms or normal quantile-quantile plots. The constant variance assumption can be checked by plotting the residuals against the fitted values. Finally, the lack of autocorrelation can be checked with the `acf` function. Some examples, based on existing gam models follow.

An example from the fictitious data in model 1 and 1.1 (Fig. 4).

```
> par(mfcol=c(3,2))
> plot(model1$fitted,residuals(model1),xlab='Fitted
+values',ylab='residuals',main='Model 1')
> qqnorm(residuals(model1))
> qqline(residuals(model1))
> hist(residuals(model1),main='Model 1')
> plot(model1.1$fitted,residuals(model1.1),xlab='Fitted
+values',ylab='residuals',main='Model 1.1')
> qqnorm(residuals(model1.1))
> qqline(residuals(model1.1))
> hist(residuals(model1.1),main='Model 1.1')
```

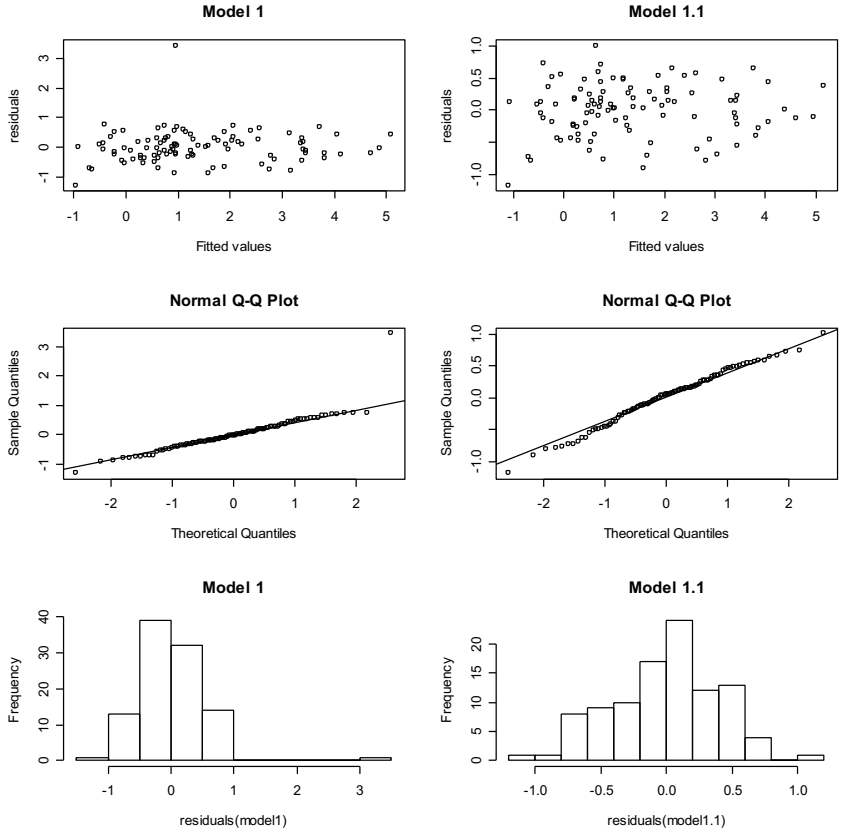


Fig. 4. Residual patterns from model1 (left) and model1.1 (right) of example 1. In each column, the plots shown are: (top) residuals versus fitted values; (center) normal quantile-quantile plot; (bottom) histogram. Note the presence of the strong outlier in model 1.

Example 3: Barents Sea cod model (`age1.2`) (Fig. 5).

```
> par(mfrow=c(1,3),omi=c(2.2,0.2,2.2,0.2))
> plot(age1.2$fitted,residuals(age1.2),xlab='Fitted
+values',ylab='residuals',main='Barents sea cod')
```

```
> qqnorm(residuals(age1.2))
> qqline(residuals(age1.2))
> hist(residuals(age1.2),main='Barents Sea cod')
```

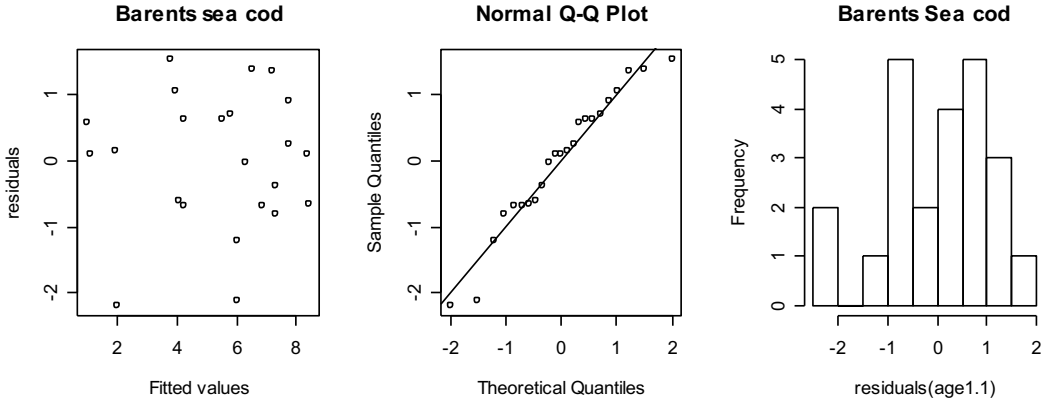


Fig. 5. Residual patterns of Barents sea cod model (`age1.2`).

Because the data used for the Barents Sea cod model are from a time series analysis (collected over time), it is also informative to verify that the residuals do not have any temporal structure, such as autocorrelation (this is done with the function `acf`) (Fig. 6). The presence of autocorrelation in the residuals may be indicative of an inadequate model or of an artificial structure introduced by the analysis. In spatial models it is informative also to check for residuals spatial correlation.

```
> par(mfcol=c(1,2),omi=c(1.8,0.5,1.8,0.5))
> acf(BScod$age1,main='Age-1 cod')
> acf(residuals(age1.2),main='Residuals')
```

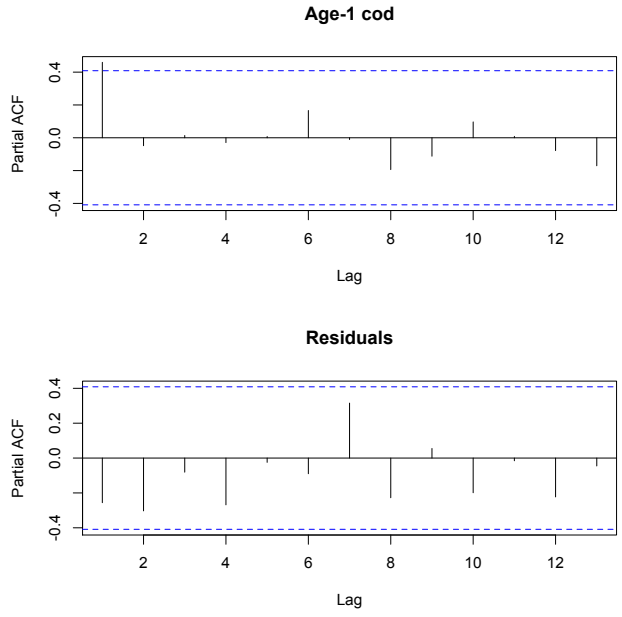


Fig. 6. Temporal autocorrelation in the the Barents Sea age-1 cod time series (left) and in the residuals of the model `age1.2`. The lack of autocorrelation in the residuals is an indication of a good model fit.

In the mgcv library of R there is a function (`gam.check`) that automatically executes some basic model diagnostics, along with information about the model convergence.

Example 4: non-constant variance

The following model has a multiplicative error structure, but it is additive on a log scale.

```
> set.seed(999)
> x1<-runif(100)
> x2<-runif(100)
> e<-0.1*rnorm(100)
> Y <-exp(0.25*x1^3+sin(3*pi*x2)+e)
> model2<-gam(y~s(x1)+s(x2))
> gam.check(model2)#Fig. 7
```

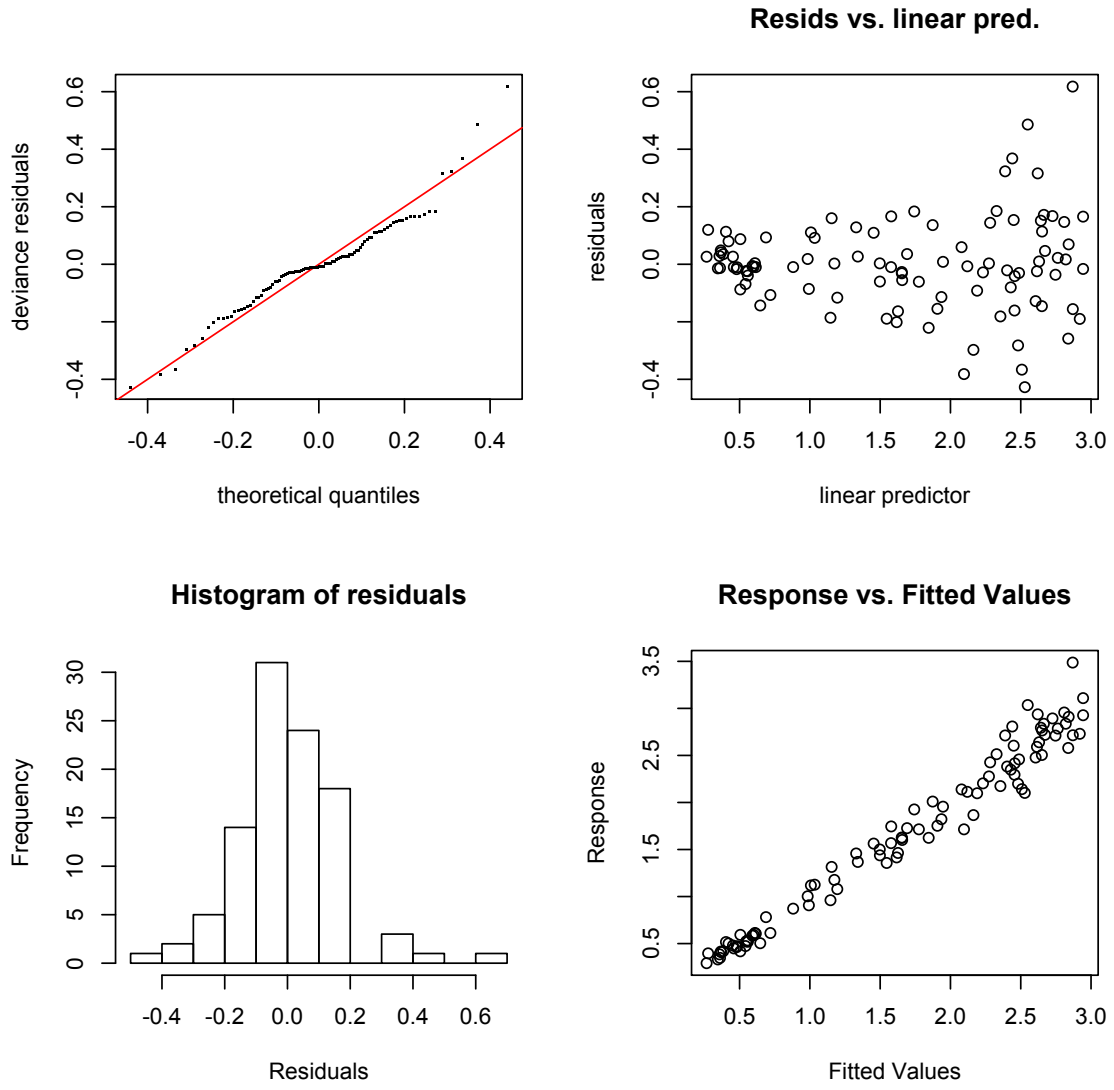


Fig. 7. Diagnostic of `model2`, obtained with the `gam.check` function. Note the increasing variance of the residuals.

Chapter 7: Generalized Additive Models

The topics and theory covered so far only pertains to instances in which the response variable Y is normally distributed (given the covariates being fixed). However, there may be interest in examining data that are not normally distributed, such as presence/absence or counts. In such circumstances the theory behind additive models can be expanded to other distribution families, so that the technique is more generally applicable. Additive models which are not constrained to admit normal errors are referred to Generalized Additive Models (GAM). Note that the additive models explored so far, are a special case of GAM applied to normally distributed errors, just like linear models are a special case of Generalized Linear Models (GLM).

The expansion of regression models to different distribution families has been well studied in linear models and the classical reference is McCullagh and Nelder (1989)[6]. Consider the following linear model:

$$y_i = \sum_{j=1}^{m} \beta_j x_{j,i} + \varepsilon_i = \hat{y}_i + \varepsilon_i \qquad \text{where } \varepsilon \text{ is } N(0,\sigma^2)$$

The model has a *fixed* (the linear predictor $\sum_{j=1}^{m} \beta_j x_{j,i}$) and a *random* ($\varepsilon$) component. In words, the observed data (y) are equal to the linear predictor (y-hat) plus a random component (ε).

The linear predictor is not always a useful representation of the observed data. Consider, as an extreme example, observations from a binomial distribution. Each value represents the probability of a certain event given a certain number of trials $y_i = s_i / a_i$, where s_i is the success and a_i is the number of trials. Thus the response value can only span from 0 to 1. However, there is nothing that constrains the linear predictor to span from 0 to 1. So there is a lack of connection between the linear predictor and the observations. Analogously, Poisson distributed data can only take positive values, but nothing constrains the linear predictor from being negative.

One possible solution to this problem is to transform the response variable. So the goal is to apply a function t to the data such that $t(y) \sim N(\hat{y}, \sigma^2)$. For binomial-like data a possible t is the arcsine-square root transformation. Data transformation affects both the fixed and the random portion of the linear model. This may be excessive, because the

[6] I also suggest reading the paper by Venables and Dichmont (2004); a very good summary of LM, GLM and GAM.

problem identified here was only in the fixed portion. There is a better solution to this problem.

A second (more valid) approach is to use a *link* function (ℓ) that connects the parameter of the distribution of the observations with the predictions:

$$\ell^{-1}(linearpredictor) = E(y) = \mu$$
$$linearpredictor = \ell(\mu)$$

The inverse of the link ℓ^{-1} connects the predictions with the observations. Note that the link function only affects the fixed portion of the liner model, while no changes are applied to the error.

In addition, the variance of the data *y*, is a function of the mean (*v(y)*) and of a scale parameter(Φ):

$$\text{var}(y) = \phi v(y).$$

The application of this second methodology requires the knowledge of the link (and the inverse of the link), the variance function and the scale parameter. For data belonging to the exponential distribution families (i.e., normal, Poisson, binomial, gamma, etc.) these requirements have been analytically found. For example,

- in the normal case the *natural link* (it is named *natural* because it is analytically derived from the distribution function) is the *identity*, the variance function is a constant typically equal to 1 (constant variance assumption), and the scale is the variance of the residuals (σ^2).
- In the binomial case the link is the *logit*: $\log(y/(1-y))$ and the inverse of link is: $e^y/(1+e^y)$, the scale parameter is 1 and the variance function is: $y(1-y)$.
- For Poisson data the link is the *log*, the scale is 1 and the variance function is the mean (variance equal to the mean).

Difference between link functions and data transformations

It is important to recognize that there is a difference between link functions and transformation of variables. In fact, they are used for two different purposes. The link is applied when observations are not on the same scale of the predictions. Transformations are typically used to fix the homogeneity of variance problem. In other words, links are used to correct problems in the fixed part of the regression equation, while transformations take care of the random part. Data transformations can also address problems in the fixed part of the regression (e.g., arcsine-

square root transformation for binomial data) however, this should be viewed more as a byproduct of their primary purpose.

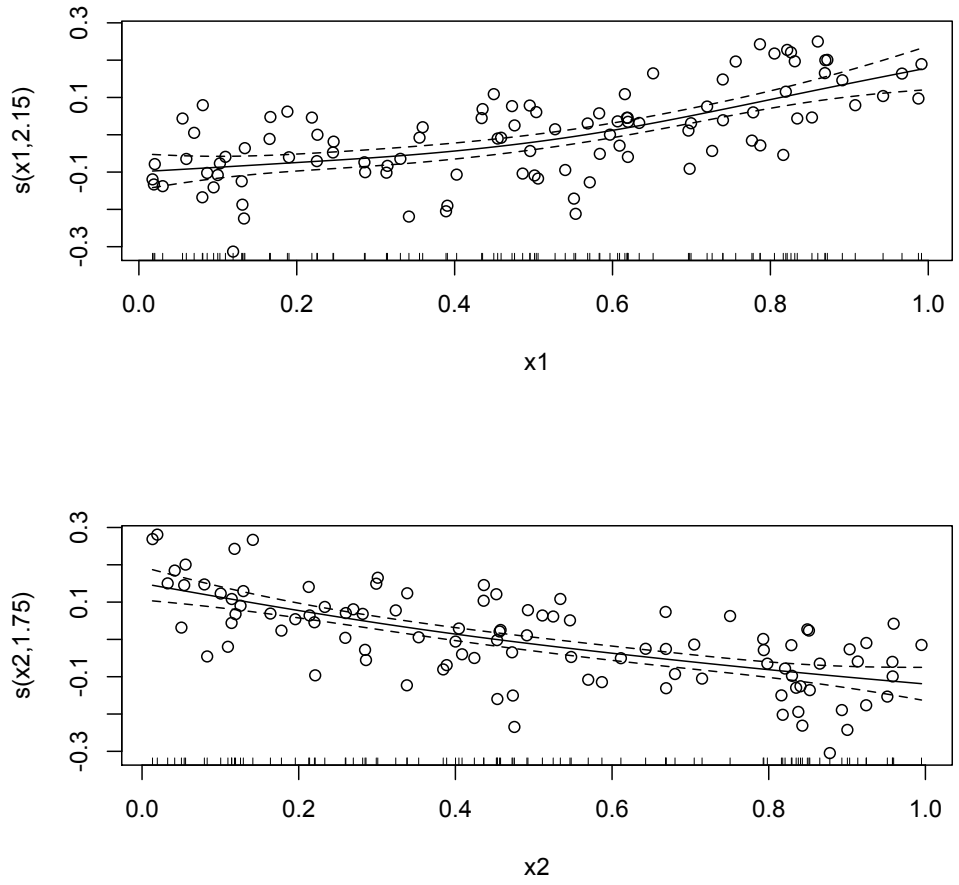
We illustrate the use of GAM on a simulated data set with binary responses (1,0) and on a real data set on larval pollock growth. In addition, in the section on thin plate regression splines (Chapter 12), we use again the binomial family to model the presence/absence of pollock eggs in various regions of the Gulf of Alaska.

Example 1: simulated binary response
We start from a regular model, similar to those used above. Here the values of Y are not yet in a binary form:

```
> set.seed(999)
> x1<-runif(100)
> x2<-runif(100)
> y<-e<-0.1*rnorm(100)
> y<-0.25*x1^3-sqrt(0.1*x2)+e
> dat<-data.frame(x1=x1,x2=x2,y=y)
> model1<-gam(y~s(x1)+s(x2),data=dat)
> summary(model1)

Family: gaussian
Link function: identity

Formula:
y ~ s(x1) + s(x2)

Parametric coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.157776   0.009006  -17.52   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 '
' 1

Approximate significance of smooth terms:
       edf Est.rank     F  p-value
s(x1) 2.149        5 15.65 3.22e-11 ***
s(x2) 1.747        4 19.87 6.45e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 '
' 1

R-sq.(adj) =  0.592   Deviance explained = 60.8%
GCV score = 0.0085292   Scale est. = 0.0081116  n = 100
```

```
> plot(model1,pages=1,residuals=T,pch=1)
```

Fig. 1. Covariate effect of model 1.



Note that the X1 has a positive effect and the X2 has a negative effect on Y. We create a binary response variable (0,1) by setting = 0 all Y value <= than the median of Y and = 1 all Y values > than the median. Then we model the new response variable using a binomial family distribution:

```
> ypres<-1*(y>=median(y))
> dat<-data.frame(x1=x1,x2=x2,ypres=ypres)
> model1.1<-gam(ypres~s(x1)+s(x2),family=binomial,data=dat)
> summary(model1.1)

Family: binomial
Link function: logit

Formula:
ypres ~ s(x1) + s(x2)

Parametric coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   0.1012     0.2796   0.362    0.717

Approximate significance of smooth terms:
```

```
        edf Est.rank Chi.sq  p-value
s(x1) 2.976        6  19.28  0.00372 **
s(x2) 1.000        1  21.96 2.79e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.459   Deviance explained = 41.6%
UBRE score = -0.090567  Scale est. = 1         n = 100>
```

Note the scale parameter is equal to 1 in the binomial case. Also, the UBRE (rather than the GCV) is criterion by default applied to estimate the smoothing parameters when the distribution of the response is binomial.

```
> plot(model1.1,pages=1,residuals=T,pch=1)# (Fig. 2)
```

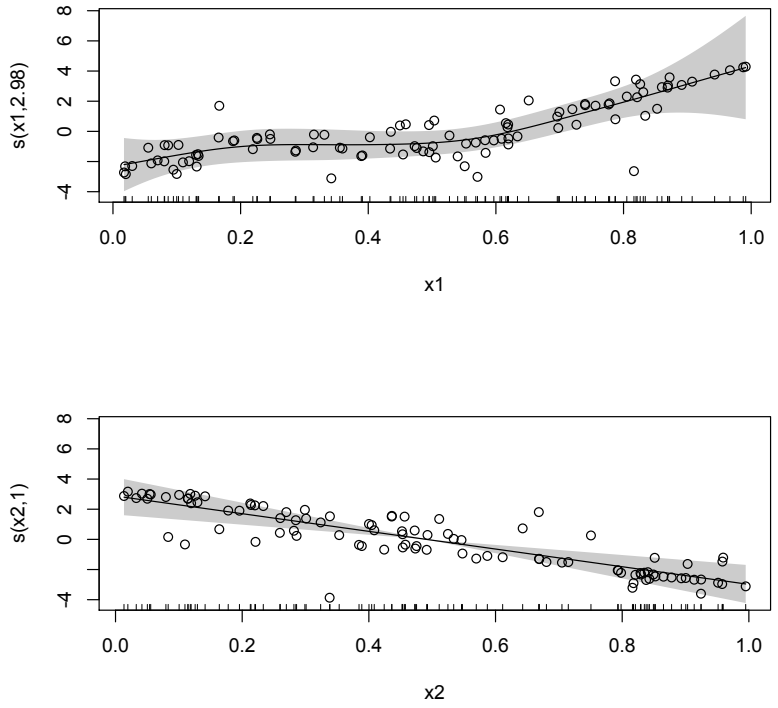


Fig. 2. Covariates effect of `model1.1`.

Expectedly, the covariate effect is similar to that seen for the model 1, when the Y was not binomial. The model predictions are on the logit scale, unless we specify the response scale (this is done by setting `type='response'` in the `predict.gam` function):

```
> round(range(predict.gam(model1.1)),3)
[1] -4.247  5.808 #logit scale
> round(range(predict.gam(model1.1,type='response')),3)
[1] 0.014 0.997 #response scale
```

Another way to get the predictions in the scale of the response is to apply the inverse link of the binomial family:

```
>
round(range(exp(predict.gam(model1.1))/(1+exp(predict.gam(model1.1)))),
3)
[1] 0.014 0.997
```

Example 2: larval pollock growth (data `[larvae]` provided)
The purpose of this study was to develop a physiological index to determine the condition of walleye pollock larvae in the field (Theilacker and Shen 2001). The response was measured as a binomial process: healthy = fast growth, unhealthy = slow growth. The study was conducted in the laboratory, and different feeding treatments were used to produce fast and slow growing pollock larvae. The covariates included in the following analysis to predict the larval condition (fast or slow growing) are fraction of muscle cells dividing (`celldiv`) and larval length (`sl`, in mm). Muscle cells from laboratory reared larvae were dissected, stained, and then run through a flow cytometer to determine the fraction of cell dividing.

```
> larvae.gam<-
gam(fastslow~s(sl,k=4)+s(celldiv,k=4),family=binomial,data=larvae)
> summary(larvae.gam)

Family: binomial
Link function: logit

Formula:
fastslow ~ s(sl, k = 4) + s(celldiv, k = 4)

Parametric coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -0.3967     0.6800  -0.583     0.56

Approximate significance of smooth terms:
             edf Ref.df Chi.sq p-value
s(sl)      2.853  2.982 10.643  0.0137 *
s(celldiv) 1.000  1.000  3.729  0.0535 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.673   Deviance explained = 62.9%
UBRE score = -0.31778  Scale est. = 1         n = 57
```

The results indicate that both terms have a marginally significant effect on the growth condition of pollock larvae (the reader is invited to eliminate the least significant variable to see whether the model improves).

```
> plot(larvae.gam,residuals=T,pch=1,pages=1) #Fig. 3
```

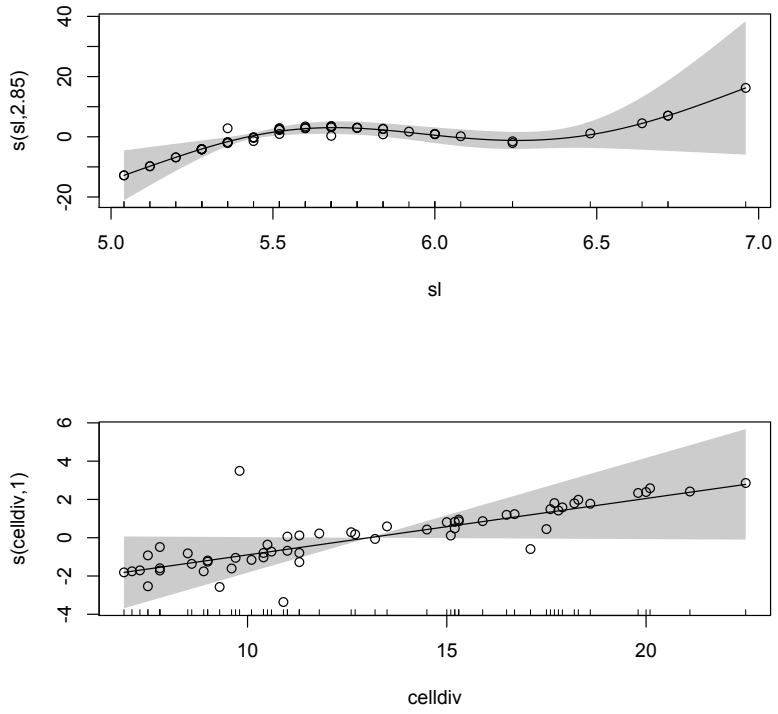


Fig. 3. Covariate effect of the `larvae.gam` model.

Both covariates have a small positive effect. However, the effect of `sl` is nonlinear and spawn a larger interval of anomalis in the link scale. In particular, larval condition was strongly correlated with sl only at highest and lowest extremes of sl, while it was uncorrelated (0 effect) in the middle range (5.7-6.3 mm). Typically, pollock larvae grow fast right after hatching, when their nourishment depends on the energy-rich reserves present in the egg (yolk-sac phase). In contrast, first feeding larvae have a slower growth rate, due to switching from an endogenous energy source (yolk) to exogenous source (prey). Several days after the first feeding stage, surviving larvae show high growth again as they are the more efficient predators left in the tank. The shape of the sl effect reflects the growth pattern of pollock larvae.

Chapter 8: Parallel functions

The following three chapters explore more elaborated GAM formulations used to model variable covariate effects, due to changes of the intercept term (i.e., parallel functions, this chapter), an interaction between two or more covariates (i.e., nonadditivity, chapter 9), or spatially variables intercepts and slope term (i.e., variable coefficient GAM, chapter 10). Collectively, these expanded formulations are said to have *variable coefficients*, as they refer to instances in which either the intercept or one of the smooth functions can change in relation to other covariates. In order to understand the syntax of variable coefficient GAM in R, it is vital to understand the use of factor and indicator variables and the use of the `'by'` argument within the smooth function. We start with a relatively simple case of parallel functions (i.e., only the intercept is changing).

The case of parallel functions in GAM is analogous to parallel lines in a linear model. Consider the following statistical model:

$$y_i = \alpha + g(x_{1,i}) + x_{2,i} + e_i$$

where X_2 is a factor variable with, say two levels (e.g., a and b). Another way to write the same model is with *indicator* variables (u):

$$y_i = g(x_{1,i}) + \alpha_1 u_{1,i} + \alpha_2 u_{2,i} + e_i$$

The covariates u_1 and u_2 are indicators composed by 0-1 values:

- $u_1 = 1$ if X_2 = a, 0 otherwise
- $u_2 = 1$ if X_2 = b, 0 otherwise

The αs are the covariates coefficients. Since the values of *u* can only be 0 or 1, the αs are also taking only two values (α_1 or α_2), each representing the overall average of response variable over the two levels of X_2. Thus, the difference of the αs represents the difference in height of $g(X_1)$ between the two regimes defined by X_2. An example should help to clarify the concept of parallel functions.

Example 1: parallel functions

We simulate data from a `sine` function with variable intercept terms. In particular, the intercept equal to 3 when the second covariate X2=0, otherwise the intercept is equal to 5.

```
> set.seed(999)
> x1<-runif(200)
> x2<-factor(c(rep(0,100),rep(1,100)))#second covariate
> y1<-3+sin(2*pi*x1[1:100])+0.3*rnorm(100)#intercept=3
> y2<-5+sin(2*pi*x1[101:200])+0.3*rnorm(100) #intercept=5
> y<-c(y1,y2)
```

```
> dat<-data.frame(x1=x1,x2=x2,y=y)
> model1<-gam(y~s(x1)+x2,data=dat)
> summary(model1)

Family: gaussian
Link function: identity

Formula:
y ~ s(x1) + x2

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.97938    0.02971  100.27   <2e-16 ***
x21          1.98712    0.04234   46.94   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘
’ 1

Approximate significance of smooth terms:
       edf Ref.df     F p-value
s(x1) 5.763  6.918 141.9  <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘
’ 1

R-sq.(adj) =  0.946   Deviance explained = 94.8%
GCV = 0.090477  Scale est. = 0.086965  n = 200
```

Note that the difference in the intercept between the two levels of the factor variable X_2 (i.e., 1.98) is approximately equal to the difference in the height (i.e., 2) of the value of Y between the two domains defined by X_2.

```
> plot(x1,y,type='n',main='Parallel functions')
> points(x1[1:100],y1,col='red',pch=16)
> points(x1[101:200],y2,col='blue',pch=16)
> lines(sort(x1[1:100]),model1$fitted[1:100]
+[order(x1[1:100])],col='red')
> lines(sort(x1[101:200]),model1$fitted[101:200]
+[order(x1[101:200])],col='blue')
```

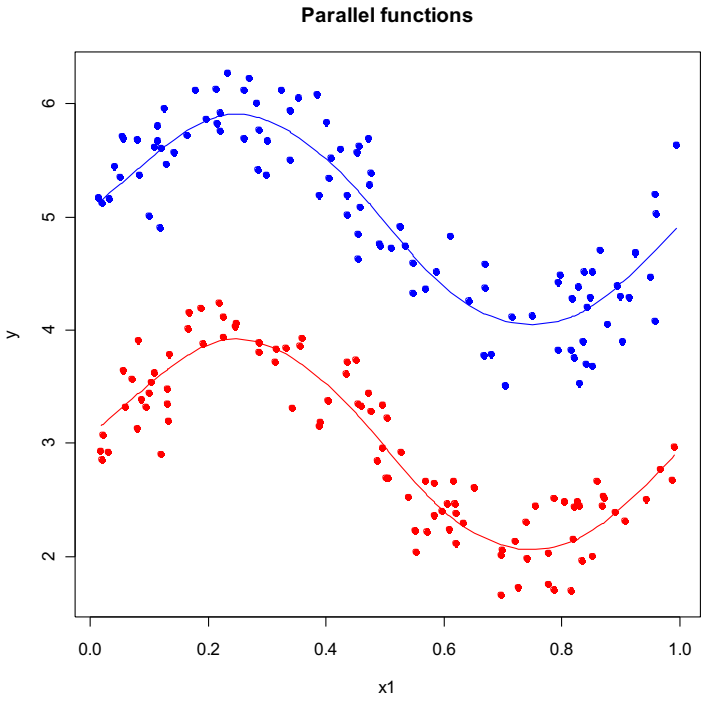


Fig.1. Plot of the data used in example 1. The data are generating parallel functions. The lines are the GAM predictions from the `model1`.

A similar result could be obtained by specifying a model without the overall intercept term (α). This is done by adding `-1` to the previous model formulation.

```
> model2<-gam(y~s(x1)+x2-1,data=dat)#-1 stands for 'no
#intercept'
> summary(model2)

Family: gaussian
Link function: identity

Formula:
y ~ s(x1) + x2 - 1

Parametric coefficients:
    Estimate Std. Error t value Pr(>|t|)
x20  2.97938    0.02971   100.3   <2e-16 ***
x21  4.96650    0.02971   167.1   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 '
' 1
```

```
Approximate significance of smooth terms:
        edf Ref.df     F p-value
s(x1) 5.763  6.918 141.9  <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘
’ 1

R-sq.(adj) =  0.946   Deviance explained = 99.5%
GCV = 0.090477  Scale est. = 0.086965  n = 200
```

Note that now the estimate of the parametric terms amounts to two constants ($X_{[2,0]}$=2.97 and $X_{[2,1]}$=4.96), approximately equal to the value of the intercept term used in the original model (i.e., 3.00 and 5.00).

Another way to specify the same model and get similar results is by creating two different indicator variables (u_1 and u_2):

```
> u1<-1*(x2==0)
> u2<-1*(x2==1)
> model1.1<-gam(y~s(x1)+u1+u2-1,data=dat)
> summary(model1.1)

Family: gaussian
Link function: identity

Formula:
y ~ s(x1) + u1 + u2 - 1

Parametric coefficients:
   Estimate Std. Error t value Pr(>|t|)
u1  2.97938    0.02971   100.3   <2e-16 ***
u2  4.96650    0.02971   167.1   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘
’ 1

Approximate significance of smooth terms:
        edf Ref.df     F p-value
s(x1) 5.763  6.918 141.9  <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘
’ 1

R-sq.(adj) =  0.946   Deviance explained = 99.5%
GCV = 0.090477  Scale est. = 0.086965  n = 200
```

Chapter 9: Nonadditivity

As implied by the name, GAM bear the implicit assumption of additivity. A model is additive when the effect of one covariate is independent from the value of another covariate. The ecological world however, is not always additive. Consider, for example, the foraging impact of a predator on a prey. Under additive conditions the effect of predator density on the foraging impact should be independent of other covariates, such as water temperature. In reality the foraging impact depends also on the spatial overlap between the predator and the prey, which in turn is affected by the thermal conditions of the water column. Thus, it is likely that at the same density, the predator foraging impact on a prey may change in relation to water temperature. In such circumstances a statistician would say that there is a nonadditive effect of predator and temperature on foraging impact, as the effect of the two covariates on the response is interdependent.

Once nonadditivity is detected, either through a test or simply suspected from biological insight, it is necessary to somehow model it. In the following chapter we describe 2 different approaches to model nonadditivity: fixed threshold and univariate threshold. In the last chapter we cover thin plate regression splines and tensor products to model continuous interactions. Most of the functions used in this chapter on threshold models have been developed by Dr. Kung-Sik Chan (University of Iowa), and are available in his `tgam` library.

Fixed threshold

Consider the particular case of nonadditivity between a continuous (X_1) and a factor (X_2) variable. In contrast to the case of parallel function, here we assume that there is a significant interaction between the covariates X_1 and X_2, and therefore the smooth function of X_1 will change its shape over the contrasting levels of X_2 (variable functions). If there are only two levels in the factor variable (X_2), the statistical model is:

$$y_i = \alpha_1 u_{1,i} + \alpha_2 u_{2,i} + u_{1,i} g_1(x_{1,i}) + u_{2,i} g_2(x_{1,i}) + e_i \qquad (1)$$

A similar model formulation is:

$$Y_t = \begin{cases} \alpha_1 + g_1(x_{1,t}) + \varepsilon_t & \text{if } X_2 = \text{level 1} \\ \alpha_2 + g_2(x_{1,t}) + \varepsilon_t & \text{if } X_2 = \text{level 2} \end{cases}$$

Essentially, the above model implies that the intercept (α) and the smooth function (g) of the covariate X_1, can change in relation to the

covariate X_2. It is possible to expand the model to include the effect other non-interacting covariates (i.e, only additive effect). The reader is invited to write an explicit formulation of such model.

In R, the variable function scenario is modeled using the '`by`' operator within the '`s`' function. A possible formulation, in the simple case in which only the covariates X_1 and X_2 are considered, is the following:

```
> model1<-gam(y~u1+u2+s(x1,by=u1)+s(x1,by=u2)-1,data=dat)
```

corresponding to the model in Eq. 1.

Thus, the `by` operator implies that the smooth function g is multiplied by the covariate u_t. As in the case of parallel functions, the covariate u is conveniently chosen to be 0 or 1, so that only one of the two smooth functions (g_1 or g_2) applies within each regime:

- u_1 = 1 if X_2 = level 1, 0 otherwise
- u_2 = 1 if X_2 = level 2, 0 otherwise.

The case of variable functions and the use of the `by` operator is clarified with an example.

Example 1: variable functions

```
> set.seed(999)
> x1<-runif(200)
> x2<-c(rep(0,100),rep(1,100))
> y1<-3+sin(2*pi*x1[1:100])+0.3*rnorm(100)
> y2<-5+cos(2*pi*x1[101:200])+0.3*rnorm(100)
> y<-c(y1,y2)
> dat<-data.frame(x1=x1,x2=x2,y=y)
```

The simulated model corresponds to:

$$y_t = 3u_{1,t} + 5u_{2,t} + \sin(2\pi x_{1,t})u_{1,t} + \cos(2\pi x_{1,t})u_{2,t} + \varepsilon$$

where

- u_1 = 1 if X_2 = 0, 0 otherwise
- u_2 = 1 if X_2 = 1, 0 otherwise

```
> model6<-gam(y~factor(x2)+s(x1,by=factor(x2))-1,data=dat)
> summary(model6)

Family: gaussian
```

```
Link function: identity

Formula:
y ~ factor(x2) + s(x1, by = factor(x2)) - 1

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
factor(x2)0  2.98017    0.02968   100.4   <2e-16 ***
factor(x2)1  4.97241    0.02968   167.5   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘
’ 1

Approximate significance of smooth terms:
                     edf Ref.df      F p-value
s(x1):factor(x2)0 5.472  6.614  71.69  <2e-16 ***
s(x1):factor(x2)1 4.412  5.425 115.92  <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘
’ 1

R-sq.(adj) =  0.948   Deviance explained = 99.5%
GCV = 0.091148  Scale est. = 0.085732  n = 200
```

The summary output shows two significant intercept terms and smooth functions. The intercept terms (2.98, 4.97) are approximately equal to those used to generate the data (3.00, 5.00). Also, the noise variance (`Scale est. = 0.085732`) is approximately equal to the noise used to generate the data (0.3^2=0.09).

Figure 1 shows the simulated data and the fitted functions. Note that the gam-fitted smooth functions and the partial residuals are correctly shown for each regime separately. In order to do so, we construct the plots from scratch, using the examples shown earlier in the book (Chapter 4). If the regular `plot` function was used, the partial residuals from both regimes would be shown in each of the fitted smooth function, which indeed is not correct (the reader is invited to try using the `plot` function on the `model6`).

```
> #Plot of smooth effect and partial residuals
> predX1<-predict.gam(model6,type='terms')[,2][x2==0]
> predX2<-predict.gam(model6,type='terms')[,3][x2==1]
> seX1<-predict.gam(model6,type='terms',se.fit=T)[[2]][,2][x2==0]
> seX2<-predict.gam(model6,type='terms',se.fit=T)[[2]][,3][x2==1]
> res<-residuals(model6)
> resX1<-predX1+res[x2==0]
> resX2<-predX2+res[x2==1]
```

```
>
> quartz()
> par(mfrow=c(3,1))
>
plot(sort(x1[x2==0]),predX1[order(x1[x2==0])],xlab='x1',ylab='additive
effect',type='l',ylim=c(-1.5,1.5),main='First regime')
>
lines(sort(x1[x2==0]),predX1[order(x1[x2==0])]+1.96*seX1[order(x1[x2==0
])],lty=2)
> lines(sort(x1[x2==0]),predX1[order(x1[x2==0])]-
1.96*seX1[order(x1[x2==0])],lty=2)
> points(x1[x2==0],resX1,pch=1)
> rug(x1[x2==0])
>
plot(sort(x1[x2==1]),predX2[order(x1[x2==1])],xlab='x1',ylab='additive
effect',type='l',ylim=c(-1.5,1.5),main='Second regime')
>
lines(sort(x1[x2==1]),predX2[order(x1[x2==1])]+1.96*seX2[order(x1[x2==1
])],lty=2)
> lines(sort(x1[x2==1]),predX2[order(x1[x2==1])]-
1.96*seX2[order(x1[x2==1])],lty=2)
> points(x1[x2==1],resX2,pch=1)
> rug(x1[x2==1])
>
> #Plot model effect on the scale of the predictions
> plot(x1,y,type='n',main='Variable functions')
> points(x1[1:100],y1,col='red',pch=16)
> points(x1[101:200],y2,col='blue',pch=16)
>
lines(sort(x1[1:100]),model6$fitted[1:100][order(x1[1:100])],col='red')
>
lines(sort(x1[101:200]),model6$fitted[101:200][order(x1[101:200])],col=
'blue')
>
```

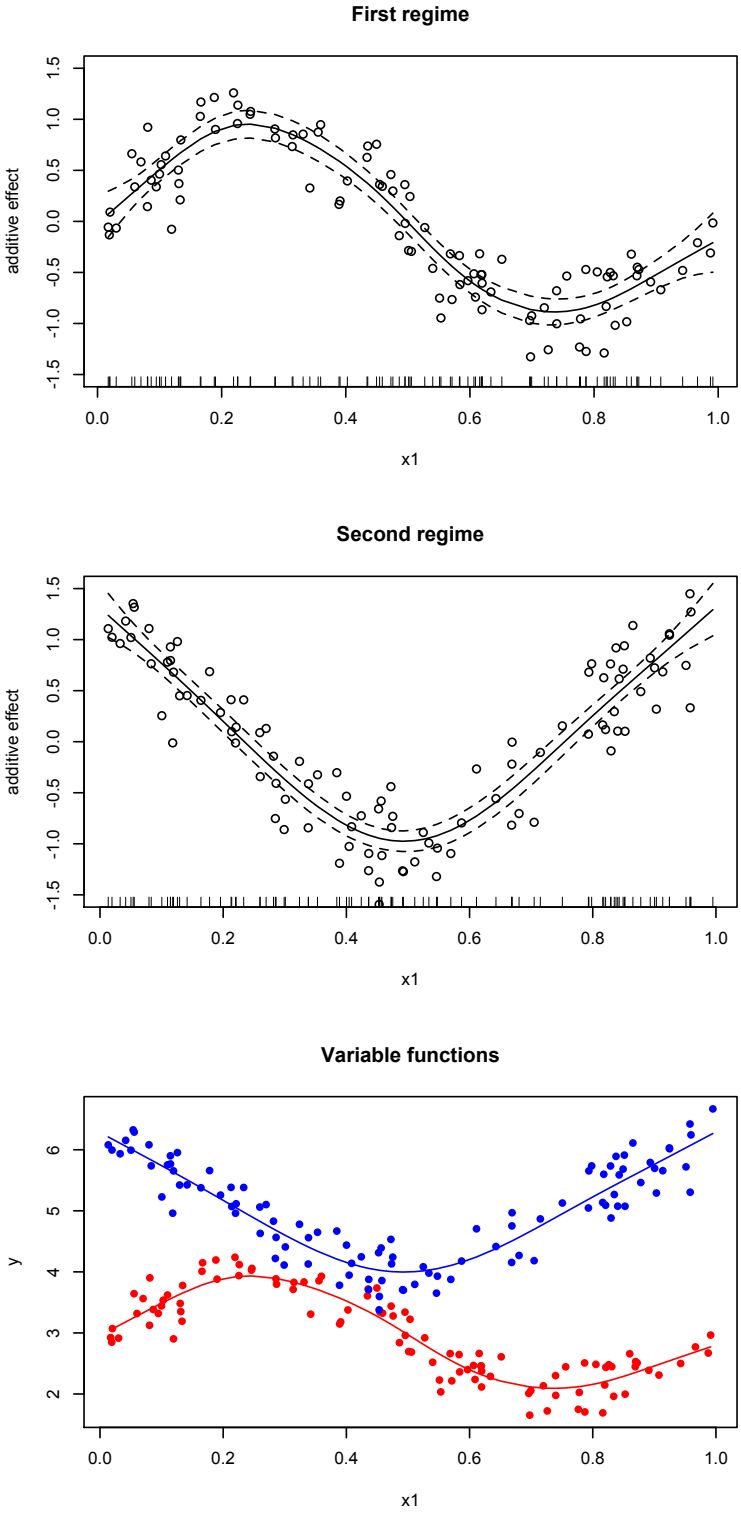


Fig. 1. The bottom plot shows the generated data from example 1. The upper two plots show the estimated functions from the gam model 6.0.

Alternative model formulation with functions not centered around 0
In the previous model formulation (model 6) we included the two regimes as factors of the two levels in the variable x2. By default, the mgcv package of R assumes that the functions are cenetered around 0 whenever the categorical variable is included as a factor. As a result of the centering around 0, we are also forced to include an interecept term for each regime. However, if we include the two regimes identified by the two levels of the variable X2 as numerical indicator variables (*u*, composed of 0 and 1), then the two functions are no longer assumed to be centered around 0, and they will be scaled to the correct magnitude. This approach of modelling threshold fucntions also relaxes the need to include an intercept term for each regime. The example below shows the syntax for this alternative formulation, with functions no longer centered around 0.

```
> model6.1<-gam(y~s(x1,by=I(1*(x2==1)))+s(x1,by=I(1*(x2==0)))-
1,data=dat)
> summary(model6.1)

Family: gaussian
Link function: identity

Formula:
y ~ s(x1, by = I(1 * (x2 == 1))) + s(x1, by = I(1 * (x2 == 0))) -
    1

Approximate significance of smooth terms:
                          edf Ref.df    F p-value
s(x1):I(1 * (x2 == 1)) 5.412  6.425 4660  <2e-16 ***
s(x1):I(1 * (x2 == 0)) 6.472  7.614 1380  <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.948   Deviance explained = 99.5%
GCV = 0.091148  Scale est. = 0.085732  n = 200
```

Note that we no longer need to include an intercept term in the model, because each function is now properly scaled to the correct magnitude. Also note tha the summary output o fthe model 6.1 is identical to the summary output of the model 6.0, except pf course for the fact that the former does not have intercept parametric terms. Lastly, below is the figure showing the partial additive effect for each regime. Note that the difference in y-magnitude of each plot. (code for the figure is in the script).

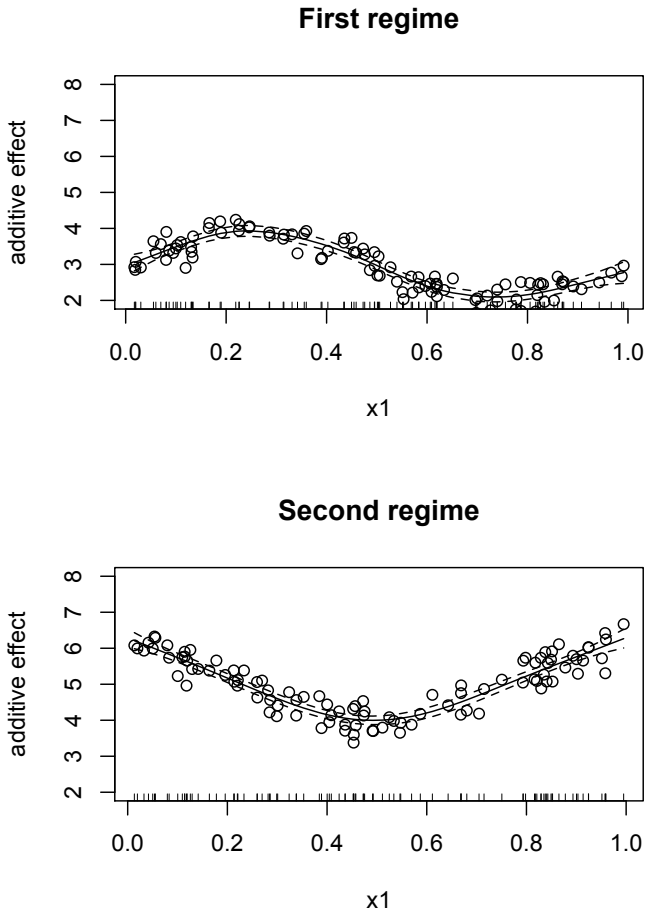


Figure 2. Estimated effects of the two functions included in the model 6.1. Note that the estimated effects are no longer centered around 0, but thety are scaled to their actual magnitude.

Univariate threshold gam

The previous scenario assumes that one of the interacting covariate (X2) is a factor. However, this need not be the case. It is possible that two numerical and continuous covariates interact with each other (e.g., water temperature and predator density on prey survival). In such circumstances one modelling approach is to partition one of the two interacting covariates in two levels, below and above a certain threshold (r), and to treat the ensuing scenario as a fixed threshold problem.

The advantage of the *threshold* approach is that it can be easily modelled as a special case of variable functions described above. The cons are that there is no *apriori* knowledge of where the threshold may be, and that a potentially continuous interaction is modelled as a discontinuous process (i.e., above or below the threshold).

The problem of making a potentially continuous process discontinuous is not ecologically unrealistic, as it relates with the theory of regime shift and phase transitions. The problem of not *apriori* knowing the threshold can also be overcome, by doing a search grid throughout the entire range of the interacting covariate, and selecting the threshold that produces the *best* model, where the *best* model is the one that minimizes the GCV score. This is in fact the approach of the univariate threshold gam described in Chan (MS) and applied in Stenseth et al. 2005. Here we show an example based on simulated data.

Example 2: univariate threshold with common intercept

```
> set.seed(999)
> x1<-runif(200)
> th.var<-1:length(x1)#threshold variable
> y1<-3+sin(2*pi*x1[1:50])+0.3*rnorm(50)
> y2<-3+cos(2*pi*x1[51:200])+0.3*rnorm(150)
> y<-c(y1,y2)
> dat<-data.frame(x1=x1,th.var=th.var,y=y)
> dat<-dat[sample(order(th.var)),]
> rm(th.var,x1,y)
```

The model used to generate the data is similar to the one shown in Example 1, except that here there is no difference in the intercept term (both intercepts are equal to 3). In addition, a new covariate (`th`), defines the two modelled regimes: when `th` ≤ 50 the function is a sine otherwise it is a cosine. The last line of the code makes a random permutation of the data rows so that the original increasing order of the covariate `th` does not influence the search for the threshold.

```
> model1<-
threshold.gam(formula(y~s(x1,by=factor(I(1*(th.var>r))))),a
=0.1,b=0.9,nthd=100,data=dat,threshold.name='th.var')
```

The function `threshold.gam` (read the help file), makes a search grid among the percentile defined by the arguments `a` and `b`. Within this interval it fits `nthd` models and select the one that minimizes the overall GCV. The function is computationally intensive, and therefore it may take some time to execute (3.82 sec in a Mac Book, processor 2.6GHz, 16GB memory). In the above formulation the `'I'` operator protect the argument with parenthesis, so that the product is actually computed, rather than treated as a formula code (typically the sign `*` within a formula means: include additive and interaction effect).

```
> summary(model1$res)

Family: gaussian
Link function: identity

Formula:
y ~ s(x1, by = factor(I(1 * (th.var > r))))

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.96921    0.02107   140.9   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
                                     edf Ref.df     F p-value
s(x1):factor(I(1 * (th.var > r)))0 4.605  5.647  37.6  <2e-16 ***
s(x1):factor(I(1 * (th.var > r)))1 5.130  6.236 145.6  <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =   0.85   Deviance explained = 85.8%
GCV = 0.091983  Scale est. = 0.087046  n = 200
```

The output of the best model (`model1$res`), indicates a correct fit in term of the overall model intercept (2.96). It also identifies two functions separated by the threshold `r`. The value of `r` is stored in:

```
> model1$mr
[1] 49.556
```

Approximately equal to the used threshold of 50.

Aside from the predicted value of the threshold the function `threshold.gam` also generates a number of other output, which are useful for model diagnostics. These are the value of all the analyzed

thresholds (`$rv`) and the GCV score of all corresponding models (`$gcvv`). Ideally, the profile of the GCV score should have a minimum corresponding to the value of the selected threshold. To plot the results we cannot directly use the `'plot'` function on the final selected model (`model1$res`); it will not work (try it). Thus, we first make a regular GAM model (`model1.1`), using the predicted threshold value (`model1$mr`), and then use this model in the plotting function. Also, remember to manually add the partial residuals to the plot, relative to each selected regime. Code examples follows:

```
> #Make a regular GAM model
> model1.1<-gam(y~s(x1,by=factor(I(1*(th.var>th)))),data=dat)
> summary(model1.1)

Family: gaussian
Link function: identity

Formula:
y ~ s(x1, by = factor(I(1 * (th.var > th))))

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.96921    0.02107   140.9   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
                                       edf Ref.df     F p-value
s(x1):factor(I(1 * (th.var > th)))0 4.605  5.647  37.6  <2e-16 ***
s(x1):factor(I(1 * (th.var > th)))1 5.130  6.236 145.6  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =   0.85   Deviance explained = 85.8%
GCV = 0.091983  Scale est. = 0.087046  n = 200
>
```

Let us take a look at the plots from the threshold.gam function. We use the same code used in the example 1 above:

```
> #Partial residuals
> predX1<-predict.gam(model1.1,type='terms')[,1][dat$th.var<=th]
> predX2<-predict.gam(model1.1,type='terms')[,2][dat$th.var>th]
> res<-residuals(model1.1)
> resX1<-predX1+res[dat$th.var<=th]
> resX2<-predX2+res[dat$th.var>th]
>
> #Plotting
> attach(dat)
> dev.new(width=7,height=7)
> par(mfrow=c(3,2))
> plot(model1.1,pch=1,select=1,main='Sin',rug=F,ylab='Anomalies')
> points(x1[dat$th.var<=th],resX1,pch=1)
> rug(x1[dat$th.var<=th])
```

```
> plot(model1.1,pch=1,select=2,main='Cosin',rug=F,ylab='Anomalies')
> points(x1[dat$th.var>th],resX2,pch=1)
> rug(x1[dat$th.var>th])
> plot(y=residuals(model1$res),x=predict(model1$res),main='Fitted vs
residuals',xlab='Fitted', ylab='Residuals')
> hist(residuals(model1$res),main='Residuals histogram')
> plot(model1$rv,model1$gcvv,type='l',xlab='Threshold
variable',ylab='GCV')
> abline(v=model1$mr)
> detach(dat)
```

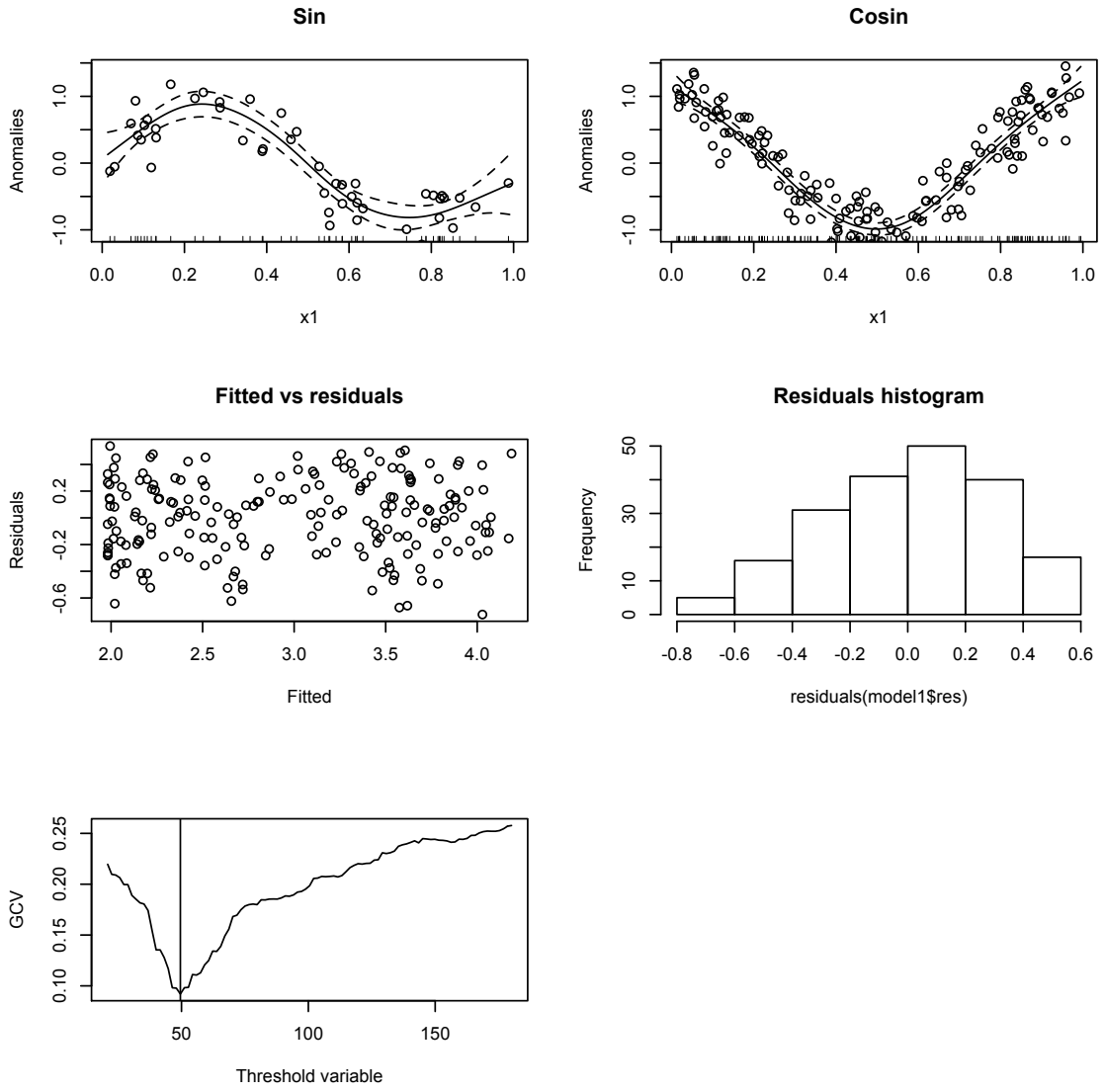


Fig. 3. Results from model in example 2. The plots in the first row show the covariate effect. The plots on the second row show model diagnostic on the full residuals. The last plot shows a profile of the GCV score as a function of the threshold variable (`th`). The vertical line shows the position of the threshold selected by the model to differentiate among the two regimes shown in the first row.

Note that the order in which the gam additive effects are plotted is the same in which they have been specified in the model formula. In this particular case, the first term corresponds to the threshold variable being < r (this is the '0' in the logical vector `I(1*(th.var>th)))`), which in turn corresponds to the sine function.

Example 3: univariate threshold with variable intercept
We now simulate data from a model with two different functions (sin and cosin) and intercepts (3.00, 5.00).

```
> set.seed(999)
> set.seed(999)
> x1<-runif(200)
> th.var<-1:length(x1)#threshold variable
> y1<-3+sin(2*pi*x1[1:49])+0.3*rnorm(49)
> y2<-5+cos(2*pi*x1[50:200])+0.3*rnorm(151)
> y<-c(y1,y2)
> dat<-data.frame(x1=x1,th.var=th.var,y=y)
> dat<-dat[sample(order(th.var)),]
```

The correct formulation of the GAM algorithm is a bit tricky, as it requires the specification of the two intercept terms as a function of the threshold:

```
> model2<-
threshold.gam(formula(y~factor(I(1*(th.var>r)))+s(x1,by=factor(I(1*(th.
var>r))))-1),a=0.1,b=0.9,nthd=100,data=dat,threshold.name='th.var')
> summary(model2$res)

Family: gaussian
Link function: identity

Formula:
y ~ factor(I(1 * (th.var > r))) + s(x1, by = factor(I(1 * (th.var >
    r)))) - 1

Parametric coefficients:
                              Estimate Std. Error t value Pr(>|t|)
factor(I(1 * (th.var > r)))0  2.92863    0.04352    67.3   <2e-16 ***
factor(I(1 * (th.var > r)))1  4.98577    0.02414   206.6   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
                                     edf Ref.df      F p-value
s(x1):factor(I(1 * (th.var > r)))0 4.551  5.585  37.22  <2e-16 ***
s(x1):factor(I(1 * (th.var > r)))1 5.063  6.163 148.71  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
R-sq.(adj) =  0.938   Deviance explained = 99.6%
GCV = 0.092781  Scale est. = 0.087393  n = 200

> model2$mr
[1] 49.556
```

Even in this case the model has correctly estimated the value of the intercept and of the threshold. See figure below. The reader is invited to make a plot of the additive effects, model diagnostics and GCV profile (or you could simply look in the R script for the code…).

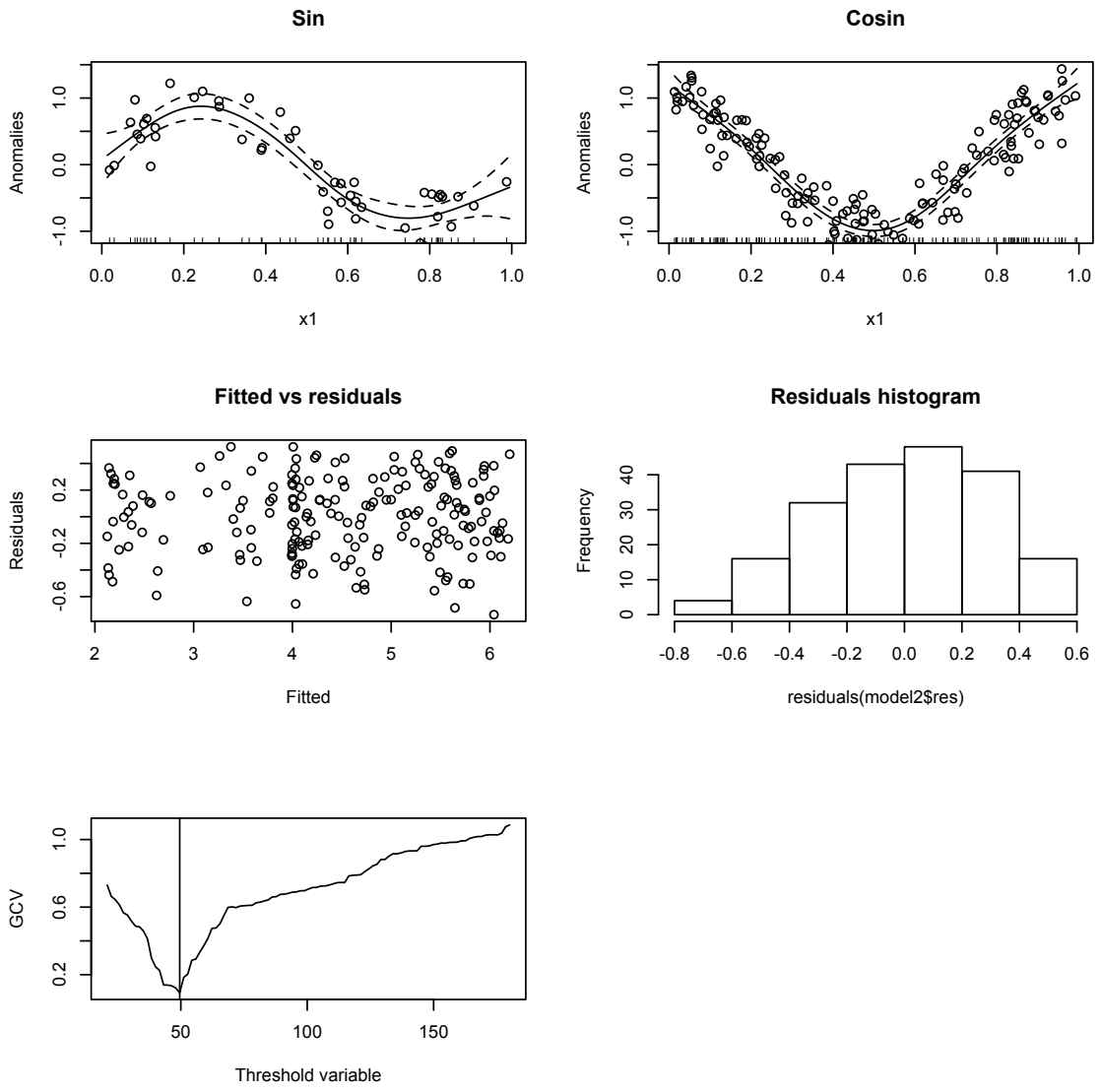


Fig. 4. Results from model in example 3 with two different functions and two different intercepts.

Lastly, we need to address the fact that we could have written the same model of example 3 with an alternative formulation that allows each

function (sin and cosin) to be scaled to their actual magnitude, as opposed to being centered around 0. Recall that to do that all we need to do is to specifiy the two regimes no longer as a factor, but with indicator variables (u, 0 and 1). We need as many indicator variables as there are levels within the categorical covariate var.th (2 in this case). The model syntax also requires that we remove the model intercept, which is readily acoomplished by including -1 at the end of the formula. Code follows:

```
> model2.2<-
gam(y~s(x1,by=I(1*(th.var<=th)))+s(x1,by=I(1*(th.var>th)))-
1,data=dat)
```

The reader is invited to verify that this formulation has a summary output similar to the one from model2.1 (except for the parametric intercept terms of course) and that the plot for the two functions is no longer centered around zero.

Example 4: Barents Sea cod

In the Barents Sea, temperature and subadult cod may have a nonadditive (i.e., indirect) effect on age-1 cod abundance. In particular, subadult cod may increase or decrease the overlap with juvenile cod (and thus the foraging impact) depending on the water temperature.

We study the hypothesis using the univariate threshold approach on the Barents Sea data used in Chapter 5. We expect to see a different effect of subadult cod on age-1 abundance over different temperature conditions. In gam and R language this is equivalent to saying that the relationship between age-1 and subadult cod abundance has a variable effect, regulated by the threshold covariate 'temperature'.

```
> age1.th1<-
threshold.gam(formula(age1~s(age4.6,k=5,by=factor(I(1*(temp<=r))))+s(ag
e0,k=5)+factor(trawl.type)),data=BScod,a=0.15,b=0.85,threshold.name='te
mp',nthd=20)
> summary(age1.th1$res)

Family: gaussian
Link function: identity

Formula:
age1 ~ s(age4.6, k = 5, by = factor(I(1 * (temp <= r)))) + s(age0,
    k = 5) + factor(trawl.type)

Parametric coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)           4.3125     0.3532  12.211 3.81e-10 ***
factor(trawl.type)2   2.0979     0.6131   3.422  0.00304 **
---
```

```
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
                                       edf Ref.df      F  p-value
s(age4.6):factor(I(1 * (temp <= r)))0   1      1 12.685  0.00211 **
s(age4.6):factor(I(1 * (temp <= r)))1   1      1  2.295  0.14692
s(age0)                                 1      1 57.122 5.32e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.833   Deviance explained = 86.4%
GCV score = 1.3678  Scale est. = 1.0704    n = 23
>
```

The output indicates that over one ‘temperature regime’ (i.e., the low temperature) the effect of subadult cod on age-1 cod is not significant (p = 0.14). Moreover, the GCV of the threshold model has decreased by a large margin compared to the one of the additive model (1.368, 1.446, respectively). However, these two measures of GCV are not directly comparable. In fact, the GCV formula used in the gam algorithm (Eq. 8) does not properly account for the fact the threshold model has an extra parameter (i.e., the threshold) determined through a search grid. Thus, in this particular case, an appropriate comparison of the model predictive power should be based on the original CV (Eq. 6).

An inspection of the model effects (Fig. 3) shows that the original hypothesis (switch of subadult cod foraging impact) may be supported by the data.

```
> #Make a GAM model
> age1.th1.1<-
gam(age1~s(age4.6,k=5,by=factor(I(1*(temp<=th))))+s(age0,k=5)+factor(tr
awl.type),data=BScod)

> summary(age1.th1.1)
> R-sq.(adj) =  0.833   Deviance explained = 86.4%
> GCV score = 1.3678  Scale est. = 1.0704    n = 23>

> #Partial residuals
> predX1<-predict.gam(age1.th1.1,type='terms')[,2][BScod$temp>th]
> predX2<-predict.gam(age1.th1.1,type='terms')[,3][BScod$temp<=th]
> res<-residuals(age1.th1.1)
> resX1<-predX1+res[BScod$temp>th]
> resX2<-predX2+res[BScod$temp<=th]
> par(mfrow=c(3,2))
> plot(age1.th1.1,select=1,main='Age4.6 effect when temp>Th',rug=F)
> points(BScod$age4.6[BScod$temp>th],resX1,pch=1)
> rug(BScod$age4.6[BScod$temp>th])
> plot(age1.th1.1,select=2,main='Age4.6 effect when temp<=Th',rug=F)
> points(BScod$age4.6[BScod$temp<=th],resX2,pch=1)
> rug(BScod$age4.6[BScod$temp<=th])
> plot(age1.th1.1,residuals=T,pch=1,select=3,main='Age-0 effect')
```

```
> plot(y=residuals(age1.th1$res),x=predict(age1.th1$res),main='Fitted
vs residuals',xlab='Fitted', ylab='Residuals')
> hist(residuals(age1.th1$res),main='Residuals histogram')
> plot(age1.th1$rv,age1.th1$gcvv,type='l',xlab='Threshold variable
(temperature)',ylab='GCV',main='GCV profile')
> abline(v=age1.th1$mr)
```

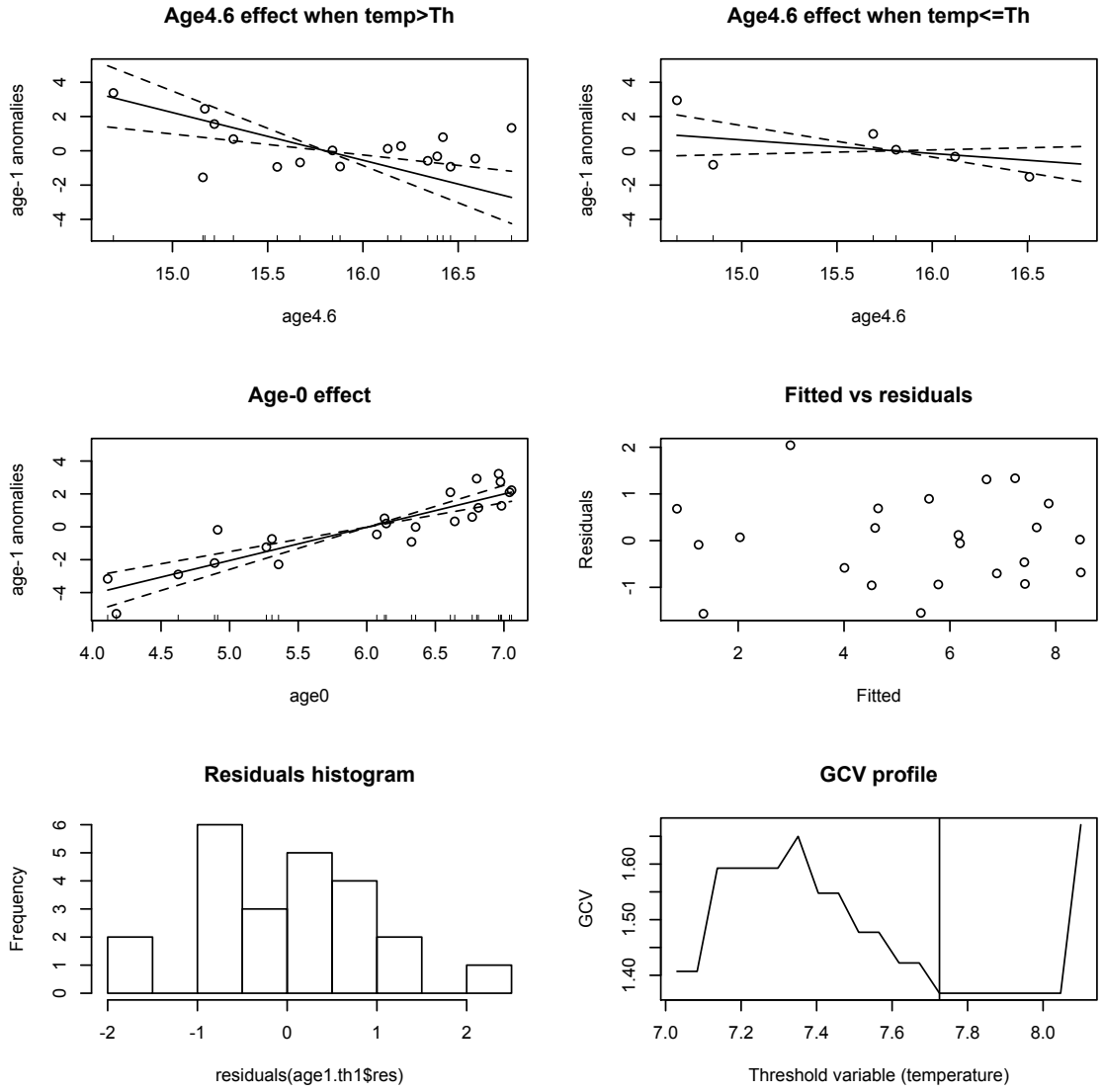


Fig. 5. Results from the `age1.th1` model. The results include the covariate effects (first 3 panels), residual diagnostic (4th and 5th panels) and the GCV profile (last panel).

In particular, subadult cod have a negative effect on age-1 abundance only when the water temperature is above the threshold, otherwise they have no effect. The GCV profile on temperature indicates a global minimum at about 7.7-7.8°C.

An alternative model formulation includes an interaction between capelin and temperature on age-1 cod abundance. Recall from the additivity test

we found that the capelin-temperature interaction was most likely to occur. This again could be the result of a shift in distribution and greater availability of capelin to adult cod (with less predation on age-01 and age-1 cod).

#**capelin and temperature**

```
> age1.th2<-
threshold.gam(formula(age1~s(capelin,k=5,by=factor(I(1*(temp<=r))))+s(a
ge0,k=5)+s(age4.6,k=5)+factor(trawl.type)),
+ data=BScod,a=0.15,b=0.85,threshold.name='temp',nthd=20)
> summary(age1.th2$res)

Family: gaussian
Link function: identity

Formula:
age1 ~ s(capelin, k = 5, by = factor(I(1 * (temp <= r)))) + s(age0,
    k = 5) + s(age4.6, k = 5) + factor(trawl.type)

Parametric coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)          4.3117     0.3703  11.644 5.45e-09 ***
factor(trawl.type)2  2.4025     0.7106   3.381  0.00403 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
                                      edf Ref.df      F  p-value
s(capelin):factor(I(1 * (temp <= r)))0 1.957  2.352  1.493   0.2517
s(capelin):factor(I(1 * (temp <= r)))1 1.000  1.000  3.538   0.0790 .
s(age0)                                1.000  1.000 64.409 4.09e-08 ***
s(age4.6)                              1.796  2.191  4.781   0.0219 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.856   Deviance explained =   90%
GCV score = 1.3989  Scale est. = 0.92729   n = 23
```

From this output we see that above the temperature threshold (i.e., when factor = 0), capelin may not have any effect (P-value = 0.251). It is worth trying to include an effect of capelin and temperature also in the intercept:

```
> age1.th2<-
threshold.gam(formula(age1~s(capelin,k=5,by=factor(I(1*(temp<=r))))+
+ factor(I(1*(temp>r)))+s(age0,k=5)+s(age4.6,k=5)+factor(trawl.type)),
+ data=BScod,a=0.15,b=0.85,threshold.name='temp',nthd=20)
> summary(age1.th2$res)

Family: gaussian
Link function: identity

Formula:
age1 ~ s(capelin, k = 5, by = factor(I(1 * (temp <= r)))) + factor(I(1
*
```

```
    (temp > r))) + s(age0, k = 5) + s(age4.6, k = 5) +
factor(trawl.type)

Parametric coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)                 2.9137     0.4877   5.975 0.000188 ***
factor(I(1 * (temp > r)))1  2.0413     0.5555   3.675 0.004897 **
factor(trawl.type)2         1.9596     0.6065   3.231 0.009970 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
                                      edf Ref.df      F  p-value
s(capelin):factor(I(1 * (temp <= r)))0 3.243  3.697  3.778 0.045106 *
s(capelin):factor(I(1 * (temp <= r)))1 1.000  1.000 15.787 0.002775 **
s(age0)                                2.521  2.964  6.709 0.010157 *
s(age4.6)                              4.000  4.000 12.721 0.000408 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =   0.94   Deviance explained = 97.5%
GCV score = 0.95352  Scale est. = 0.38289   n = 23
```

Indeed, the operation resulted in a much better model, according to GCV. We next plot the GCV profile for the search of the threshold and the smooth term effects with partial residuals (Fig. 5).

```
> #Partial residuals
> predX1<-predict.gam(age1.th2.1,type='terms')[,3][BScod$temp>th]
> predX2<-predict.gam(age1.th2.1,type='terms')[,4][BScod$temp<=th]
> res<-residuals(age1.th2.1)
> resX1<-predX1+res[BScod$temp>th]
> resX2<-predX2+res[BScod$temp<=th]
>
> par(mfrow=c(3,2))
> plot(age1.th2.1,select=1,main='Capelin effect when temp>Th',rug=F)
> points(BScod$capelin[BScod$temp>th],resX1,pch=1)
> rug(BScod$capelin[BScod$temp>th])
> plot(age1.th2.1,select=2,main='Capelin effect when temp<=Th',rug=F)
> points(BScod$capelin[BScod$temp<=th],resX2,pch=1)
> rug(BScod$capelin[BScod$temp<=th])
> plot(age1.th2.1,residuals=T,pch=1,select=3,main='Age0 effect')
> plot(age1.th2.1,residuals=T,pch=1,select=4,main='Age4.6 effect')
> plot(y=residuals(age1.th1$res),x=predict(age1.th1$res),main='Fitted
vs residuals',xlab='Fitted', ylab='Residuals')
> #hist(residuals(age1.th2$res),main='Residuals histogram')
> plot(age1.th2$rv,age1.th2$gcvv,type='l',xlab='Threshold variable
(temperature)',ylab='GCV',main='GCV profile')
> abline(v=age1.th2$mr)
```

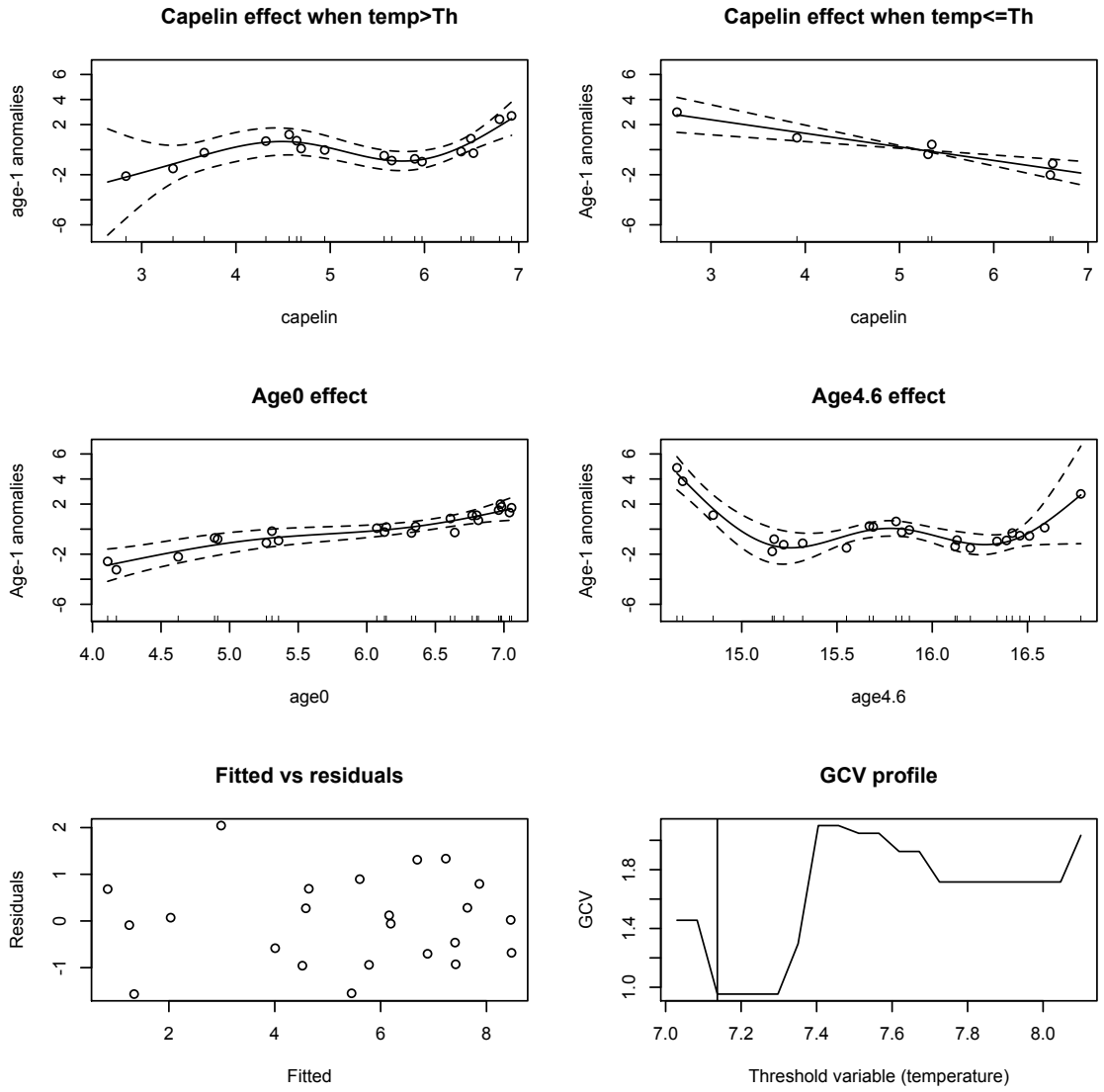


Fig. 6. Results from the `age1.th1` model. The results include the covariate effects (first 4 panels), residual diagnostic (4th panel) and the GCV profile (last panel).

Chapter 10: Genuine cross validation

To compare threshold models with fully additive model formulations it is necessary to account for the fact that the former have an additional parameter (used for the threshold), and that such parameter is assessed through a grid search. Recall that the GCV is only a (good) approximation of the real cross validation (CV) and that does not take into account the fact that a grid search has been put in place to find the value of the threshold. Thus to compare models we use the *genuine* CV, calculated as shown in Eq. 6 by removing one datum at a time from the coefficient estimation and predicting its value from the resulting model. The operation is repeated for as many data points available in the sample size. The functions `gam.cv`, `threshold.gam.cv` compute the genuine CV in a gam and univariate threshold model, respectively.

Example 1: Barents Sea cod.

We go back to the Barents Sea cod example to calculate the genuine CV from the three models formulation shown in the examples (additive, interaction between temperature and subadult cod, interaction between temperature and capelin).

```
> age1.2.cv<-
gam.cv(response.name='age1',formula(age1~s(age0,k=5)+s(age4.6,k=5)+
+factor(trawl.type)),data=BScod)
> age1.2.cv$cv
[1] 2.184501

> age1.th1.cv<-
threshold.gam.cv(response.name='age1',formula(age1~s(age4.6,k=5,by=fact
or(I(1*(temp<=r))))+s(age0,k=5)+factor(trawl.type)),data=BScod,a=0.15,b
=0.85,threshold.name='temp',nthd=20)
> age1.th1.cv$cv
[1] 2.623411

> age1.th2.cv<-threshold.gam.cv(response.name='age1',
formula(age1~s(capelin,k=5,by=I(1*(temp<=r)))+
+ s(capelin,k=5,by=I(1*(temp>r)))+s(age0,k=5)+s(age4.6,k=5)+
factor(trawl.type)),data=BScod,a=0.15,b=0.85,threshold.name='temp',nthd
=+ 20)
> age1.th2.cv$cv
[1] 4.397023
```

The results indicate that the fully additive model formulation is the best among the three. Thus, for this particular set of data using a threshold variable may have improved the within data model predictions (R^2), but yields worse results when predicting new data values.

The genuine CV, as introduced in the Barents Sea cod example works very efficiently for relatively small sample sizes (e.g., < 100). For larger sample sizes the computational time becomes expensive. In such circumstances it may be useful to cross-validate the model against a

random sample of more than one data point. This issue is illustrated in the next example, using Atlantic cod egg data from the Skagerrak coast in southern Norway.

Example 2: Spatial distribution of cod eggs (Knutsen et al. 2007). The objective of this analysis was to study cod egg distribution along fjords of the Norwegian coast. The main hypothesis was that adult cod spawn very close to the coast in order to optimize egg retention within each fjord. This would in turn contribute to the high degree of genetic population structure of Norwegian coastal cod. We tested the effect of local geographic features on cod egg distribution using different GAM formulations. The response variable (*y*) was the natural logarithm of egg density (the sum of all stages), standardized by volume filtered through the sampling net. The inspected predictor variables or covariates were the natural logarithm of bottom depth (*bdepth,* the logarithm was taken to achieve a uniform distribution of sampling depths), distance in meters from the sampling station to the sill location of the fjord (*distsill*), depth (*sdepth*) and latitude (*latsill*) of the sill. The distance from the sill was negative if the sampling station was inside of the sill and positive otherwise. Only tows located at less than 10 km from the sill (either side) and with positive egg count were included in the analysis. Removal of tows outside of the 10 km range (7 out of 233) was necessary to reach a uniform spread of samples on either side of the sill. Removal of zero-egg tows (9 out of 233) was necessary to stabilize the variance and normalize the distribution. The final number of stations used in the data set was 217.

We compared results from 3 different model formulations, each addressing a specific hypothesis (Table 1). The notation in the Table is the following: z = bottom depth, d = distance from the sill, K_t = year effect, K_f = fjord effect, l = latitude of the sill, s = sill depth.

Model type	Formulations	Threshold
Model 1	$y_{i,t} = k_t + k_f + g_1(z_i) + g_2(d_i) + \varepsilon_i$	No threshold
Model 2	$y_{i,t} = k_t + k_f + g_1(z_i) + \begin{cases} g_2(d_i) + \varepsilon_i & \text{if } l \le r \\ g_3(d_i) + \varepsilon_i & \text{otherwis} \end{cases}$	Latitude
Model 3	$y_{i,t} = k_t + k_f + g_1(z_i) + \begin{cases} g_2(d_i) + \varepsilon_i & \text{if } s \le r \\ g_3(d_i) + \varepsilon_i & \text{otherwis} \end{cases}$	Sill depth

In Model 1 (the 'null' model) it is assumed that the distribution of eggs along a fjord (i.e. relationship between egg density and distance from the sill) is similar across all sampled fjords. This hypothesis is in contrast to those of Models 2 (the 'geographic' effect model) and 3 (the 'sill' effect model), in which the distribution of eggs along a fjord can change according to a threshold value of either the latitude or the depth of the sill, respectively. The selection of the threshold was based on the minimization of the generalized cross validation (GCV) among a variety of GAM models spanning the entire range of possible threshold values. For more details on threshold estimation see Ciannelli et al. (2004). The GCV, although suitable for comparing models having a similar formulation (e.g. models with thresholds), does not properly account for the presence of additional parameters (i.e. the threshold) when comparing models with different formulations. Thus, to properly compare models with and without a threshold, we numerically computed the genuine cross validation as follows. A random selection of 20 data points (about 10% of the entire data set) was excluded from the model calibration. The resulting model was then used to predict these 20 out-of-sample observations, and the mean-squared predictive error was calculated. The operation was repeated 500 times, with the final CV being the mean among the 500 evaluations of the mean-squared predictive error. Low CV values indicate models with better fit to the data and lower complexity.

```
> #Model 1: Additive GAM
> gam5<-> gam1<-gam(logd~year+area+s(distsill)+s(log(bdepth))-
1,data=codegg)
> summary(gam1)
#R-sq.(adj) =  0.687   Deviance explained = 87.1%
#GCV score = 0.57103  Scale est. = 0.48576   n = 217

> #Model 2: Threshold, latitude of the sill
> gam5<-threshold.gam(logd~year+area+s(distsill,by=I(1*(latsill<=r)))+
+ s(distsill,by=I(1*(latsill>r)))+s(log(bdepth))-1,
+ data=codegg,threshold.name='latsill',nthd=20,a=0.1,b=0.9)
> summary(gam5$res)
#R-sq.(adj) =  0.727   Deviance explained = 89.2%
#GCV score = 0.51704   Scale est. = 0.4235    n = 217

> #Model 3: Threshold, depth of the sill
> gam7<-
threshold.gam(logd~factor(year)+area+s(distsill,by=factor(I(1*(sdepth<=
r))))+
+ s(log(bdepth))-1,data=codegg,threshold.name='sdepth',
+ nthd=20,a=0.1,b=0.9)
> summary(gam7$res)
#R-sq.(adj) =  0.722   Deviance explained = 88.6%
#GCV score = 0.51179   Scale est. = 0.43149   n = 217
```

The figures from each model are shown below:

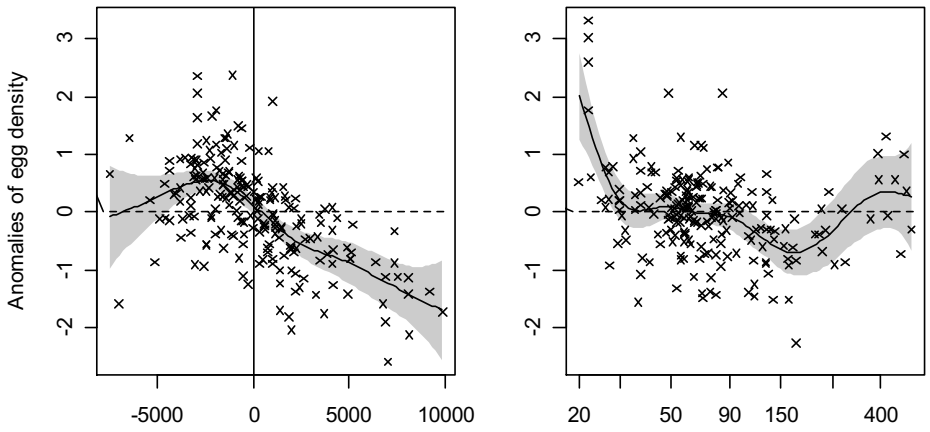


Figure 1. Model 1. Effect of distance from sill (in meters, left) and bottom depth (in meters, log scale, right) on cod egg density.

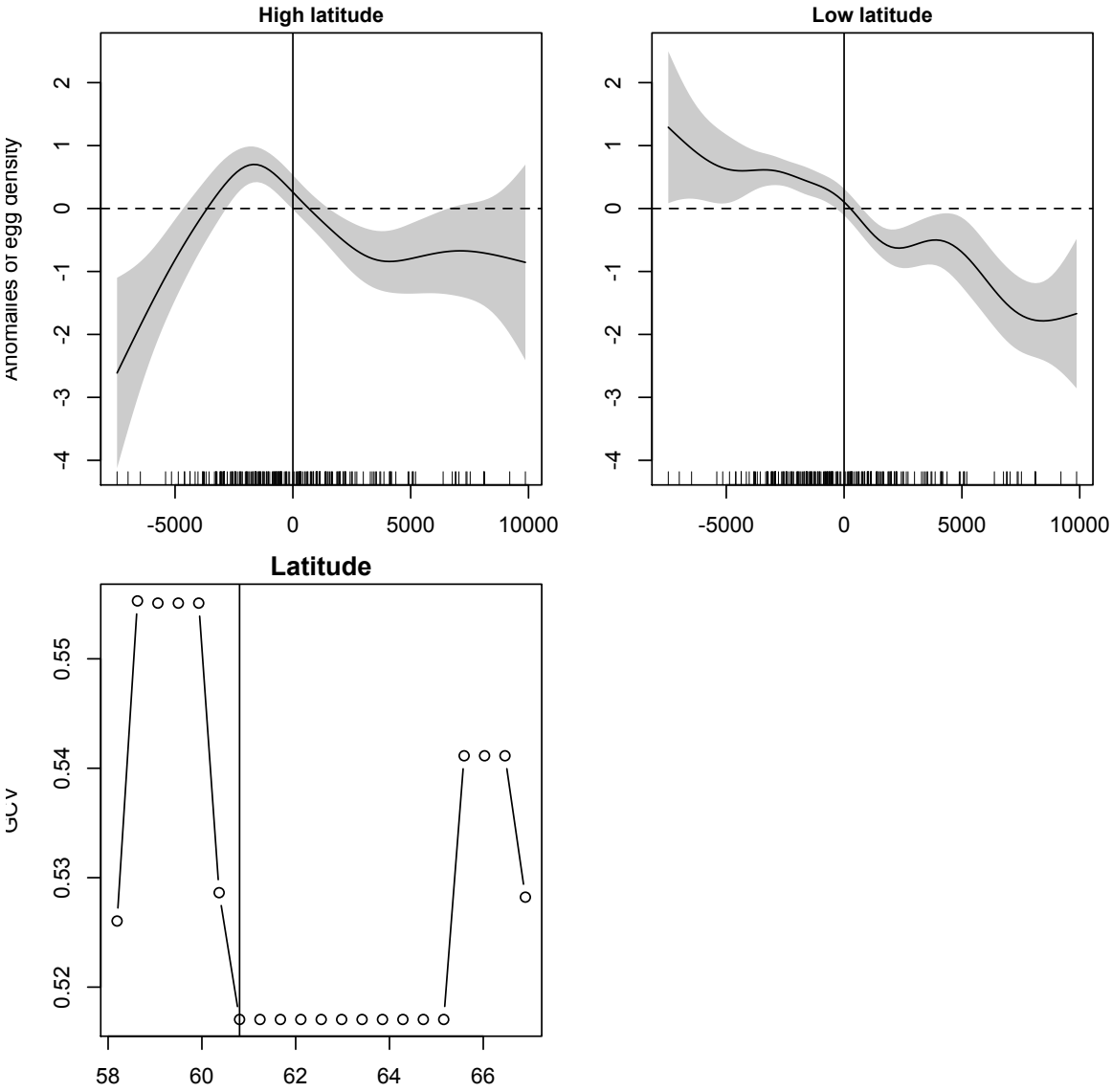


Figure 2. Model 2: Effect of distance from the sill for fjords located at high (left) and low (right) latitudes. GCV profile for latitude threshold estimation is shown on the bottom panel.

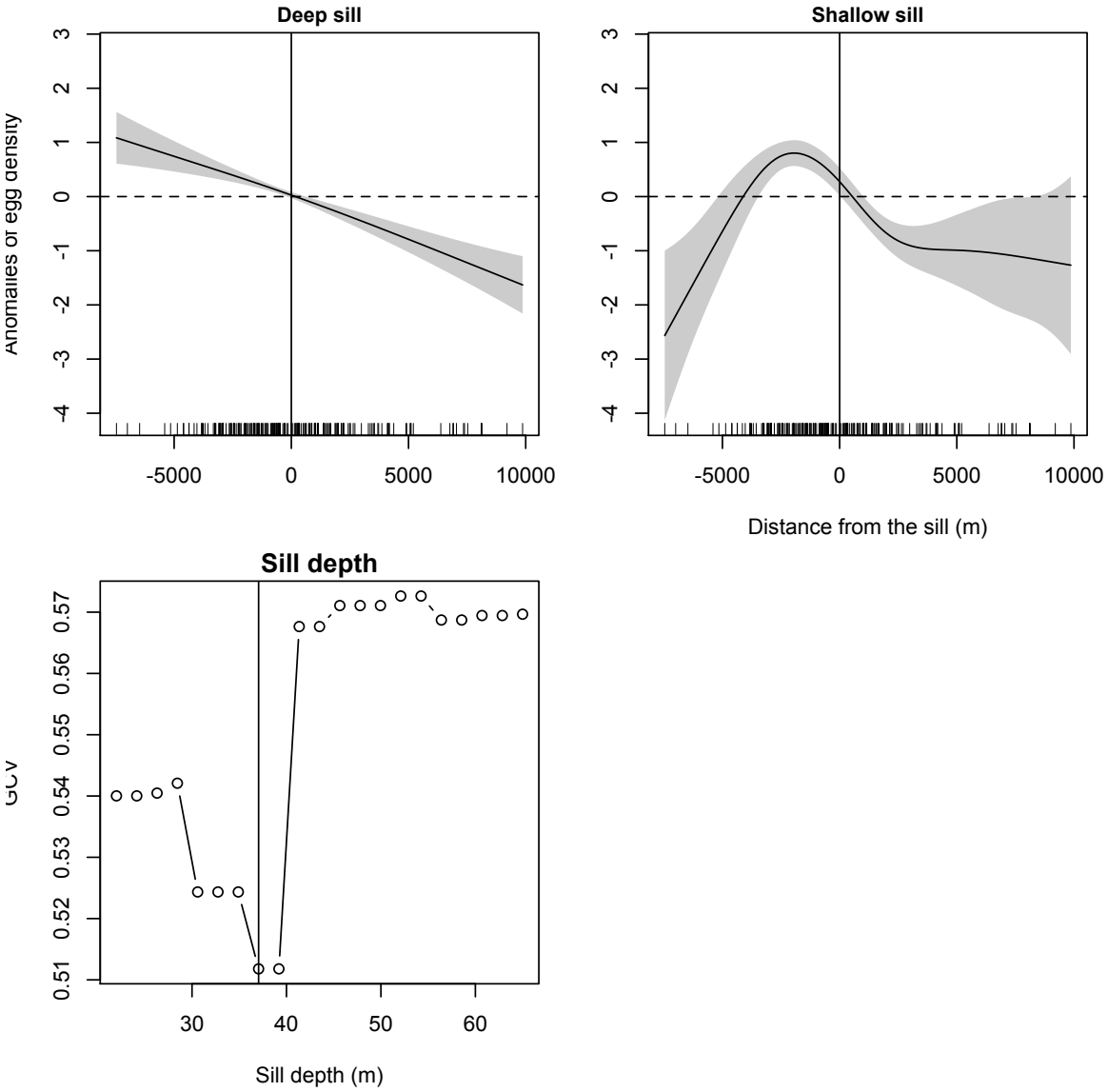


Figure 3. Model 3: Effect of distance from the sill (m) on cod egg density for fjords with deep (left) and shallow (right) sill. GCV profile for sill-depth threshold estimation is shown on the bottom panel.

We then proceed with the estimation of the genuine CV:

```
> n<-20
> subset<-codegg
> cv1.na<-1:500
> cv1.a<-cv1.na*NA
> cv2.na<-cv1.na*NA
> for(i in 1:length(cv1.na)){
+ index<-sample(1:nrow(subset))
+ data.in<-subset[index[1:(length(index)-n)],]
+ data.out<-subset[index[(length(index)-(n-1)):length(index)],]
+ data.in$area<-factor(data.in$area)
+ data.in$year<-factor(data.in$year)
+ data.out$area<-factor(data.out$area)
+ data.out$year<-factor(data.out$year)
```

```
+ gam.na<-
gam(logd~year+area+s(distsill,by=factor(I(1*(sdepth<=37.05))))+s(log(bd
epth))-1,data=data.in)
+ gam.na2<-gam(logd~year+area+s(distsill,by=factor(I(1*(latsill<=
60.80326))))+s(log(bdepth))-1,data=data.in)
+ gam.a<-gam(logd~year+area+s(distsill)+s(log(bdepth))-1,data=data.in)
+ pred.na<-predict.gam(gam.na,newdata=data.out)
+ pred.na2<-predict.gam(gam.na2,newdata=data.out)
+ pred.a<-predict.gam(gam.a,newdata=data.out)
+ cv1.na[i]<-mean((pred.na-data.out$logd)^2)
+ cv2.na[i]<-mean((pred.na2-data.out$logd)^2)
+ cv1.a[i]<-mean((pred.a-data.out$logd)^2)}
>
> mean(cv1.na)#CV from model 3
[1] 0.5522084
> mean(cv2.na)# CV from model 2
[1] 0.6109599
> mean(cv1.a)# CV from model 1
[1] 0.6246828
```

These results indicate that as expected there is a strong effect of distance from the sill on cod egg density, regardless of the model formulation. In particular, egg density increases inshore. Model 3 (threshold, depth of the sill) is the best among the three, in term of its prediction power or CV. Model 2 is the second best, and model 1 is the worst. From the covariate effects we see that in fjords having a shallow sill (< ~37 m) cod egg density is highest immediately inshore of the sill, while in fjords with deep sill (> ~37m) cod egg density linearly decreases toward the offshore direction (Fig. 3). Also in fjords located at high latitudes (> ~61^{o}N) cod egg density remains fairly high also offshore of the sill, while this pattern is not observed in Sothern Norway fjords (Fig. 2).

Although the highest egg densities were consistently found in the inshore sections of the sampled fjords, some eggs where also found offshore. This raises the interesting question of where these offshore eggs originate. No firm conclusion can be reached from the data available in our study; however, it is still worthwhile to discuss some potential mechanisms. The origin of offshore eggs most likely differs for the different regions sampled. In southern Norway, the eggs found offshore may have been transported from the inshore locations due to occasional reverse conditions in the typical fjord circulation. Alternatively, offshore eggs in southern fjords may have originated in the North Sea, having been entrained in the fast coastal flow, or they may have been the result of some limited offshore spawning of costal cod. In northern Norway, the model depicts a slightly different picture in that the egg density stabilizes at a plateau outwards, and does not further decrease. This is expected, as oceanic migratory Arcto-Norwegian cod are present in this northern area and spawn in offshore locations. These cod have a life-history strategy of performing long spawning migrations from the Barents Sea

and south along the Norwegian coast, where they spawn. Because the eggs float, they are exposed to the strong ocean currents in this region which transport them towards the nursery areas in the Barents Sea.

The last thing to do is to check how the model genuine corss validation from each of the three formulations has converged toward its mean. Recall that we are using 500 randomly selected series of 20 out of sample predictions to validate our models. Is 500 iterations enough? One way to check that is to plot the mean of the prediction error at progressively larger increase of the oteration number. If, the progressive mean converges quickly toward the final mean of 500 iterations, then we have a pretty strong evidence for convergence. Enough talking - let us check this. The code is below. And the figure follows.

```
> #Check convergence
> cv1.na_cum<-cv1.na*NA
> cv2.na_cum<-cv2.na*NA
> cv1.a_cum<-cv1.a*NA
> for(i in 1:length(cv1.na)){
+                           cv1.na_cum[i]<-mean(cv1.na[1:i])
+                           cv2.na_cum[i]<-mean(cv2.na[1:i])
+                           cv1.a_cum[i]<-mean(cv1.a[1:i])
+ }

> dev.new()
> par(mfrow=c(3,1))
>
plot(1:length(cv1.na_cum),cv1.na_cum,type='l',xlab='Iteration',yl
ab='gCV',main='Sill depth model')
> abline(h=mean(cv1.na),lty=2)
>
plot(1:length(cv2.na_cum),cv2.na_cum,type='l',xlab='Iteration',yl
ab='gCV',main='Geographic model')
> abline(h=mean(cv2.na),lty=2)
>
plot(1:length(cv1.a_cum),cv1.a_cum,type='l',xlab='Iteration',ylab
='gCV',main='Additive model')
> abline(h=mean(cv1.a),lty=2)
```

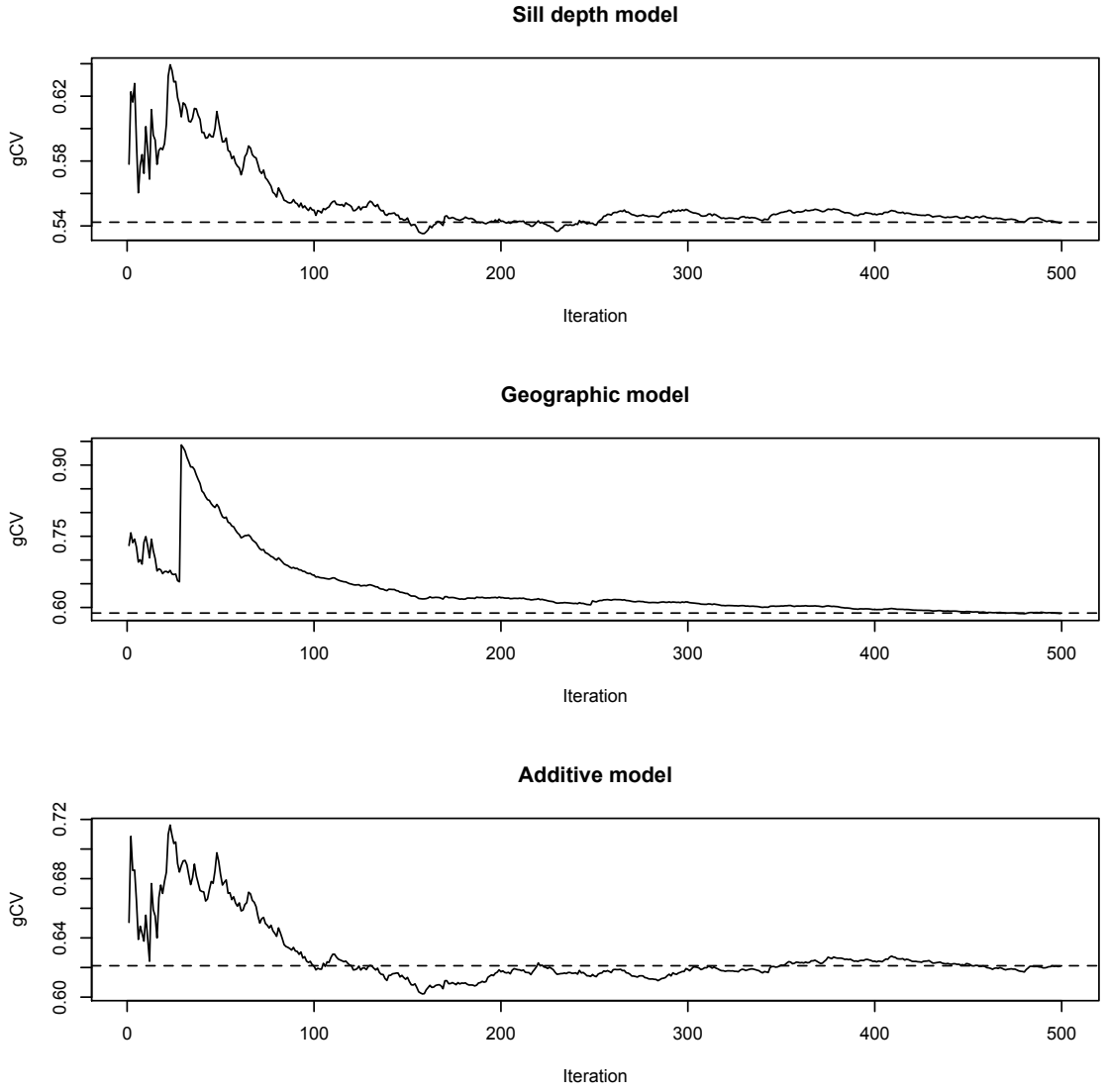


Figure 4. Genuine corss validation as a function of the iteration steps used to calculate the long-term average (500).

All three models seem to converge rather well toward the long-term average prediction error. The geographic model (middle plot) does havea very worrisome spike in the gCV right around the 35th iteration. Upon checking, that iteration resulted in a very large out of sample prediction (on the oprder of 9). This is an indication that the geographic model is very sensitive to some data points.

Chapter 11: Variable coefficients GAM

In this section we proceed to examine a more general form of GAM with variable regression coefficients. The threshold GAM is a special case of a variable coefficient GAM, where the regression parameters of a design variable X_1 are abruptly changing as a function of a threshold variable X_2. Typically X_2 is a factor, with two levels, one above and the other below the threshold value. Here we consider the more general case in which the regression coefficients of the design variable X_1 are continuously and smoothly (i.e., no break in the smooth with continuous first and second derivative) changing as a function of a second variable, X_2. Namely:

$$y_i = a_i + b_i X_{1i} + e_i$$

where:

$$a_i = g_1(X_{2i})$$
$$b_i = g_2(X_{2i})$$

Substituting for *a* and *b*:

$$y_i = g_1(X_{2i}) + g_2(X_{2i})X_{1i} + e_i$$

Note that both the intercept (*a*) and the slope (*b*) are changing in relation to X_2. Also note that while the functions (*g*) that link X_2 to the regression coefficients (*a* and *b*) can be smooth, the design variable X_1 is still linearly related to the dependent variable Y. Therefore, the variable coefficient GAM with continuous terms is no more no less than a fancy linear regression model with coefficients that are determined by a smooth function (g). From an ecological perspective, variable coefficients GAM are very useful to model the interaction between two covariates (X_1 and X_2) by dropping the assumption of the strong nonlinear (i.e., threshold) effect of these two covariates. However to free ourselves from the threshold assumption we pay the price of having to assume a linear effect of the covariate X_1 at any given value of X_2. While apparently limiting, this assumption is, ecologically, not too far fetched, and increases tremendously our inference potential. We will get back to this concept in the next chapter, when dealing with spatial data. For the time being we stick with the generalized case of variable coefficients GAM applied to a fictitious data set. Specifically, we will learn how to formulate a variable

coefficient GAM, estimate the parameters *a* and *b*, and extract the estimated values and standard errors from the fitted gam object – all of which is indispensable to make ecological inference when using real data.

First, we generate the data and examine the relationships among covariates. Note that in the fictitious data set, while the values of intercept *a* are centred around 0, those of the slope *b* are not, in particular they are all assumed to be positive, ranging from 2 to 4 (because we added the constant 3).

```
> set.seed(999)
> X2<-runif(100)
> X1<-runif(100)
> Y<-sin(2*pi*X2)+(3+cos(2*pi*X2))*X1+0.3*rnorm(100)
> dat<-data.frame(X1=X1,X2=X2,Y=Y)
> pairs(dat)
```

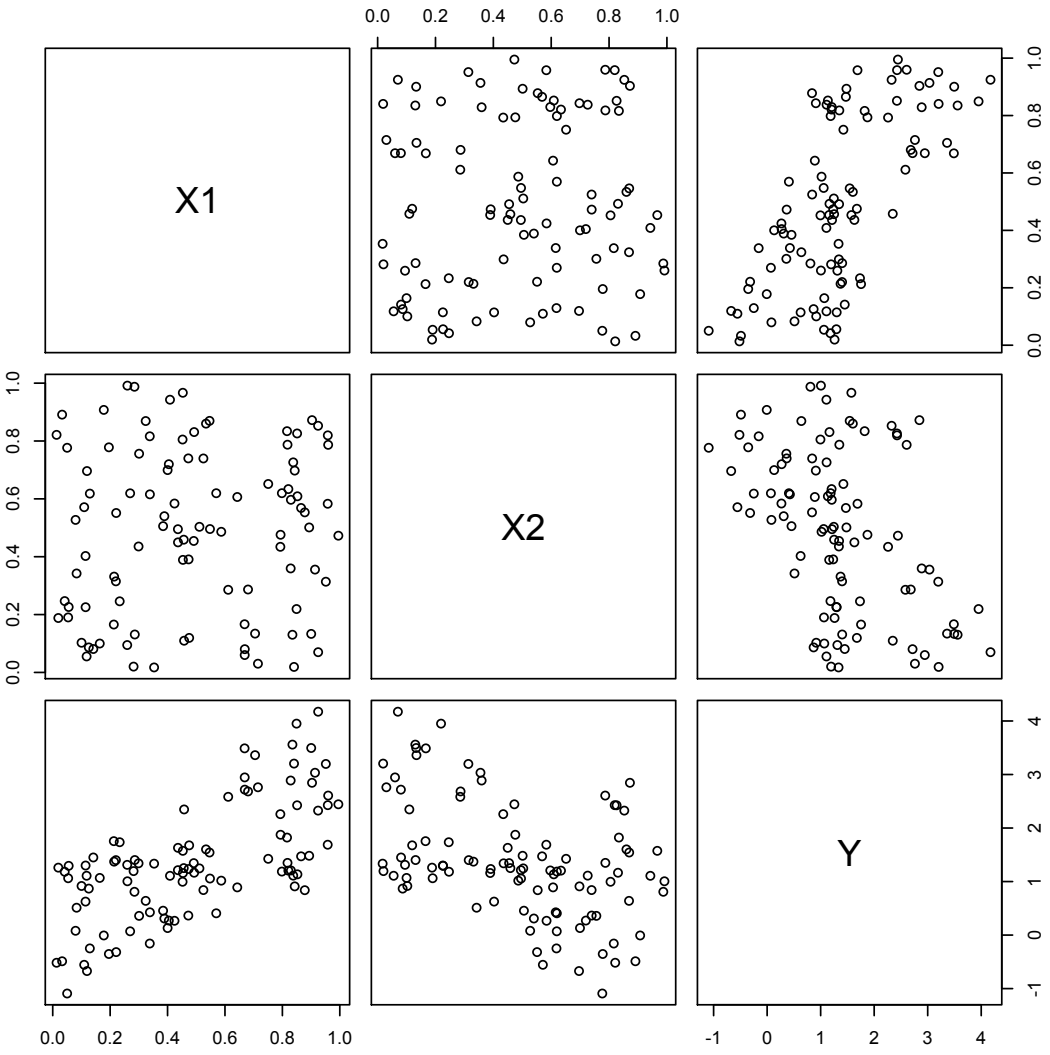


Figure 1. Relationships among covariates generated for the variable coefficient GAM. Note that while in reality the relationship between X2 and Y is depicted as a nonlinear effect, when in reality X2 is only modifying the slope and intercept.

We proceed by fitting the above data set using first a straightforward additive GAM with X_1 and X_2 as covariates and then a proper variable coefficient GAM, which assumes a linear effect of X_1 on Y but with coefficients dependent on the value of X_2.

```
> #Fully additive GAM
> gam1<-gam(Y~s(X1)+s(X2),data=dat)
> summary(gam1)
> #R-sq.(adj) =  0.917   Deviance explained = 92.2%
> #GCV score = 0.10593  Scale est. = 0.097737  n = 100
```

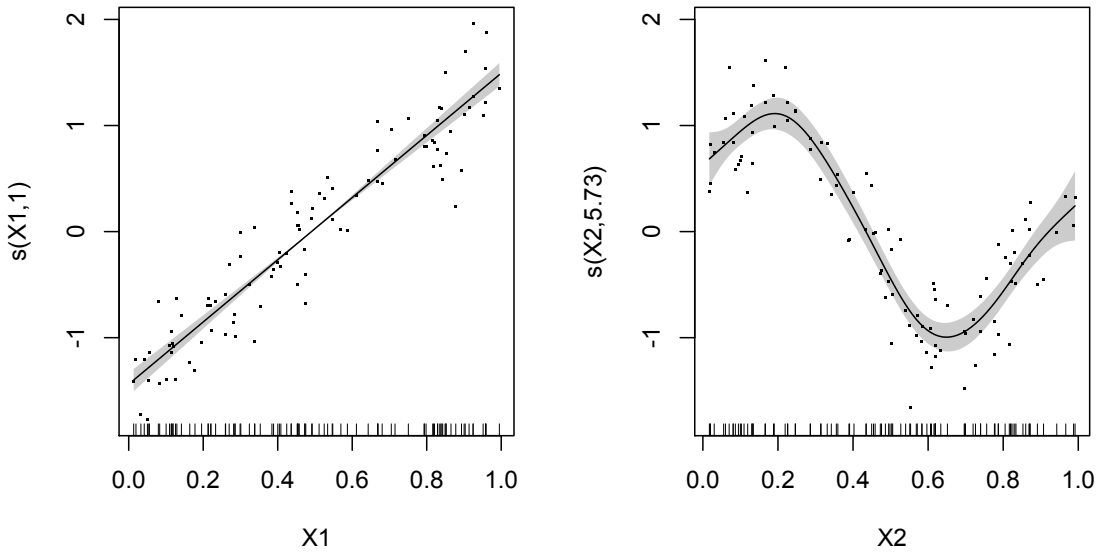


Figure 2. Estimated effects of the additive model gam1.

Note that all the estimated smooth terms of the additive gam are centred around zero while in reality we generated a response variable (Y) that mostly has positive values, up to 4. Therefore, in the additive gam, the actual scale of the response variable is absorbed by the intercept term, which in this case is expectedly positive (1.356) and highly significant ($p<<0.001$).

We now proceed with the variable coefficient GAM. Here the argument 'by' with the smooth term signifies the product of the smooth term by the covariate within the smooth term (i.e., X_1).

```
> #Variable coefficient GAM
> gam2<-gam(Y~s(X2)+s(X2,by=X1),data=dat)
> summary(gam2)
#R-sq.(adj) =  0.939   Deviance explained = 94.5%
```

```
#GCV score = 0.079154  Scale est. = 0.070953  n = 100
> plot(gam2,pages=1,res=T,shade=T)
```

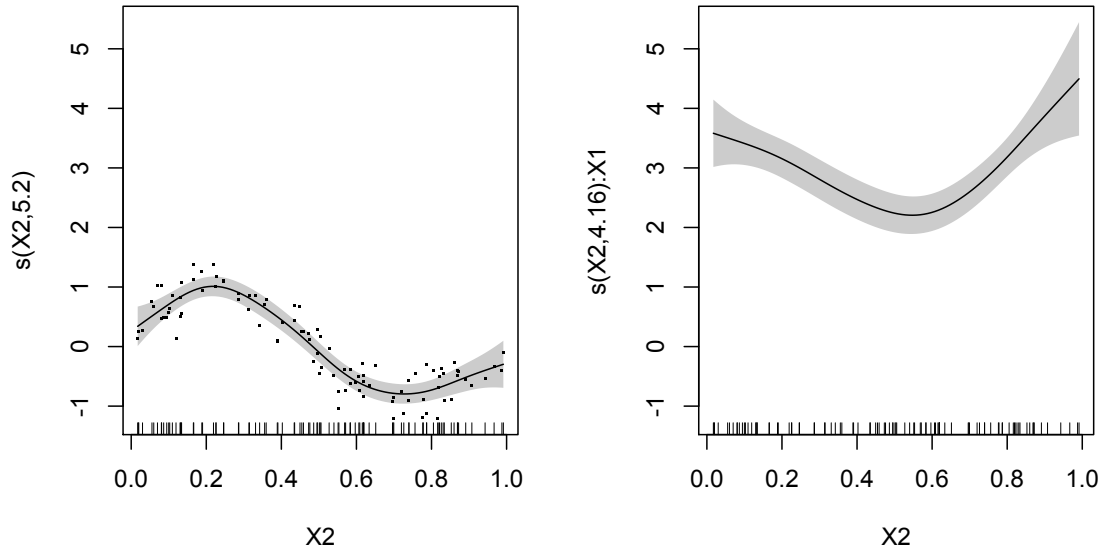


Figure 3. Estimated effects for the variable coefficient GAM (gam2).

Note that with the variable coefficient GAM formulation we obtained a better fit both in term of R^2 and in term of GCV. Most importantly, note that now the effect of X2 (i.e., the regression coefficients) is properly displayed on the right scale, with a positive increase of about 3 (due to the addition of the constant 3 to the value of the slope in the fictitious data set). In producing this figure, I left the Y-axis labels equal to the names in the prediction data frame. Thei is helpful to understand what is being plotted. The left side panle of Figure 3 shows the values of the variable intercept term (a_i in the equation on page 88), while the right panel shows the values of the slope term (b_i in the equation on page 88).

We could settle with the gam2 formulation, since it constitutes a significant improvement compared to the fully additive formulation. We are however concerned by the fact that the opposite ends of the smooth terms for the estimated effect of X_2 are not ending at the same value. This is not what we generated in the data with the sine and cosine functions, which are cyclical. Of course, we are aware of this issue because the data were generated. In nature, we may have reasons to suspect that the effect of a covariate is cyclical (e.g., effect of time of the day or day of the year) and therefore force it to be so by using cyclical cubic regression splines. Here is how to do so in the same fictitious data set:

```
> gam3<-gam(Y~s(X2,bs='cc')+s(X2,by=X1,bs='cc'),data=dat)
> summary(gam3)
```

```
#R-sq.(adj) =  0.943   Deviance explained = 94.9%
#GCV score = 0.076033  Scale est. = 0.066852  n = 100
> plot(gam3,pages=1,res=T,shade=T)
```

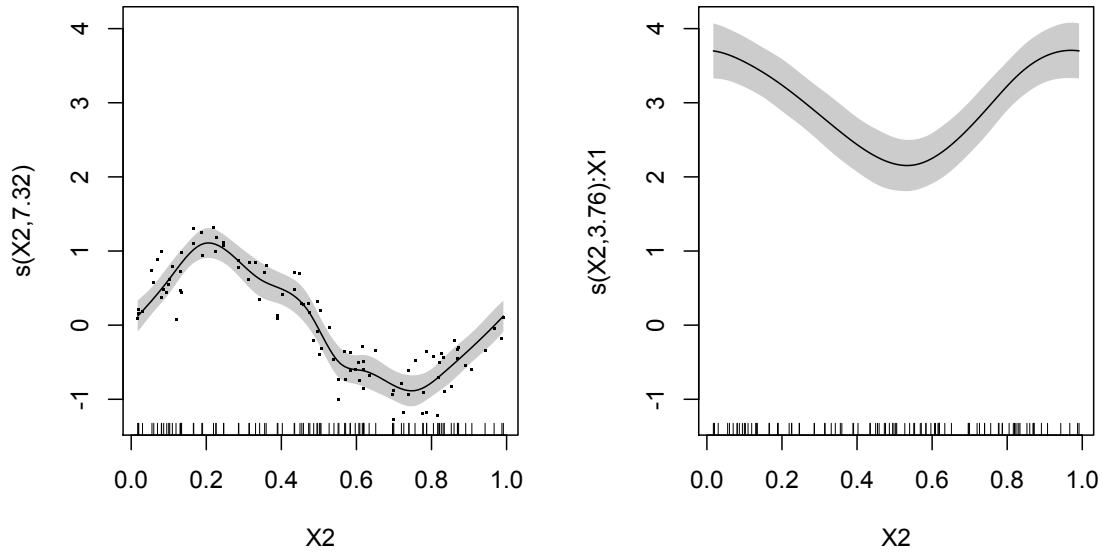


Figure 4. Estimated effect for the variable coefficient gam, with cyclical cubic regression splines.

The model fit is only marginally improved with cyclical splines, however our feeling is much at rest now, knowing that we made ends meet. One tiny concern left is the excessive wiggliness of the first smooth term (which depicts the intercept term), as also indicated by the large estimated degrees of freedom. One way to correct for this effect is by using the '`gamma`' (γ) multiplier within the gam function. The argument γ is a factor of the λ in the penalty function shown way back, in equation 5. By setting $\gamma > 1$ we assign extra penalty to excessive wiggly functions. By default γ is assumed to be equal to 1. Another way to obtain less wiggly fit is by constraining the degrees of freedom within a smooth term, by setting `k` equal to a relatively small number (say 5). The advantage of using γ as opposed to `k` to correct for excessive wiggliness is that γ is a control parameters for ALL terms, as opposed to `k`, which acts on one covariate at a time. Also, γ exerts its effect on the penalty function only when the estimated fits for the covariate are excessively wiggly while `k` is a cap imposed on the degrees of freedom of a smooth, regardless of whether that smooth term was very wiggly to start with. The reader is invited to experiment with γ and `k` with gam3 model and setting `gamma = 1.5` and `k = 5` in separate formulations.

We noted that the estimated effects for the covariate X_2 are no longer cantered around zero in a variable coefficient gam. More in general, this

happens when a 'by' argument followed by a numerical variable is included in the smooth function. It is **important** to realize that this only occurs if the 'by' covariate is numerical. Whereas when the 'by' covariate is a factor (cf., threshold gam formulation) each smooth term is still centred on zero. This fact has very important consequences, namely there is no need to include a regime specific intercept term when that regime is denoted with a numerical as opposed to a factor covariate.

Suppose that we are now interested in extracting the linear regression coefficients (b_i) estimated with the gam3 formulation (i.e., variable coefficient with cyclical cubic splines). Such operation is not very straightforward, and requires some coding, which is shown below. In the next chapter, we will present a case study based on field data to illustrate the ecological importance of being able to extract regression coefficients (and relative error information) from a gam fit.

```
> predX2<-predict(gam3,type='terms')[,2]
> SEX2<-predict(gam3,type='terms',se.fit=T)[[2]][,2]
> res<-residuals(gam3)
> slope_X1<-predX2/X1
```

Note that to obtain the actual value of the slope coefficient we need to divide the predicted term extracted from the GAM object by X1. Can you explain why?

```
> dev.new()
> plot(sort(X2),slope_X1[order(X2)],type='l',ylim=c(-
1,4),ylab='Estimated coefficients (X2)',xlab='X2')
> lines(sort(X2),((predX2+1.96*SEX2)/X1)[order(X2)],lty=2)
> lines(sort(X2),((predX2-1.96*SEX2)/X1)[order(X2)],lty=2)
> points(X2,slope_X1+res)
> lines(sort(X2),3+cos(2*pi*X2)[order(X2)],col='red')
> legend(0.03,0.5,legend=c("95% CI",'Estimated
coefficients','True coefficients','partial
residuals'),lty=c(2,1,1,NA),pch=c(NA,NA,NA,1),col=c('black'
,'black','red','black'))
```

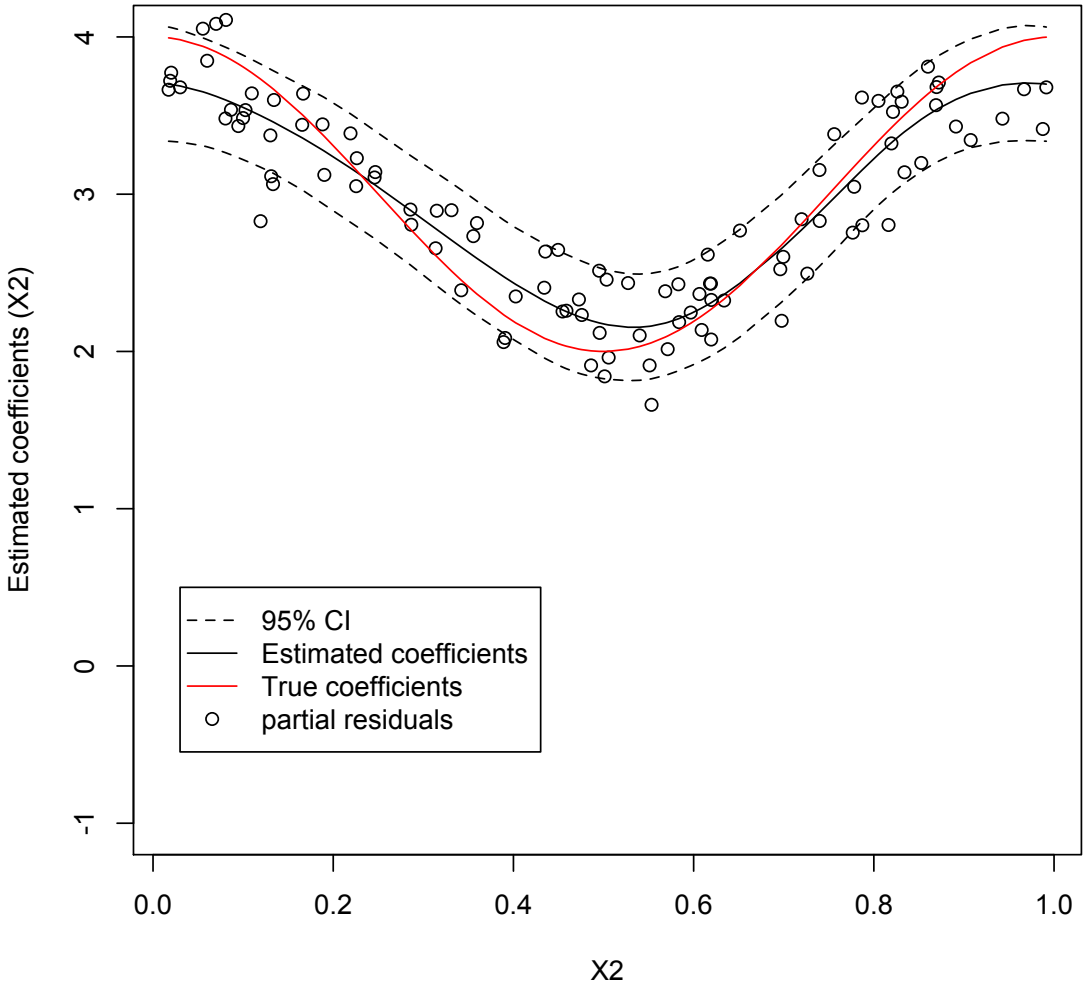


Figure 5. Estimated regression coefficients (b_i), actual regression coefficients, partial residuals, and relative 95% confidence intervals.

Chapter 12: Modeling spatial data: multidimensional smoothing, thin plate regression splines and tensor product

In many cases it is necessary to model data as a function of a multidimensional variable. For example, when modelling fish distribution it is expected to have an effect of position, defined by a set of two variables: latitude and longitude. In such a case, it is possible to use bidimensional smoother (e.g., `s(lat, lon)`), so that the result is a smooth surface rather than a line. Multidimensional smoothing is ideal to model data rich scenarios (as it is typically the case in spatial ecology) in which the covariate are expected to interact.

The multidimensional smoothing problem is an extension of the unidimensional case. The coefficients of a multidimensional smoother are also derived by minimizing the roughness penalty function (Eq. 5). In two-dimensional cases it has been proved that the *thin plate regression spline* is the solution of the penalty function, just like the cubic regression spline is the solution in the unidimensional case. Specifics on how thin plate splines are applied within the GAM framework are presented in Wood (2003, 2006) and further illustrated by Chan (MS). Here, we present the use of thin plate spline applied to study the distribution of adult stages of flatfish.

Arrowtooth flounder spatial distribution (*Atherestes stomias*) in the eastern Bering Sea

Arrowtooth flounder is a large flatfish species found both in the Bering Sea and in the Gulf of Alaska. ATF is not a commercial species (flesh is mushy and not tasty). However, the adults are voracious predators of other commercially important fish, like juvenile pollock and cod. So, understanding the sources of variability of ATF interannaul changes in distribution is important for the study of overlap and predation with its prey.

Objectives.
The objective of this preliminary analysis is that of determining whether the distribution of ATF changes according to their population biomass. If so, then the species is expected to increase their distribution range during years of high average abundance so that intraspecific competition is reduced in highly crowded areas. We are also interested in understanding how co-located variables (i.e., water temperature and

sediment characteristics) affect on the distribution of ATF. We consider the following covariate: *latitude, longitude, bottom temperature* (surface is supposedly less important as these fish are located toward the bottom*),* and *bottom type* (grain size). Results of this analysis have been published in Ciannelli et al. (2012).

Data

We analyze data form the groundfish survey of the eastern Bering Sea from 1982 to 2010. This is a bottom trawl survey, on a stratified grid. The survey grid is regular. The catch is standardized by area swept (cpue) and is further subdivided by sex and length. It is possible that fish of different length/sex have different distributions. Here we only examine the distribution of fish => 35 cm in total length. Sex are combined.

Results

Before diving in the statistical modeling it is important to develop our own perception of the data by doing few exploratory analyses. A look at the time series of ATF population biomass (from stock assessment) indicates a positive trend over time (Fig. 1). An examination of the occupancy (i.e., number of consistently sampled stations which are occupied by ATF) over time indicates that there is an expansion of habitat during years of high biomass (Fig. 1). Bottom temperature may also have something to do with it but we do not know yet. We will test these hypotheses using the various gam formulations learned so far.

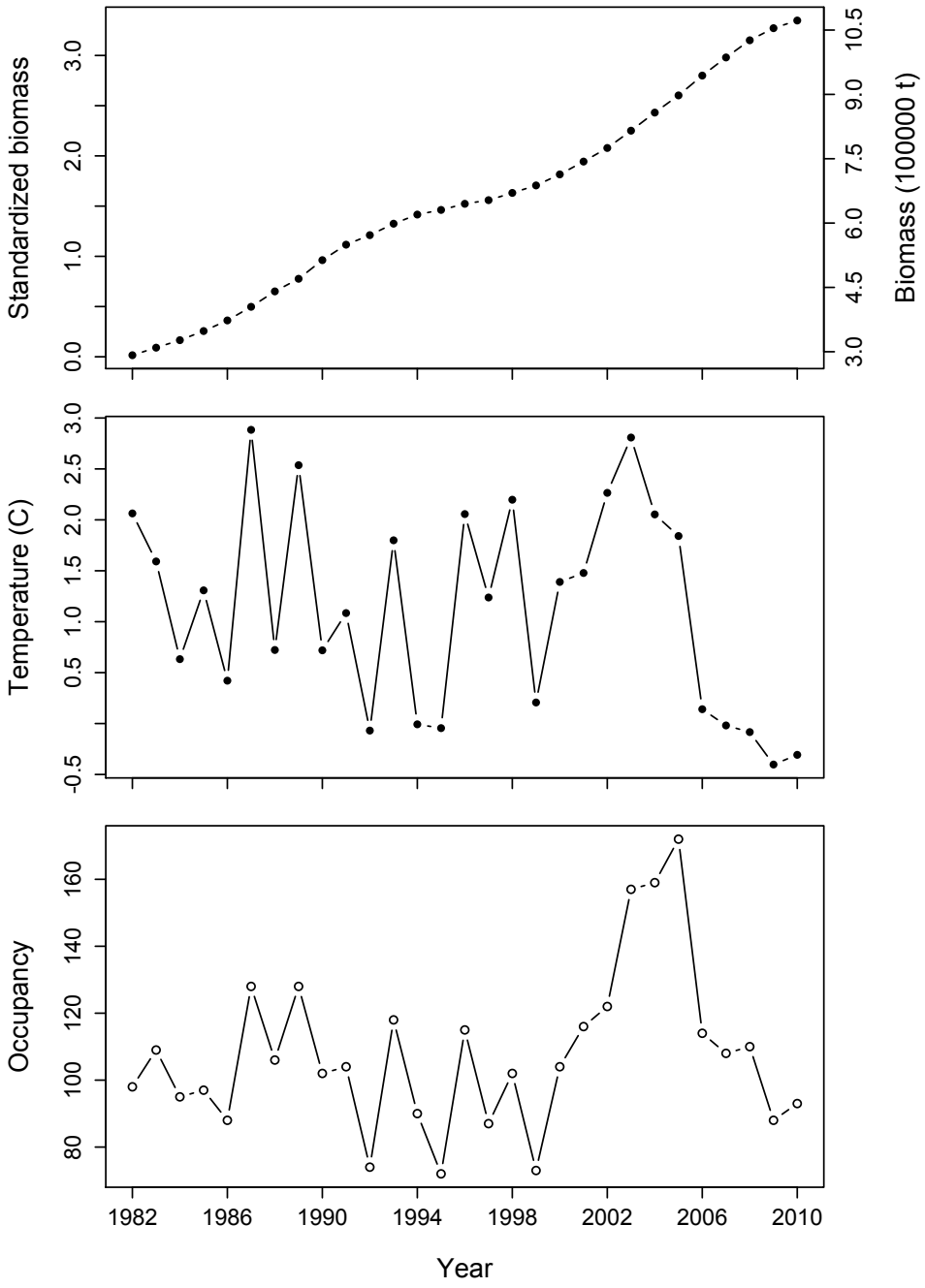


Fig. 1. Top panel: population biomass of arrowtooth flounder in the eastern Bering Sea. Middle panel: average bottom temperature of the middle shelf. Lower panel: ATF occupancy, that is number of consistently sampled stations, which are occupied by ATF.

Looking at the catch per unit effort of ATF over contrasting biomass and temperature years we get a better picture of how their distribution is influenced by bottom temperature and their own biomass (Fig. 2). In particular we see that there may be a potential interaction between temperature and population biomass on ATF occupancy, so that when biomass is low occupancy is also low, regardless of temperature.

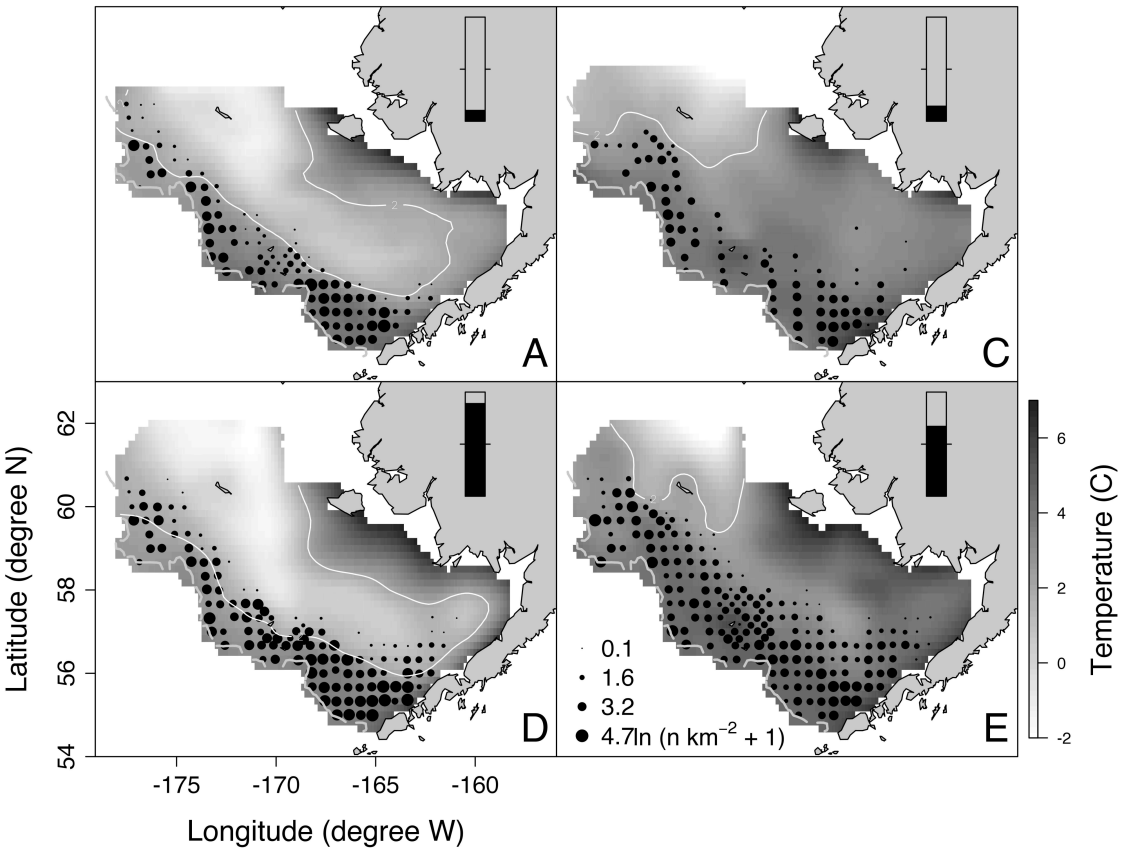


Figure 2. Catch per unit effort (bubbles) of ATF during contrasting population biomass and temperature years. The ATF population biomass is indicated on a relative scale from 0 (min) to 1 (max) by the upper right corner thermometer. The bottom temperature is shown as a color image[7].

We now proceed with the statistical analysis of the data by comparing results from four different model formulations:

1) a fully additive model (GAM), representative of a scenario with no changes of processes affecting the distribution of ATF over time. Ecologically, this formulation is representative of a scenario in which changes of ATF catches in relation to the overall abundance of ATF and of the local temperature are homogenous over space and time.

[7] The code used to make all of these figures is in the accompanying R script

2) a univariate threshold model with average biomass as the threshold variable (TGAM), representative of a scenario with abrupt changes of ATF distribution in relation to population biomass
3) a tensor product between position and average annual biomass, representative of a more gradual change of ATF distribution in relation to population size
4) a variable coefficient formulation with spatially variable effects of temperature and population biomass. This formulation is representative of a scenario in which ATF catches in relation to temperature and overall biomass change homogenously over time (i.e., no time threshold) but variably over space. It is further assumed that locally temperature and biomass have a linear effect on ATF catches (recall assumptions of variable coefficient GAM from previous chapter).

In all scenarios, the zeros are removed from the analysis. While this operation may appear statistically sinful, practically it is not, because the NOAA bottom trawl survey, where the data we analyze come from, was not designed with the intent of only catching ATF, and thus covers areas of the Bering Sea (most notably the inner shelf) wher ATF is very unlikely to occur. Models are compared against each other using AIC.

1) *Additive model*

```
> gam1 <-
gam(log(cpuelt+1)~avg.wt+bt+s(lon,lat)+s(depth,k=5)+s(phi,k=5),data=sub
data)
> summary(gam1)

Family: gaussian
Link function: identity

Formula:
log(cpuelt + 1) ~ avg.wt + bt + s(lon, lat) + s(depth, k = 5) +
    s(phi, k = 5)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.41341    0.07947   5.202  2.1e-07 ***
avg.wt       0.31071    0.01418  21.916  < 2e-16 ***
bt           0.33314    0.01783  18.681  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
              edf Ref.df      F  p-value
s(lon,lat) 26.513  28.59 20.944  < 2e-16 ***
s(depth)    3.816   3.97 36.609  < 2e-16 ***
s(phi)      3.803   3.97  6.928 1.66e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.535   Deviance explained =   54%
```

```
GCV score = 0.62159  Scale est. = 0.61454    n = 3272

> AIC(gam1)
[1] 7734.298
```

Note that the effects of local bottom temperature and of population biomass are included linearly in this model to make it comparable to the variable coefficient gam (which also considers locally linear effects). All the variables included in the model have a significant contribution toward the spatio-temporal variability of ATF CPUE. We now look at the additive effects of the covariate (Fig. 5).

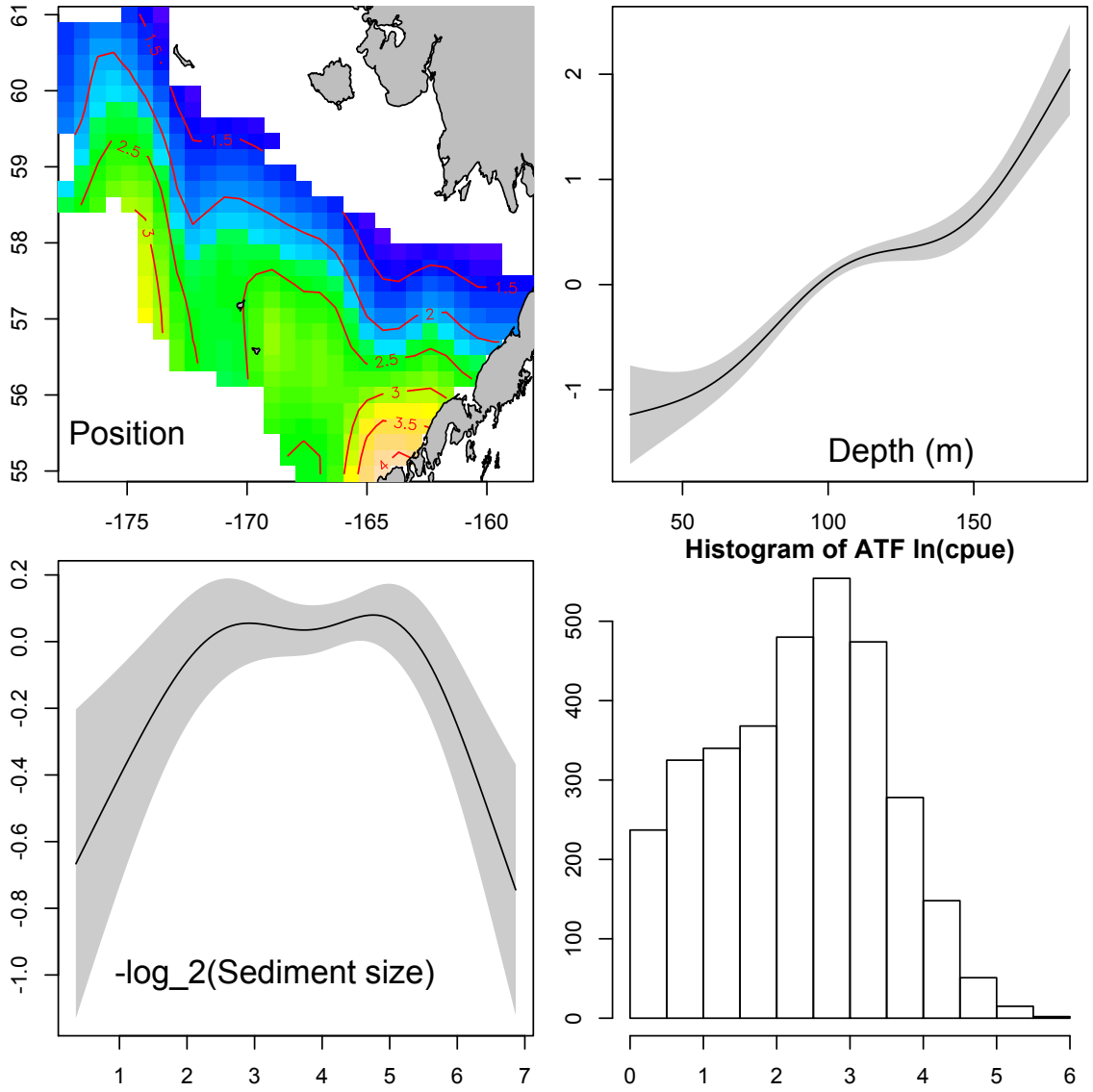


Figure 3. Model predictions of ATF distribution and covariate effects from fully additive model formulation (gam1).

2) *Threshold model*
Recall that this formulation assumes an abrupt shift of ATF distribution as the population biomass crosses a threshold value (th) to be estimated from the data. Following a slightly modified syntax from that used for the univariate threshold models (see chapter 9) we write our own loop to estimate the threshold of ATF biomass from a restricted search:

```
> biom<-sort(unique(subdata$avg.wt))[5:25]
> aic.bio<-biom*NA
> for(i in 1:length(biom)){
+ gam.obj <- gam(log(cpuelt+1)~avg.wt+bt+
+s(lon,lat,by=I(1*(avg.wt<=biom[i])))+
+s(lon,lat,by=I(1*(avg.wt>biom[i])))+
+s(depth,k=5)+s(phi,k=5),data=subdata)
+ aic.bio[i]<-gam.obj$aic}
```

Note that the search for the threshold was applied to the middle 21 observations, which excludes the first and last 4 years. This ensures that even if the threshold is estimated at the lowest or highest extreme of the restricted vector of ATF biomass, we still have few years left to actually define a regime. We next plot the value of the AIC for each corresponding threshold model:

```
> th<-biom[order(aic.bio)[1]]
> subdata$th<-biom[order(aic.bio)[1]]
> par(mfrow=c(2,1),omi=c(0,1,0,1))
> plot(biom,aic.bio,type='b',xlab='Standardized biomass',ylab='AIC')
> abline(v=th,lty=2)
> text(0.7,7500,'Low')
> text(2.5,7500,'High')
> plot(years[5:25],biom,type='b',ylab='Standardized
biomass',xlab='Years')
> abline(h=th);abline(v=1995,lty=2)
> text(1990,2.7,'Before')
> text(1999,2.7,'After')
```

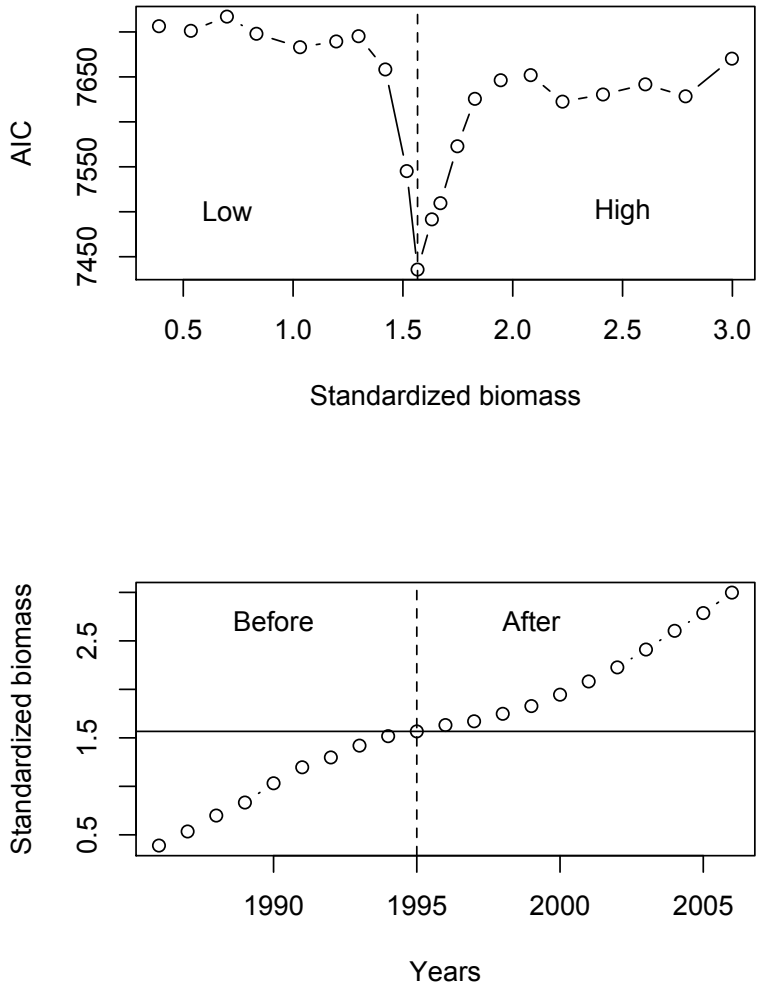


Fig. 4. AIC profile for each threshold GAM. The lowest AIC defines the location of the threshold and divide the data set in a low and high ATF biomass regime, or (because increase in monotone through time) a 'before' and an 'after' regime.

Following the same procedure used in Chapter 9, now that we know what the value of the threshold is, we can include it in a gam formulation, which is then used to make ecological inference.

```
> #The real GAMM
> subdata <- na.omit(atf.350)
> subdata$bt<-(subdata$bt-mean(subdata$bt))/sd(subdata$bt)
> subdata$bt<-subdata$bt+1.01*abs(min(subdata$bt))
> subdata$avg.wt<-(subdata$avg.wt-
mean(subdata$avg.wt))/sd(subdata$avg.wt)
> subdata$avg.wt<-subdata$avg.wt+1.01*abs(min(subdata$avg.wt))
> subdata$th<-th
```

```
> gam.2<- gam(log(cpuelt+1)~avg.wt+bt+s(lon,lat,by=I(1*(avg.wt<=th)))+
+ s(lon,lat,by=I(1*(avg.wt>th)))+s(depth,k=5)+s(phi,k=5),data=subdata)
```

The AIC of the 'real GAM' should be the same value of the lowest AIC in Fig. 4, and in fact it is.

```
>  gam.2$aic
[1] 7435.738
```

```
> summary(gam.2)

Family: gaussian
Link function: identity

Formula:
log(cpuelt + 1) ~ avg.wt + bt + s(lon, lat, by = I(1 * (avg.wt <=
    th))) + s(lon, lat, by = I(1 * (avg.wt > th))) + s(depth,
    k = 5) + s(phi, k = 5)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.75047    1.00816  -0.744    0.457
avg.wt       0.64802    0.02410  26.892   <2e-16 ***
bt           0.43550    0.01814  24.008   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
                                    edf Ref.df      F  p-value
s(lon,lat):I(1 * (avg.wt <= th)) 24.180 27.736  9.857  < 2e-16 ***
s(lon,lat):I(1 * (avg.wt > th))  27.072 29.212 14.440  < 2e-16 ***
s(depth)                          3.643  3.908 41.044  < 2e-16 ***
s(phi)                            3.809  3.967  7.756 3.69e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.579   Deviance explained = 58.7%
GCV score = 0.56719  Scale est. = 0.55655   n = 3272
```

The TGAM formulation represents an improvement compared to the GAM, in term of both AIC and R-sq. However, these two models presented are not directly comparable, as the latter one contains an extra parameter (the threshold) hard to account for in the AIC. In order to compare the two we would have to estimate the genuine cross validation (CV). However, using the `threshold.gam` .cv function would be very time consuming. One way to get around this problem is to randomly exclude a larger subset of stations from the model calibration (say 400), which can then be used to estimate the prediction error. The operation can be repeated several hundred times (say 500) in order to get an average prediction error, which indeed is the CV score. Examples of how this routine is implemented on large data set can be found in Ciannelli et al. (2007a,b, 2012). It is also worth noting that tgam has no significant intercept term. While initially bewildering, this result has some sense,

because the average CPUE for each regime is now absorbed by the regime-specific position effects. Recall that the smooth of a variable coefficient gam is no longer centered around zero when it is indentified using a numerical rather than a factor variable, as we have done here (cf. formulations in chapter 9 and 10). Let us take a look at effect of the terms included in the tgam model.

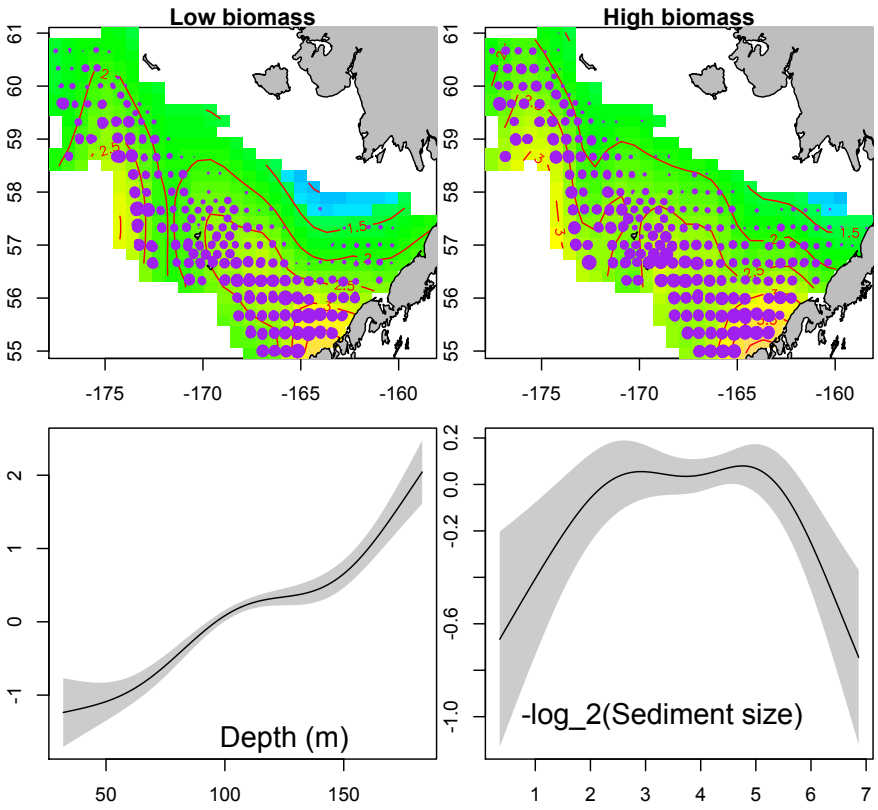


Figure 5. The top two panels show the effect of position on ATF cpue overlaid with the observations of ATF cpue for each regime. The lower two panels show the covariate effects. The codes for this and other figures are in the accompanying R script.

3) *Tensor product formulation*

We now move on to the tensor product formulation. For a thorough description of tensor product refer to Wood 2006 (pag. 162). Briefly, the tensor product allows for application of different penalty functions over different covariate values. In classical multidimensional smoothing (e.g., thin plate regression splines) the penalty function is isotropic, that is equal in all covariate directions. Isotropic penalty functions are ideal when dealing with set of covariates which have similar units (e.g., lat and lon). However, when dealing with covariates with different units the isotropic penalty function is no longer desirable. Tensor products may instead be more effective. In our specific case we will use a tensor product between a two-dimensional isotropic (lat and lon) and unidimensional (average ATF biomass) smoother. With this formulation

we assume a more gradual change of distribution in relation to average population size. Note the syntax of the tensor product (`te`). Now the arguments for the smoothing function include three variables (`lon, lat` and `avg.wt`), two of which are modeled with a thin plate regression spline (`tp`) and the other with a cubic regression spline (`cr`).

```
> gam.3<-gam(log(cpuelt+1)~avg.wt+bt+
+ te(lat,lon,avg.wt,k=c(30,5),d=c(2,1),bs=c("tp","cr"))+
+ +s(depth,k=5)+s(phi,k=5),data=subdata)
> summary(gam.3)
Family: gaussian
Link function: identity

Formula:
log(cpuelt + 1) ~ avg.wt + bt + te(lat, lon, avg.wt, k = c(30,
    5), d = c(2, 1), bs = c("tp", "cr")) + +s(depth, k = 5) +
    s(phi, k = 5)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.28954    1.45230  -0.199    0.842
avg.wt       0.50693    0.75968   0.667    0.505
bt           0.41775    0.02029  20.585   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
                       edf  Ref.df      F  p-value
te(lat,lon,avg.wt) 108.892 127.451 10.331  < 2e-16 ***
s(depth)             3.755   3.950 42.818  < 2e-16 ***
s(phi)               3.755   3.953  8.604 8.27e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.608   Deviance explained = 62.2%
GCV score = 0.53796  Scale est. = 0.51839   n = 3272

> AIC(gam.3)
[1] 7258.248
```

The summary is interesting. It indicates that the ATF population biomass no longer has a significant (linear) effect on ATF cpue. The overall model intercept is not significant, while bottom temperature still has a positive and significant effect on ATF cpue. The fact that the linear coefficient associated with the ATF population biomass is no longer significant is not very surprising, because with the biomass term also included in the tensor product there is a greater flexibility to locally adjust for changes of ATF biomass. The AIC indicates that this is (by far) the best model formulation so far. Recall however, that AIC may not properly account for the number of parameters of complex smooth terms and threshold effect. The ultimate test rests on the prediction capability of the model, which is assessed with the genuine CV (not shown here). Plotting model predictions of ATF cpue over four regimes of ATF biomass allows us to

visually assess the effect of increasing population biomass on the spatial distribution of ATF. The four increasing values of average population biomass specified in the argument `cond` within the `vis.gam` function and are conveniently chosen so that two are below and two above the threshold biomass value estimated from the second model formulation (Fig. 6).

```
> quartz()
> par(mfrow=c(2,2),omi=c(0.5,0.5,0.5,0.5),mai=c(0.35,0.5,0.5,0.05))
> vis.gam(gam.3,view=c('lon','lat'),cond=list(avg.wt=0.6),
+plot.type="contour",color="topo",too.far=.05,
+zlim=range(predict(gam.2)),main='B = 0.6')
> map("worldHires",fill=T,col="grey",add=T)
> #just change avg.wt for the other 3
```

Note the expansion of ATF habitat with increasing of average biomass. Colors may be misleading. It is better to pick a contour line (e.g., 2) and follow it through the four images.

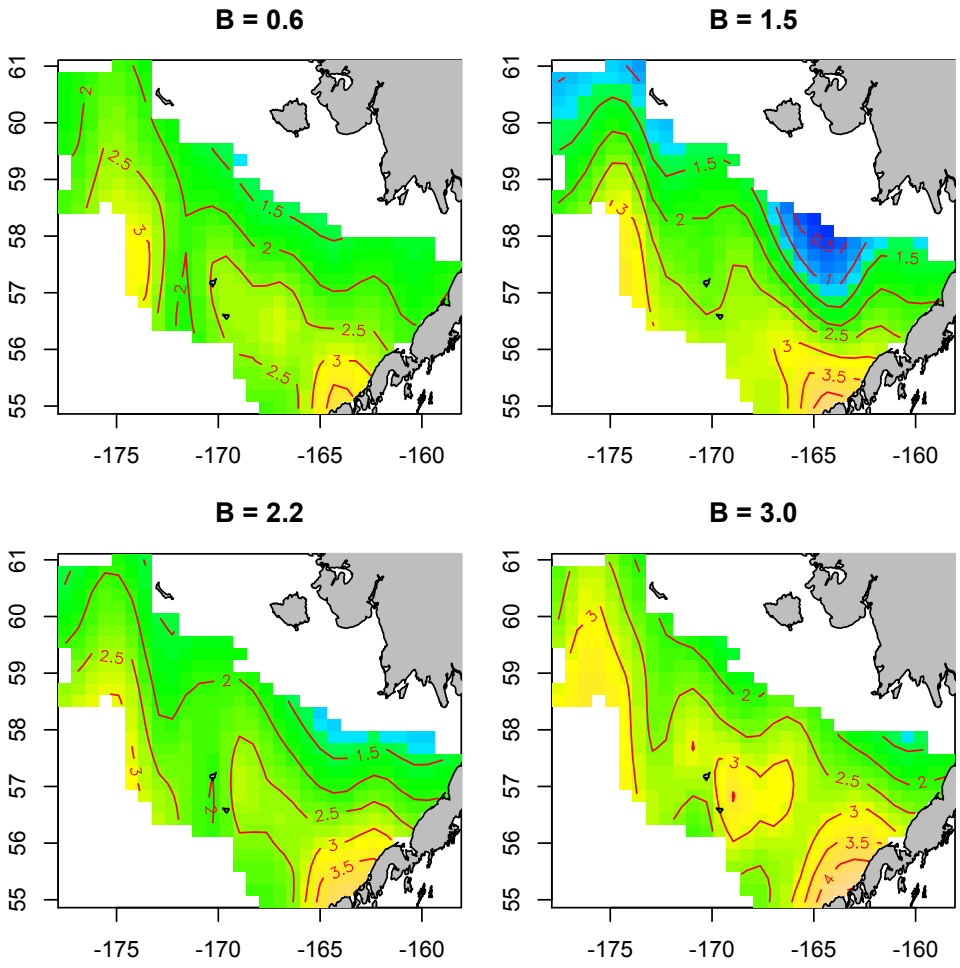


Figure 6. Model predictions of the ATF distribution from the tensor product model formulation over four increasing scenarios of average biomass (B). Note that he top two panels have B < than the threshold value estimated from the tgam.

4) Variable coefficient formulation

In this model formulation we assume that bottom temperature and ATF population biomass have spatially variable effects on local abundance of ATF. We furthermore assume that these effects are locally linear, but the rate of change (i.e., slope coefficients) of these effects can nonlinearly and smoothly change over space. In essence:

```
> gam.4 <- gam(log(cpuelt+1)~s(lon,lat)+s(lon,lat,by=avg.wt)+
s(lon,lat,by=bt)+s(depth,k=5)+s(phi,k=5),data=subdata)
> summary(gam.4)

Family: gaussian
Link function: identity
```

```
Formula:
log(cpuelt + 1) ~ s(lon, lat) + s(lon, lat, by = avg.wt) + s(lon,
    lat, by = bt) + s(depth, k = 5) + s(phi, k = 5)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.79660    0.09185   8.673   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
                   edf Ref.df      F  p-value
s(lon,lat)        26.293 27.739  7.356  < 2e-16 ***
s(lon,lat):avg.wt 23.713 27.546 20.563  < 2e-16 ***
s(lon,lat):bt     24.149 27.436 19.372  < 2e-16 ***
s(depth)           3.826  3.980 35.955  < 2e-16 ***
s(phi)             3.699  3.949  6.252 6.03e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.566   Deviance explained = 57.7%
GCV score = 0.58863  Scale est. = 0.57376   n = 3272

> AIC(gam.4)  #7551.393
[1] 7553.222
```

We got a significant and spatially variable effect of bottom temperature (`s(lon,lat):bt`) and population biomass (`s(lon,lat):avg.wt`) on local ATF CPUE. The model fit is slightly better than the fully additive formulation but a bit worse compared to the threshold and tensor product formulations. The tricky part now is to get and plot on a map the slope coefficients for BT and avg.B. This has already been shown in the previous chapter for the fictitious data set and will be shown for this case study in the code below. In the process of doing so we will also determine whether the slopes are significantly different from zero, based on the 95% confidence interval of each slope estimate, in turn calculated from the standard error (95%CI(μ) = μ ±1.96*se). When getting the slopes from the gam object, remember that the term predictions are already multiplied by the covariate (in our case BT or avg.B) and therefore, if we are only interested in getting the slopes and determining their significance, we need to divide the term predictions by their respective covariate. Likewise, the standard error of each term prediction also refers to the product between the slope and covariate. Therefore, to determine the 95% CI of the slope, we first need to derive the upper and lower boundaries of the term and then divide them by the covariate. In essence:

```
> #Get slope coefficients for average biomass and determine
significance (tricky!)
> pred<-predict(gam.4,type='terms',se.fit=T)
> pred.slope.b<-pred[[1]][,2]/subdata$avg.wt
> pred.slope.se.b<-1.96*pred[[2]][,2]
```

```
> pred.slope.up.b<-(pred[[1]][,2]+pred.slope.se.b)/subdata$avg.wt
> pred.slope.low.b<-(pred[[1]][,2]-pred.slope.se.b)/subdata$avg.wt
> sign.slope.pos.b<-(1:length(pred.slope.b))[pred.slope.low.b>0]
> sign.slope.neg.b<-(1:length(pred.slope.b))[pred.slope.up.b<0]
```

We now plot these slopes as histograms, to get a feeling of their range in values and variation (Fig. 7):

```
> par(mfrow=c(2,2))
> hist(pred.slope.b, main='All slopes, biomass')
> hist(pred.slope.se.b,main='Standard error, biomass')
> hist(pred.slope.b[sign.slope.pos.b], main='Significantly positive,
biomass')
> hist(pred.slope.b[sign.slope.neg.b], main='Significantly negative,
biomass')
Error in hist.default(pred.slope.b[sign.slope.neg.b], main =
"Significantly negative, biomass") :
  invalid number of 'breaks'
> max.b<-max(abs(pred.slope.b))
```

Note that there are no significant negative slope values for avg.wt (i.e., population biomass only has positive effects on local ATF), therefore we get an error when trying to plot them.

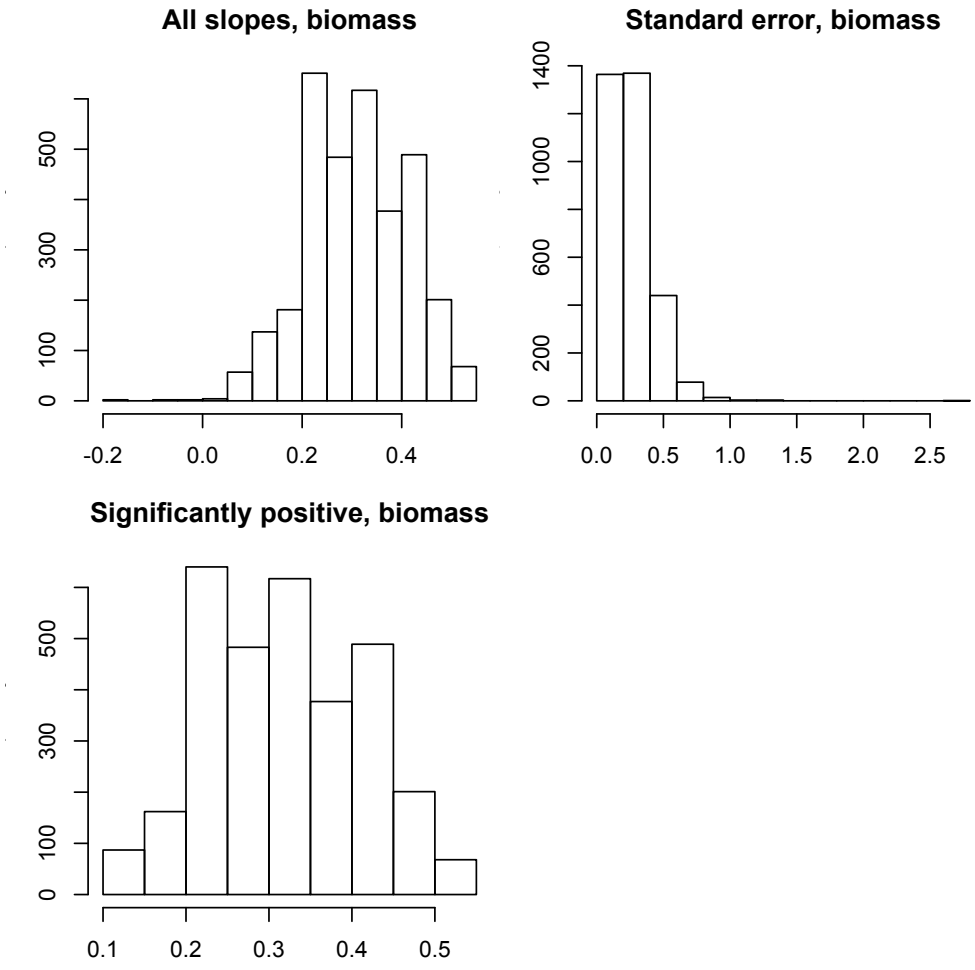


Figure 7. Histogram for the slope values determining the local rate of change of ATF CPUE in relation to ATF population biomass. The top two plots show all slopes, while the bottom two show only those that are significantly different from zero. Note absence of negative slopes.

The same steps are repeated to get the slope associated to the effect of bottom temperature on local ATF CPUE from the GAM object. Results are shown in Fig. 8, note that this time we end up with few significant negative slopes, but most of them are still positive and greater than those associated to biomass.

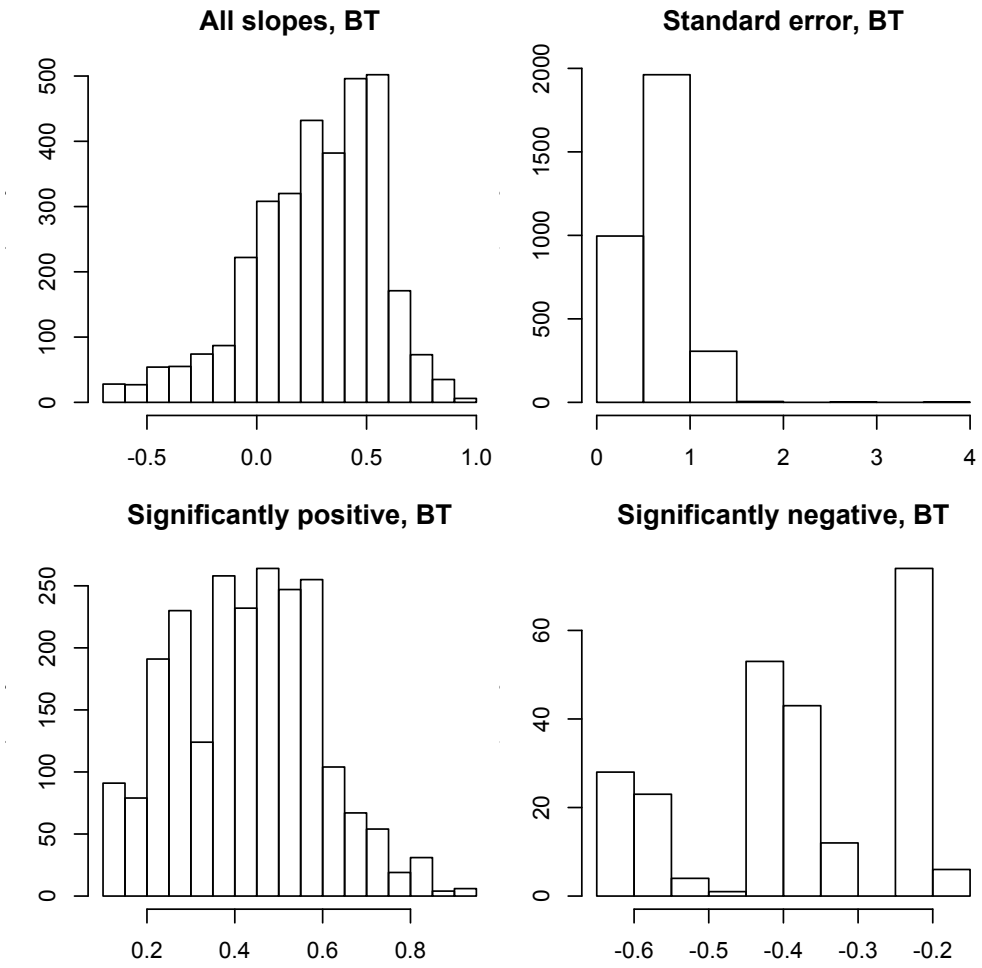


Figure 8. Histogram for the slope values determining the local rate of change of ATF CPUE in relation to bottom temperature. The top two plots show all slopes, while the bottom two show only those that are significantly different from zero. Note the presence of few significant negative slopes.

Now, some tricky code to show these bubbles on a map (Fig. 9).

```
> par(mfrow=c(2,2),omi=c(0.5,0.5,0.5,0.5),mai=c(0.35,0.5,0.5,0.05))
> #Plot average ATF distribution and local effect of avg.B
>
vis.gam(gam.4,view=c('lon','lat'),plot.type="contour",color="gray",too.
far=.04,main='',ylab='')
```

```
> symbols(subdata$lon[sign.slope.pos.b],subdata$lat[sign.slope.pos.b],
+
circle=pred.slope.b[sign.slope.pos.b],inches=0.03*max(pred.slope.b[sign
.slope.pos.b])/max(max.b,max.t),add=T,fg='white',bg='white')
> map("worldHires",fill=T,col="grey",add=T)
> text(-159,60.7,labels='A',cex=1.5)
> mtext('Longitude (degrees west)',1,line=2.5)
> mtext('Latitude (degrees north)',2,line=2.5)
> #Plot average ATF distribution and local effect of BT
> #Plot average ATF distribution and local effect of BT
>
vis.gam(gam.4,view=c('lon','lat'),plot.type="contour",color="gray",too.
far=.04,main='',ylab='')
> symbols(subdata$lon[sign.slope.pos.t],subdata$lat[sign.slope.pos.t],
+
circle=pred.slope.t[sign.slope.pos.t],inches=0.03*max(range(pred.slope.
t[sign.slope.pos.t],finite=T))/max(max.b,max.t),add=T,bg='white',fg='wh
ite')
> symbols(subdata$lon[sign.slope.neg.t],subdata$lat[sign.slope.neg.t],
+
circle=abs(pred.slope.t[sign.slope.neg.t]),inches=0.03*max(range(pred.s
lope.t[sign.slope.pos.t],finite=T))/max(max.b,max.t),add=T,bg='blue',fg
='blue')
> map("worldHires",fill=T,col="grey",add=T)
> text(-159,60.7,labels='B',cex=1.5)
> symbols(rep(-
176.7,4),seq(55.2,by=0.5,length=4),circles=seq(max(max.b,max.t),max(max
.b,max.t)/5,length=4),add=T,inches=0.04,bg='black')

> #Add bubble legend
> text(rep(-
175,6),seq(55.2,by=0.5,length=4),format(round(seq(max(max.b,max.t),max(
max.b,max.t)/5,length=4),1),digits=5,trim=T))
> text(-171.2,55.2,"ln([n/h]+1)")

> #Plot remaining model terms
> plot(gam.4,select=4,shade=T,scale=0,rug=F,ylab='')
> text(50,2.15,labels='C',cex=1.5)
> mtext('CPUE anomalies',2,line=2.5)
> mtext('Bottom depth (m)',1,line=2.5)
> plot(gam.4,select=5,shade=T,scale=0,rug=F,ylab='')
> text(6.5,0.175,labels='D',cex=1.5)
> mtext('Sediment size (phi)',1,line=2.5)
```

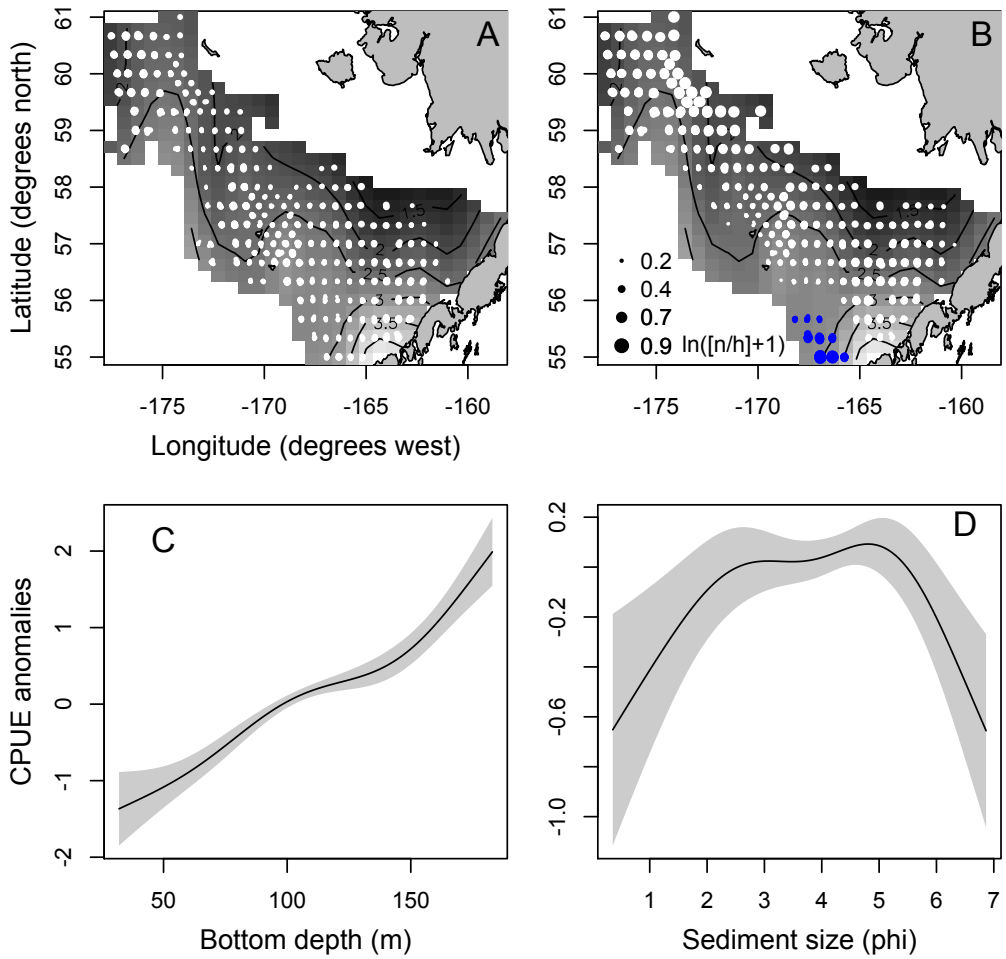


Figure 9. Top panels: rate of change (i.e., slope) of local ATF CPUE in relation to unit changes of ATF population biomass (left) and bottom temperature (right). Slopes are shown as bubbles and are overlaid on the image of average ATF cpue. Note that bottom temperature has some negative effects (blue bubbles) on local ATF CUPE. The bottom two panels show the additive effects of bottom depth and sediment size on ATF cpue. All estimated from model 4.

Conclusions

The ecological picture that has emerged from this analysis is that ATF distribution can change in relation to both population biomass and bottom temperature in a way that is potentially nonstationary (i.e., effects are not the same through time, see threshold and tensor product formulation) and spatially variable (i.e., variable coefficient gam). From a statistical perspective, based on the AIC, the tensor product formulation is the most aligned with the data. However, this should really be assessed with the calculation of the genuine cross validation, which is

not shown here, because the code takes too long to run. But the steps involved are very similar to the one shown for the cod eggs case study (Chapter 10). It is interesting and somewhat reassuring to see that the local positive effect of bottom temperature on local ATF CPUE are mostly concentrated around the coldest area of the Bering Sea middle shelf (i.e., the so called 'cold pool' of the Bering Sea shelf, cf. Figs. 9 and 2). This result in turn indicates that the low temperatures of this region limit ATF incursions on the middle shelf.

All of what was shown above was done ignoring the spatial autocorrelation to be expected in the data. We will try to address this and the overdispersion issue in the next and last chapter of this manual. For a more in depth look at the analysis of the ATF data see Ciannelli et al. (2012).

Chapter 13: Dealing with overdispersion and spatial autocorrelation

Overdispersion

Example 1: Pollock egg distribution in the Gulf of Alaska (data provided, see Ciannelli et al. 2007)
The data set `eggdata` contains walleye pollock egg catches (standardized by volume filtered and gear depth) collected across the western GOA in spring 1986. The goal here is to find out whether there are specific regions within the sampled area characterized by having a high density of pollock eggs (i.e., an hot spot), in turn indicative of an active pollock spawning ground. A problem common to spatial data (particularly for patchy organisms) is that of the zero inflated counts (i.e., many zeros). This makes the response variable highly skewed even when log transformed. The data set is said to be overdispersed.

```
> source(eggdata.R)
> hist(log(eggdata$catch+1), main='Egg density',density=50,
+col='grey',xlab='LN(density+1)')
>
```

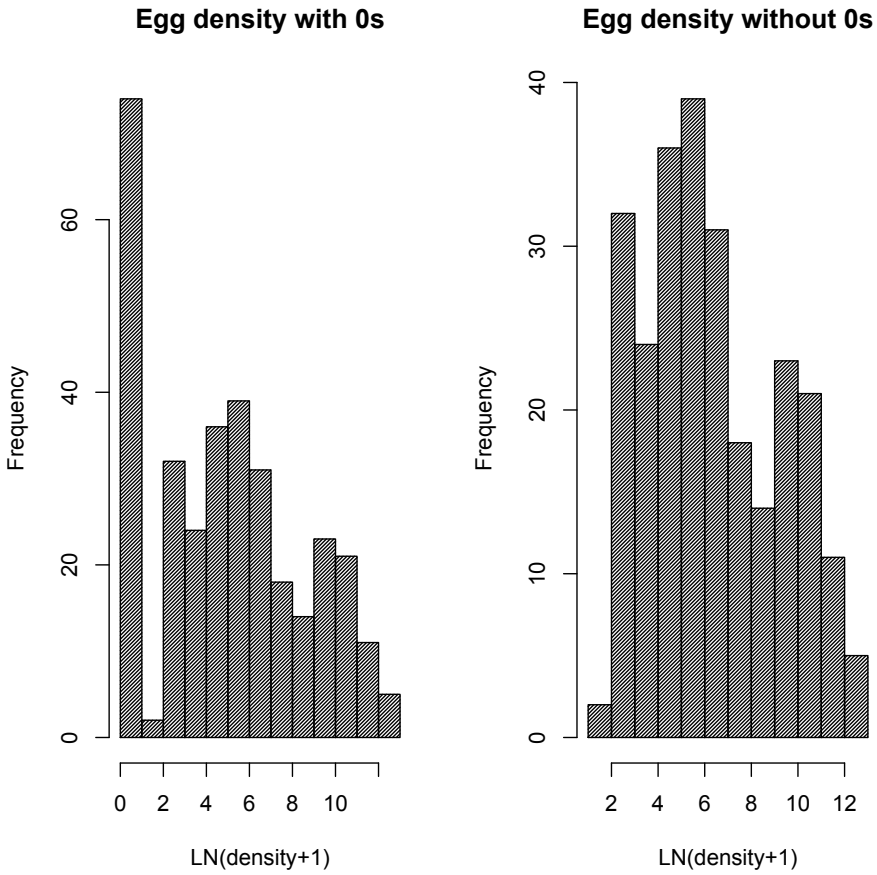


Fig. 1. Histogram of pollock egg densities with (left) and without (right) zero catches

One approach to deal with zero inflated counts is to eliminate the zero from the data used in the model fit. However, while this approach may correct the dispersion problem (Figure 1), it may also yield unrealistic results (Ciannelli et al. 2008). In fact we loose the valuable information of where eggs are not found. A second approach to model zero inflated counts is the one adopted by Fox et al. (2000), and summarized in the example below. Essentially we apply different family distributions to the data set, first modeling the presence/absence of eggs using binomial family and logit link function and then modeling the positive catches using an overdispersed distribution family. These are the steps to follow:

- Fit a model on presence/absence (1,0) data using the binomial distribution. Obtain the model predictions P_{ij} (where i and j specify the lat and lon), indicating the probability of finding eggs at a particular location
- Fit a model on egg density using the log-normal distribution (other families are also possible) and obtain the predictions D_{ij}.
- Finally, obtain the overall predictions (Y_{ij}) by multiplying the result of the previous two model fit: $Y_{ij} = P_{ij} * D_{ij}$.

The code below shows these steps and relative figures.

```
> #Remove 0 catches from data set and fit log-normal model
> subdata<-eggdata[eggdata$catch!=0,]
> dim(subdata)
[1] 256   6
> #note there were 74 zero catches, about 22% of original data
> egg1<-gam(log(catch+1)~s(lon,lat)+s(log(bottom)),data=subdata)
> summary(egg1)

Family: gaussian
Link function: identity

Formula:
log(catch + 1) ~ s(lon, lat) + s(log(bottom))

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.32843    0.09591   65.98   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
                 edf Ref.df      F  p-value
s(lon,lat)     20.983 25.353 16.758  < 2e-16 ***
s(log(bottom))  2.707  3.383  7.499 4.53e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.704   Deviance explained = 73.2%
```

```
GCV score = 2.6063  Scale est. = 2.3549    n = 256
```

Plot the covariate effects using the built in `vis.gam` function for the two-dimensional smoother of `lat` and `lon` (Fig. 2, see R script for figure code).

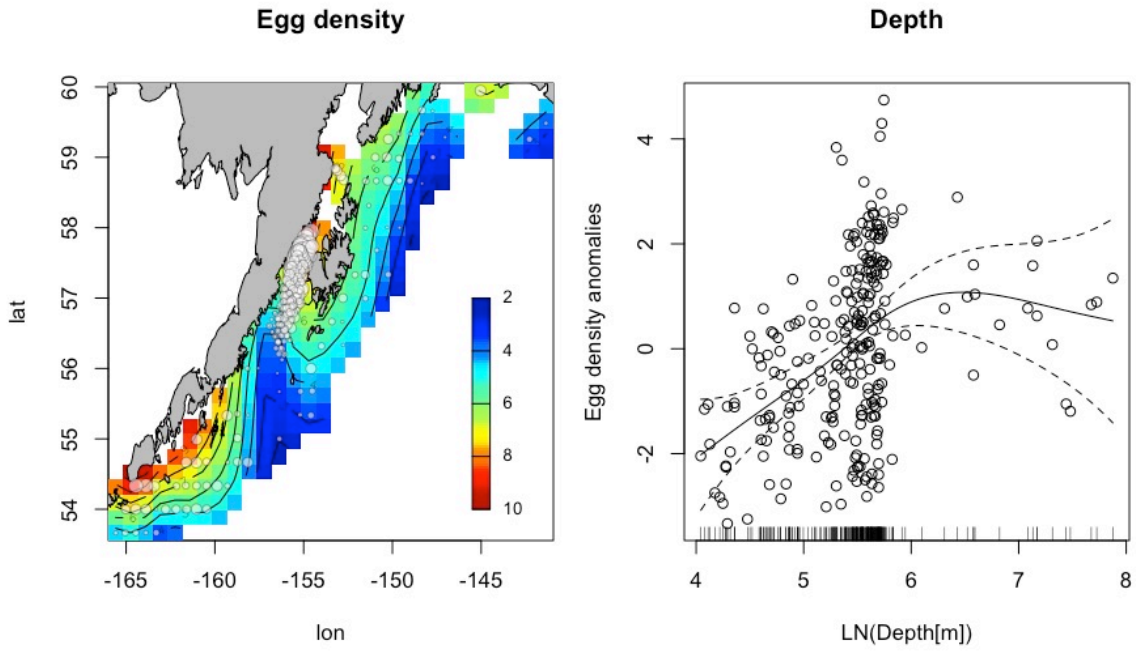


Figure 2. Pollock egg data. Covariate effect of the `egg1` model. The first plot shows the effect (on additive scale) of latitude and longitude modeled with thin plate splines. Overimposed are the actual observations. The second plot shows the effect of bottom depth, log-transformed.

Modeling the presence absence using binomial distribution:

```
> eggdata$pre<-1*(eggdata$catch>0)#vector of presence/absence
> egg1.p<-
gam(pre~s(lon,lat,k=20)+s(log(bottom),k=5),data=eggdata,family=binomial
)
> summary(egg1.p)

Family: binomial
Link function: logit

Formula:
pre ~ s(lon, lat, k = 20) + s(log(bottom), k = 5)

Parametric coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   2.7669     0.4044   6.843 7.78e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
Approximate significance of smooth terms:
                edf Ref.df Chi.sq  p-value
s(lon,lat)     9.301  11.83  57.41 6.35e-08 ***
s(log(bottom)) 2.744   3.26  19.45 0.000336 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =   0.57   Deviance explained = 58.1%
UBRE score = -0.47472  Scale est. = 1         n = 330
```

Figure 3 shows the model fit along with the observed density of pollock eggs (see scrip for figure code)

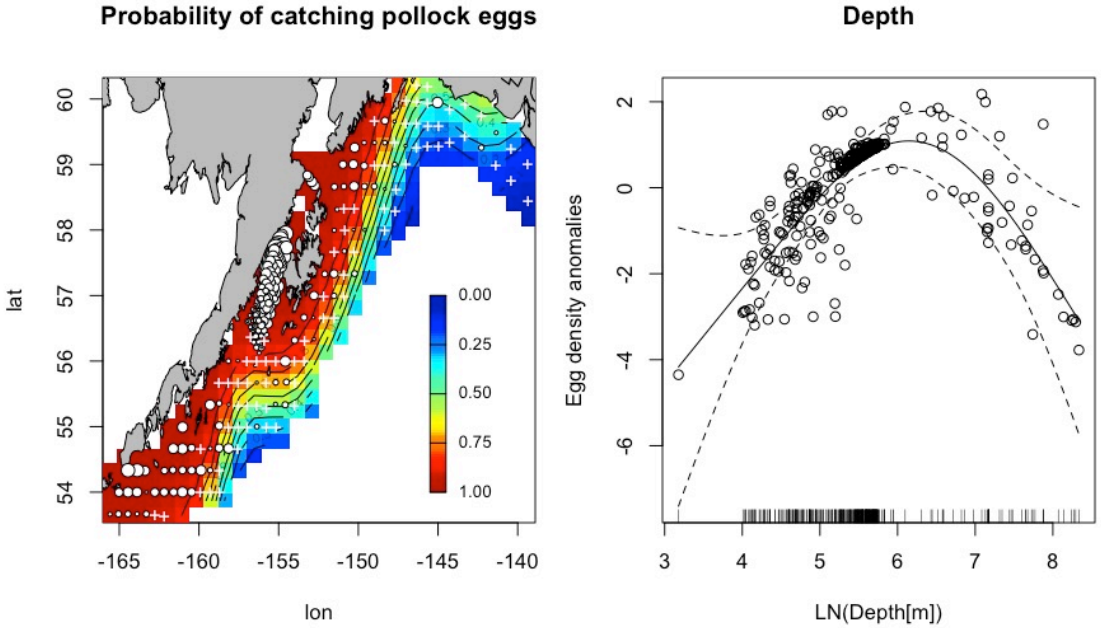


Figure 3. Pollock egg data. Covariate effect of the `egg1.p` model (presence/absence). The first plot shows the effect (on additive scale) of latitude and longitude modeled with thin plate splines. Overimposed are the actual observations, white crosses indicate zero cacthes. The second plot shows the effect of bottom depth, log-transformed.

We finally predict the pollock egg density by combining results from the two models. Results shown in Figure 4.

```
> #Predict egg cpue based on the results of the previous two models
> pred<-egg1.p$fitted*predict(egg1,newdata=eggdata)
.
```

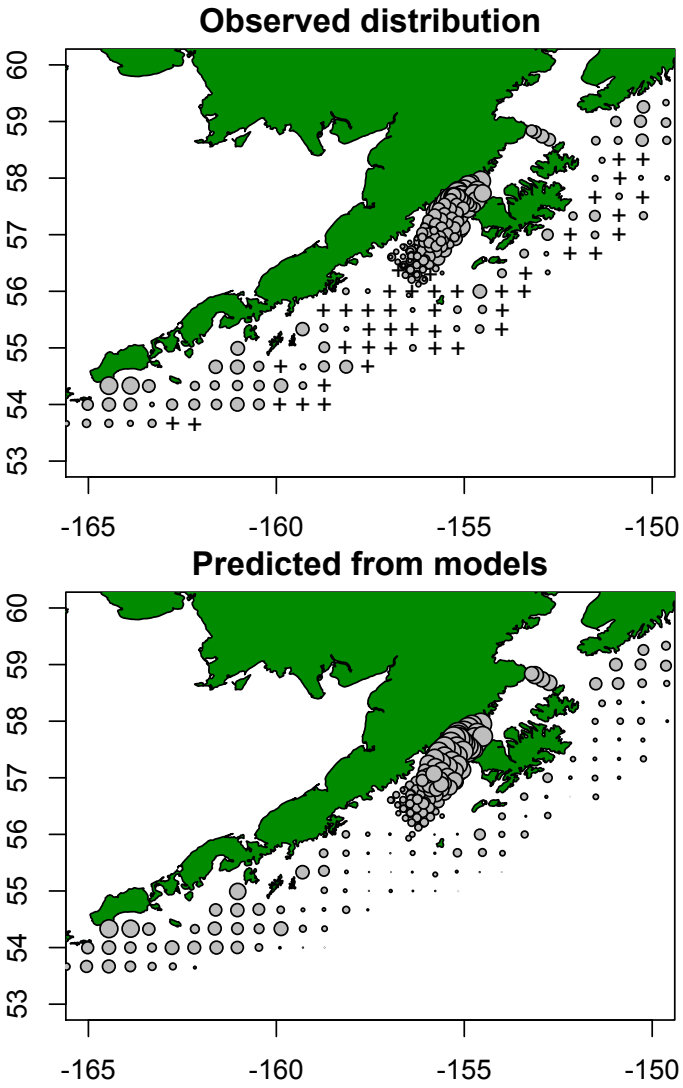


Figure 4. Observed (top panel) and predicted (bottom panel) density of pollock eggs. Predictions were obtained by combining the presence/absence and the presence only model.

Example 2: using overdispersed distribution family.
Atlantic cod eggs in Tvedestrand, Skagerrak Coast, Norway (data not provided, sorry, but see Ciannelli et al. 2010)

If the dispersion of the data is not too accentuated, it is possible to retain the zero and use and overdispersed distribution family to model the data. In this study we were interested in the horizontal and vertical distribution of cod eggs in Tvedestrand fjord, about 30 km north-east of Flodevigen. We were interested in the fine scale distribution of cod eggs throughout the entire spawning season and in relation to water circulations. Therefore the fjord was sampled multiple times from February to April (which includes the bulk of cod spawning phenology), while two ADCPs (Acoustic Doppler Current Profilers) permanently in the water during the same time period, provided data on water circulation. The fjord was sampled during two consecutive years, 2006 and 2007.

There were several objectives in this study, among which: 1) determine whether the inshore offshore gradient of eggs density observed for other fjord locations held also in Tvedestrand over two environmentally contrasting years; 2) determine whether the same inshore-offshore gradient holds at all sampled depths. Most of the cod eggs are located in the upper 15m and very rarely at depth > 30 m. Therefore in the analysis we considered egg density in two layers, shallower and deeper than 15m and did not consider catches below 30m. Our a-priori hypothesis was that the bulk of cod eggs were caught in the upper 15 m. These shallower eggs had an inshore-offshore distribution gradient (more eggs inshore) due to the effect of the dominant surface currents, which tend to flow up-fjord. Conversely, we did not expect an inshore-offshore gradient of egg density below 15m, as deeper currents flow out of the fjord. The eggs data are overdispersed and 36% of the observations were zero counts. Therefore, our hypotheses were tested with an overdispersed Poisson gam, which included depth (shallower and deeper than 15 m) and year (2006 and 2007) as interacting factors and distance from shore as a continuous smooth variable. Below we report the code used for this analysis. The data are not provided (sorry).

```
#Define depth and year levels in the egg data set
> egg$level1 <- 1*(egg$depthmax<=15 & egg$year==2006)
> egg$level2 <- 2*(egg$depthmax>15 & egg$year==2006)
> egg$level3 <- 3*(egg$depthmax<=15 & egg$year==2007)
> egg$level4 <- 4*(egg$depthmax>15 & egg$year==2007)
> egg$level<-egg$level1+egg$level2+egg$level3+egg$level4
> egg$outlier <- 1*(egg$allegg>30)#this adjusts for three outlier
observations
>

> gam.poisson <-
gam(allegg~outlier+factor(level)+offset(log(hauled))+s(dist,by=factor(l
evel))-1,data=egg,family=poisson,scale=-1,gamma=1.4)#gamma puts extra
penalty on roughness, see Wood book
```

There are several important items to note in the way the model was formulated. First, the scale of the model is left open (i.e., set `scale = -1`) to allow overdispersion. Typically in a Poisson model scale = 1, which means that variance and the mean are equal. If the scale is > 1 (i.e., variance > mean) then there is overdispersion, which is our case here. Second, we included an offset term. Poisson only takes integers. Numbers of eggs caught are indeed integers but they are not standardized by effort expanded to catch those eggs. So to standardize egg catches we include the natural log of meter hauled as an offset term. The coefficient of an offset term is not computed and implicitly assumed to be equal to 1. So these terms only contribute algebraically to the model fit. In our case meter hauled was log-transformed because the link of a Poisson distribution family is the natural log. Third, the model includes an interaction between depth level and year, both in the parametric (i.e., model intercept) and in the smooth terms (i.e., distance from shore). Fourth, the model overall intercept is not computed (`- 1` included at the end of the formula), so that the intercept values of each `depth x year` level express the mean number of eggs caught within each level.

```
> summary(gam.poisson)

Family: poisson
Link function: log

Formula:
allegg ~ outlier + factor(level) + offset(log(hauled)) + s(dist,
    by = factor(level)) - 1

Parametric coefficients:
               Estimate Std. Error t value Pr(>|t|)
outlier         3.12883    0.16637   18.81   <2e-16 ***
factor(level)1 -0.74268    0.05903  -12.58   <2e-16 ***
factor(level)2 -2.15198    0.11710  -18.38   <2e-16 ***
factor(level)3 -1.94170    0.12487  -15.55   <2e-16 ***
factor(level)4 -2.64549    0.16526  -16.01   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
                          edf Ref.df      F  p-value
s(dist):factor(level)1 4.420  5.444  9.757 2.17e-09 ***
s(dist):factor(level)2 1.005  1.009  3.774   0.0521 .
s(dist):factor(level)3 1.000  1.000 22.738 2.34e-06 ***
s(dist):factor(level)4 1.335  1.600  1.308   0.2545
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.676   Deviance explained = 49.9%
GCV score = 2.6527  Scale est. = 2.5484    n = 583
>
```

The model summary and figures of terms effects (Fig. 5) are also very interesting and confirm our original hypothesis. First, from the level intercepts we note that there were more eggs caught in 2006 (levels 1-2) than in 2007 (levels 3-4). Second, the smooth terms associated with distance from shore in the deeper layer (i.e., < 15m) is not significant, while it is highly significant in the shallower layers. In particular we see that there are more eggs caught inshore than offshore, but only above the 15 m (Fig. 5). Finally, the scale estimate for the model is > 1, confirming the overdispersed nature of the egg data set.

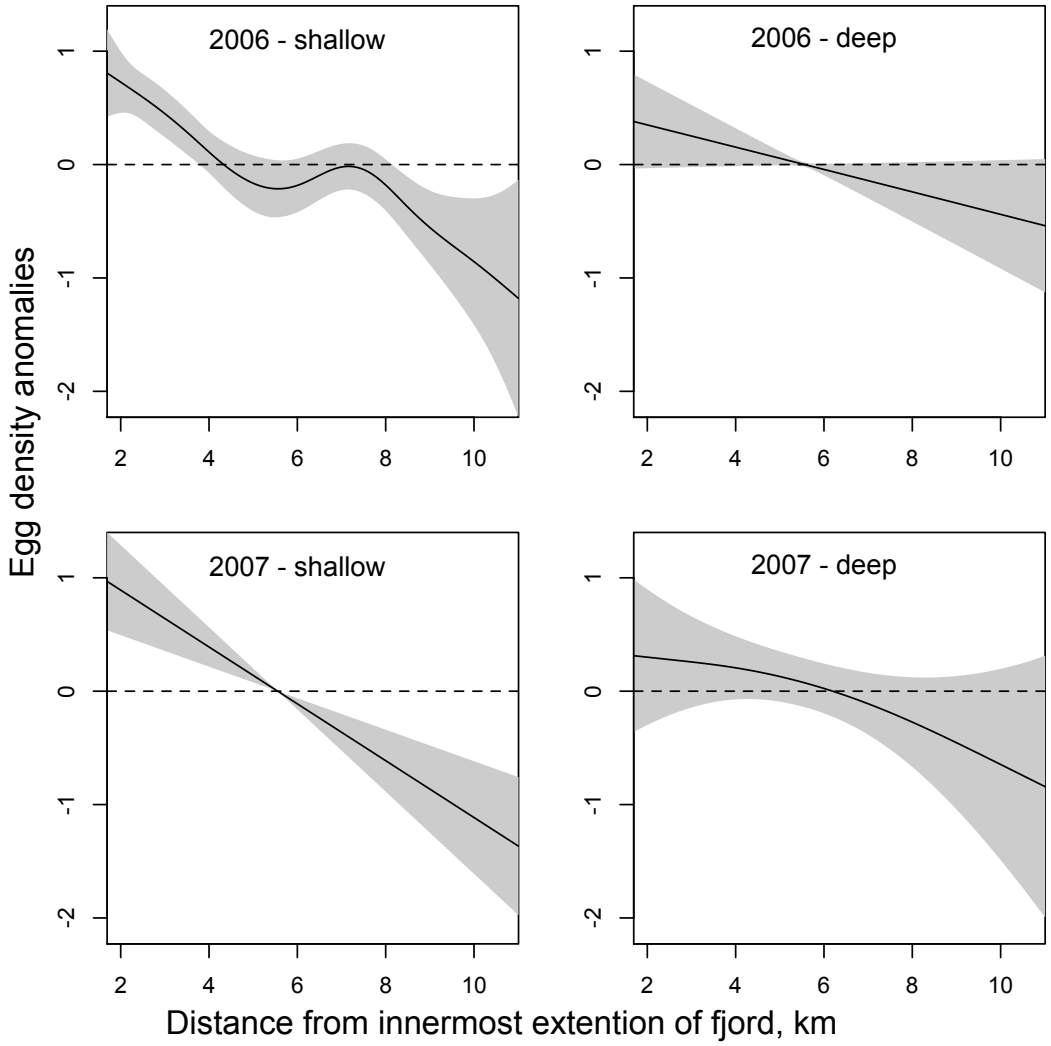


Figure 5. Effect of distance from shore on cod eggs catches in Tvedestrand fjord. Distance from shore is interacted with year (2006-07) and depth layer (<15m and >15m) so that there are a total of four combination of year and depth.

Spatial autocorrelation

All of what we have done above ignores the spatial autocorrelation of the observations. To get a sense for how much of a problem spatial autocorrelation is for our data set we can compute the empirical variograms of the y-variable (e.g., egg density or ATF cpue) and of the model residuals. The process is shown below for the pollock egg data in the Gulf of Alaska and in the accompanying R-script for the ATF CPUE data and relative model fit (gam1 in the case of ATF). For the computation of the variograms we will make use of the `geoR library`. The first step however, is to transform the coordinates from latitude and longitude to the Universal Transverse Mercator coordinate system. This allows us to express distances between pairs of points in metric length units (km) irrespective of their latitude. You will need to load the `PBSmapping library` to perform these operations. Here is the example for the pollock egg data (Fig. 6). See accompanying R script for another example on the ATF data.

```
> #Convert coordinates from Lat & Lon to UTM (km)
> subdataLL<-eggdata[,c(3,2)]
> names(subdataLL)<-c('X','Y')
> attr(subdataLL,'projection')<-'LL'
> subdataUTM<-convUL(subdataLL)
convUL: For the UTM conversion, automatically detected zone 5.
convUL: Converting coordinates within the northern hemisphere.
> subdataUTM$resd<-log(eggdata$catch+1)-pred
> subdataUTM$eggs<-log(eggdata$catch+1)

> #Estimate variogram of data and residuals using geoR library
> subset.geor.egg<-as.geodata(subdataUTM, coords.col=c(1,2),data.col=4)
> subset.geor.res<-as.geodata(subdataUTM, coords.col=c(1,2),data.col=3)
> vario.egg<-
variog(subset.geor.egg,uvec=seq(20,500,by=25),pairs.min=10)
variog: computing omnidirectional variogram
> vario.res<-
variog(subset.geor.res,uvec=seq(20,500,by=25),pairs.min=10)
variog: computing omnidirectional variogram
> plot(vario.egg,main='Variogram',xlab='Distance
(km)',ylab='Semivariance',type='b',scaled=T)
> lines(vario.res,lty=2,scaled=T)
> legend(350,0.5,c('egg density','residuals'),lty=c(1,2))
```

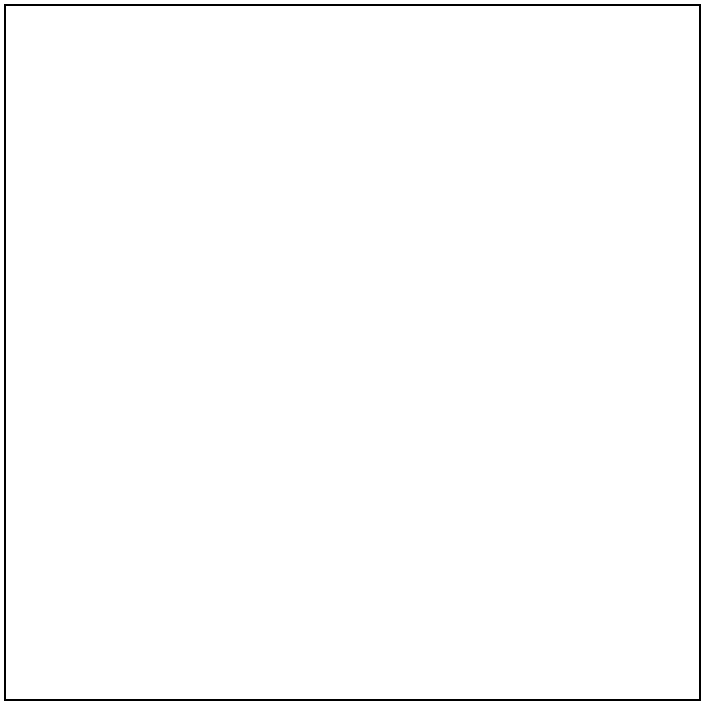


Figure 6. Empirical variogram of egg density and of residuals from two-step model fit for the pollock egg data. Note that we have computed the omnidirectional variograms up to 500km. Going beyond this distance between two points is not really necessary, because most of the action is expected to occur at smaller spatial scales, particularly with something as patchy as pollock eggs. Note also that the variograms are scaled, which means that the computed semivariance values are divided by the sample variance. This allows us to compare variograms from different data sets (i.e., density and residuals). The R note shows an example of variograms computation also for the ATF data.

We can fit the pollock egg data using a mixed effect model with a spatial autocorrelation to model the residuals. This approach will mostly affect the significance of the terms, and therefore inference, rather than the estimation of the parameters associated with the fixed effects (Fig. 7). It is advisable to keep the data case relatively small and model structure relatively simple, because the gamm routines take time to run and are known to have problems with convergence, in which case you can try to modify the control parameters as shown in the R notes.

```
> source('/Users/lciannel/Documents/MyDocuments/GAM
class/Flodevigen2/data/eggdata.R')
> subdata<-eggdata[eggdata$catch!=0,]
> egg1.corr<-
gamm(log(catch+1)~s(lon,lat)+s(log(bottom)),correlation=corGaus(form=~l
on+lat),data=subdata)
> summary(egg1.corr$gam)

Family: gaussian
Link function: identity

Formula:
```

```
log(catch + 1) ~ s(lon, lat) + s(log(bottom))

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.32843    0.09586   66.02   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
                 edf Ref.df     F  p-value
s(lon,lat)     20.024 20.024 21.10  < 2e-16 ***
s(log(bottom))  2.543  2.543 10.41 1.18e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.703  Scale est. = 2.3431    n = 256
```

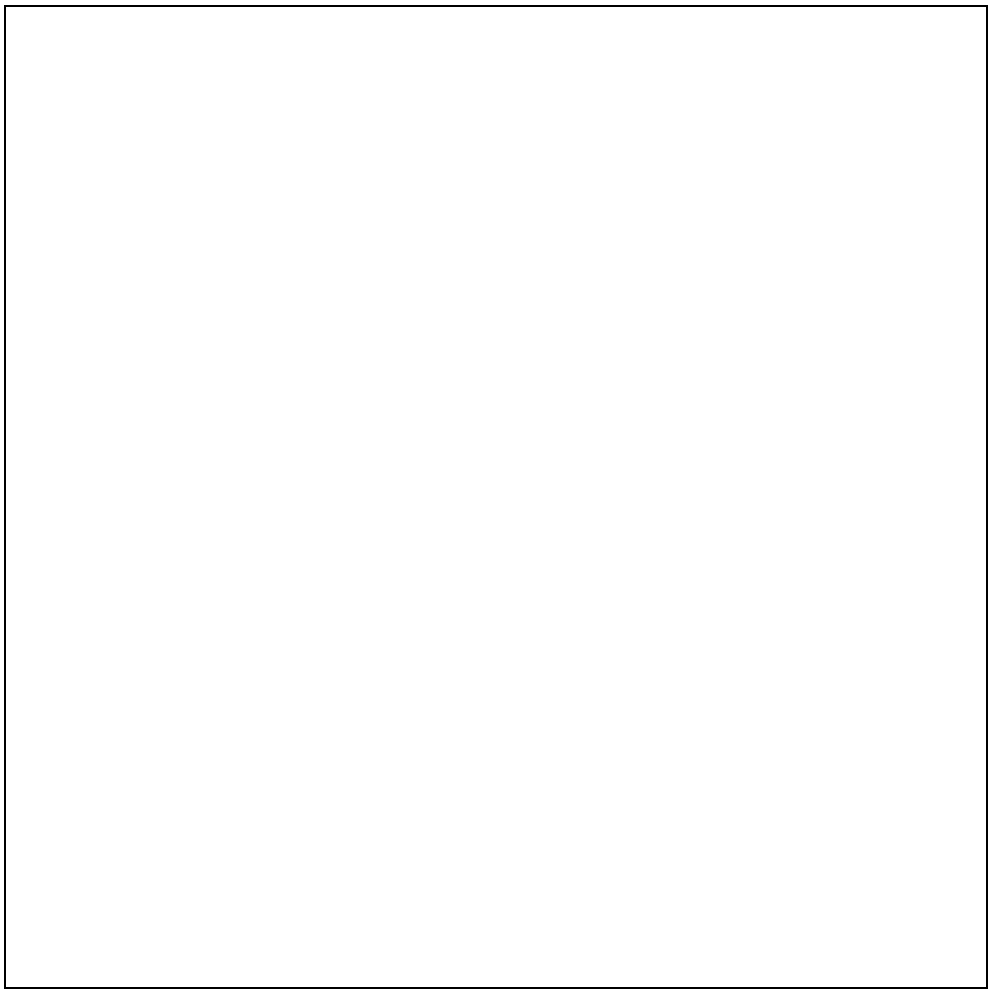


Figure 7. Pollock egg data. Covariate effect of the `egg1.corr` model. The first plot shows the effect (on additive scale) of latitude and longitude modeled with thin plate splines. Overimposed are the actual observations. The second plot shows the effect of bottom depth, log-transformed.

Bootstrap of p-value and confidence intervals

Another approach to deal with correlated error structures is by performing a wild bootstrap on the data and residuals obtained from a model with only fixed terms (i.e., gam not gamm) in order to estimate the confidence intervals and p-values of each fixed effect term. The logic behind this approach is rather simple. Namely, if there is a spatial correlation structure in the residuals, it is likely that the estimated p-values are too small. To control for this problem we can refit the same model using the (potentially) autocorrelated residuals as the response variable. This operation is repeated many times (say 1000) to generate a new reference value for the F and t of each smooth or parametric fixed term upon which we can recalculate the p-value of the entire model. The tricky part is to generate a random pool of residuals that is uncorrelated to the fixed term included in the model but that preserves the spatial structure of the original data. One relatively simple way to do so is to change the sign of the residuals within each grouping variable of the correlated error structure (e.g., year). Let see this in practice.

The wild bootstrap approach was applied by Llope et al. (2009) to model the distribution of phytoplankton in the North Sea. Here we apply it to the ATF data set, in which we suspect that there is a spatial autocorrelation of survey catches within each year (i.e., lat and lon are the design variables, and year is the grouping variable). The steps to compute the unbiased p-value for each fixed effect are the following.

1) Extract and rescale the residuals so that their variance is similar to the error variance of the model (i.e., scale of model in case of Normal distribution).

2) Within each grouping variable of the error structure (year in the case of the ATF data set), randomly change the sign of the residuals. Note: it is important that ALL the residuals within each grouping variable have the SAME sign change. This will ensure that the spatial residual structure is preserved.

3) Fit a GAM on the rescaled residuals and extract the new F and t values for the smooth and parametric terms, respectively.

4) Repeat steps 2-3 many times (i.e., 1000).

5) Calculate the new p-value for each term as the probability of the bootstrapped F or t values ≥ observed ones.

For the computation of the term confidence intervals we follow the steps 1 and 2,after which we 3) add the re-scaled residuals to the model predictions and fit a new GAM. 4) Repeat the steps 2-3 many times (i.e., 1000) and obtain a new family of predictions for each fixed effect term, upon which the 95% confidence intervals are calculated. The accompanying R notes have the script to run the wild bootstrap on the ATF data. The p-values obtained confirm the significance of the term included in the model.

P(S(lat,lon) = 0) <0.001
P(S(depth) = 0) <0.001
P(S(phi) = 0) = 0.001
P(avg.wt=0) = <0.001
P(bt=0) = <0.001.

The bootstrapped confidence intervals are shown in Fig. 8.

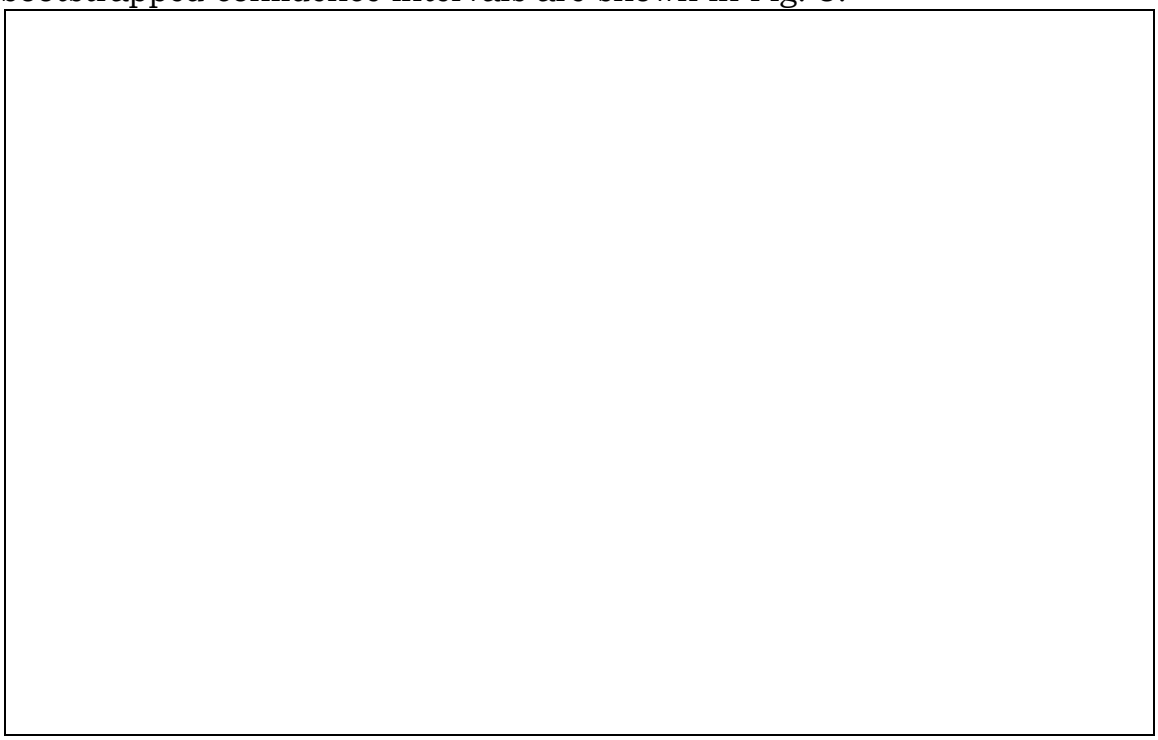


Figure 8. Bootstrapped term effect and confidence interval for the ATF model gam1.

Acknowledgements
Kung-Sik Chan has developed the threshold gam library and has been instrumental in helping me to better understand bootstrapping techniques and variable coefficients gam. Marcos Llope provided the code for the wild bootstrap. Valerio Bartolino was instrumental in the analysis of the ATF data set and in developing formulations for the mixed effect gam. Halvor Knutsen and Esben Moland Olsen provided the Norwegian cod eggs data. Kevin Bailey provided the pollock egg data. Steve Porter provided the pollock growth and condition data. Geor Ottersen provided the Barents Sea cod data. This version of the manual has benefitted tremendously from suggestions received by students who attended previous GAM workshops (Seattle, Newport, Flodevigen, Corvallis, Mallorca). I am indebted to them for their comments.

Literature cited

Chan KS (MS) Applied penalized regression analysis: from additivity to nonadditivity

Chan KS, Kristoffersen AB, Stenseth NC (2003) Bürmann Expansion and Test for Additivity. Biometrika 90: 209-222

Ciannelli L, Chan KS, Bailey KM, Stenseth NC (2004) Nonadditive effects of the environment on the survival of a large marine fish population. Ecol 085 (12): 3418-3427

Ciannelli L, Bailey K, Chan KS, Stenseth NC (2007a) Phenological and geographical patterns of walleye pollock spawning in the Gulf of Alaska. Can J Aquat Fish Sci 64:713-722

Ciannelli L, Dingsør G, Bogstad B, Ottersen G, Chan KS, Gjøsæter H, Stiansen JE, Stenseth NC. (2007b) Spatial anatomy of species survival rates: effects of predation and climate-driven environmental variability. *Ecology* 88: 635-646

Ciannelli L, Fauchald P, Chan KS, Agostini VN, Dingsør GE. (2008) Spatial fisheries ecology: recent progress and future prospects. *Journal of Marine Systems*.

Ciannelli L, Knutsen H, Moland E, Espeland SH, Asplin L, Jelmert A, Knutsen JA, Stenseth NC (2010) Small-scale genetic structure in a marine population in relation to water circulation and egg characteristics. *Ecology* 91(10): 2918-2930

Ciannelli L, Bartolino V, Chan KS (2012) Nonadditive and nonstationary properties in the spatial distribution of a large marine fish population. *Proceeding Royal Society London B* 279: 3635-3642

Fox CJ, O'Brien CM, Dickey-Collas M, Nash RDM (2000) Patterns in the spawning of cod (*Gadus morhua* L.), sole (*Solea solea* L.) and plaice (*Pleuronectes platessa* L.) in the Irish Sea as determined by generalized additive modeling. Fish Oceanogr 9: 33-49

Green PJ and Silverman BW (1994) Nonparametric regression and generalized linear models. Chapman and Hall, London (UK). Pp. 182

Hastie TJ, Tibshirani RJ (1990) Generalized Additive Models. Chapman and Hall, London UK. Pp. 335

Knutsen H, Olsen EM, Ciannelli L, Espeland SH, Knutsen JA, Simonsen JH, Skreslet S, Stenseth NC (2007). Egg distribution, bottom topography

and small-scale population structure in a coastal marine system. Marine Ecology Progress Series 333: 249-255

Llope M, Chan KS, Ciannelli L, Reid PC, Stige LC, Stenseth NC (2009) Effects of environmental conditions on the seasonal distribution of phytoplankton biomass in the North Sea. *Limnology & Oceanography* 54(2): 512-524

McCullagh P, Nelder JA (1989) Generalized Linear Models. Chapman and Hall, London.

Porter S.M, Ciannelli L., Hillgruber N., Bailey K.M., Chan K.S., Canino M.F., Haldorson L.J. (2005) Analysis of factors influencing larval walleye pollock *Theragra chalcogramma* feeding in Alaskan waters. *Marine Ecology Progress Series* 302: 207-217

Stenseth NC, Chan KS, Tavecchia G., Coulson TN, Mysterud A, Clutton-Brock TH, Grenfell BT (2005) Modelling nonadditive and nonlinear signals from climate noise in ecological time series: Soay sheep as an example. Proc. R Soc B: to appear.

Theilacker and Shen (2001) Evaluating growth of larval walleye pollock, *Theragra chalcogramma*, using cell cycle analysis. Mar. Biol. 138: 897-907.

Venables WN, Ripley BD (2002) Modern Applied Statistics with S. Fourth Edition. Springer, 2002. ISBN

Venables WN, Dichmont CM (2004) GLMs, GAMs and GLMMs: an overview of theory for applications in fisheries research Fish Res 70: 315-333

Wood SN (2000) Modelling and Smoothing Parameter Estimation with Multiple Quadratic Penalties. J R Statist Soc B 62(2):413-428

Wood SN (2003) Thin plate regression splines. J R Statist Soc B 65(1):95-114

Wood SN (2004) Stable and efficient multiple smoothing parameter estimation for generalized additive models. J Amer Statist Ass

Wood SN, Augustin NH (2002) GAMs with integrated model selection using penalized regression splines and applications to environmental modeling. Ecol Model 157: 157-177

Wood, S.N. (2006) *Generalized additive models: an introduction with R*. New York: Chapman and Hall, 391 pp.

Additional references on the applications of GAM to ecological data and on GAM theory

Applications

Swartzman G, Huang C, Kalzuny S (1992) Spatial analysis of the Bering Sea groundfish survey data using generalized additive models. Can J Fish Aquat Sci 49: 1366-1378

Maravelias CD, Reid DG (1997) Identifying the effects of oceanographic features and zooplankton on prespawning herring abundance using generalized additive models. Mar Ecol Progr Ser 147: 1-9

Schulter D (1998) Estimating the form of natural selection on a quantitative trait. Evol 42(5): 849-861

Fox CJ, O'Brien CM, Dickey-Collas M, Nash RDM (2000) Patterns in the spawning of cod (*Gadus morhua* L.), sole (*Solea solea* L.) and plaice (*Pleuronectes platessa* L.) in the Irish Sea as determined by generalized additive modeling. Fish Oceanogr 9: 33-49

Denis V, Lejeune J, Robin JP (2002) Spatio-temporal analysis of commercial trawler data using General Additive models: patterns of Loliginid squid abundance in the north-east Atlantic. ICES J. Mar Sci 59: 633-648

Cardinale M, Svedang H (2004) Modelling recruitment and abundance of Atlantic cod, *Gadus morhua*, in the eastern Skagerrak-Kattegat (North Sea): evidence of severe depletion due to a prolonged period if high fishing pressure. Fish Res 69: 263-282

Ciannelli L, Chan KS, Bailey KM, Stenseth NC (2004) Nonadditive effects of the environment on the survival of a large marine fish population. Ecol 085 (12): 3418-3427

Granadeiro JP, Andrade J, Palmerin JM (2004) Modelling the distribution of shorebirds in estuarine areas using generalized additive models. J Sea Res 52: 227-240

Hedger R, McKenzie E, Heath M, Wright P, Scott B, Gallego A, Andrews J (2004) Analysis of the spatial distributions of mature cod (*Gadus morhua*) and haddock (*Melanogrammus aeglefinus*) abundance in the North Sea (1980-1999) using generalized additive models. Fish Res 70 (1): 17-25

Seoane J, Bustamante J, Diaz-Delgado R (2004) Competing role for landscape, vegetation, topography and climate in predictive models of bird distribution. Ecol Model 171: 209-222

Zagaglia CR, Lorenzzetti JA, Stech J (2004) Remote sensing data and longline catches of yellowfin tuna (*Thunnus albacore*) in the equatorial Atlantic. Rem Sens Env 93: 267-281

Entire issue of Fisheries Research vol. 70 (2-3): ***Models in Fisheries Research: GLMs, GAMS and GLMMs*** *Edited by Yongshun Xiao, Andre E. Punt, Russell B. Millar and Terrance J. Quinn II.* Some articles from this issue are listed below.

Xiao Y, Punt AE, Russel BM, Quinn TJ II (2004) Models in fisheries research: GLMs, GAMS and GLMMs. Fish Res 70 (2-3) 137-139

Venables WN, Dichmont CM (2004) GLMs, GAMs and GLMMs: an overview of theory for applications in fisheries research Fish Res 70: 315-333

Statistical methods

Hastie TJ, Tibshirani RJ (1990) Generalized Additive Models. Chapman and Hall, London UK. Pp. 335

Green PJ and Silverman BW (1994) Nonparametric regression and generalized linear models. Chapman and Hall, London (UK). Pp. 182

Wood SN (2000) Modelling and Smoothing Parameter Estimation with Multiple Quadratic Penalties. J R Statist Soc B 62(2):413-428

Wood SN (2001) mgcv:GAMs and Generalized Ridge Regression for R. R News 1(2):20-25 (Available on the R-project web site)

Crawley MJ (2002) Statistical computing: an introduction to data analysis using S-Plus. Wiley, England UK. Pp 761

Venables WN, Ripley BD (2002) *Modern Applied Statistics with S. Fourth Edition*. Springer, 2002. ISBN

Entire issue of Ecological Modelling (2002) 157. Some articles from this issue are listed below:

Austin MP Spatial predictions of species distribution: an inference between ecological theory and statistical modeling. Ecol Model 157: 101-118

Barry SC, Welsh AH (2002) Generalized additive modeling and zero inflated count data. Ecol Model 157: 179-188

Guisan A, Edwards TC, Hastie T (2002) Generalized linear and generalized additive models in studies of species distributions: setting the space. Ecol Model 157: 89-100

Heegaard E (2002) The outer border and central border for species-environmental relationships estimated by non-parametric generalized additive models. Ecol Model 157: 131-139

Wood SN, Augustin NH (2002) GAMs with integrated model selection using penalized regression splines and applications to environmental modeling. Ecol Model 157: 157-177

Wood SN (2003) Thin plate regression splines. J R Statist Soc B 65(1):95-114

Wood SN (2004) Stable and efficient multiple smoothing parameter estimation for generalized additive models. J Amer Statist Ass