Hierarchical Generalized Additive Models: an introduction with mgcv

Abstract

As the amount and quality of ecological data has grown, and as ecological questions have become increasingly complex, ecological statistics has moved from linear models to more flexible and structured models, including hierarchical generalized linear models (HGLMs) to model complex patterns of variabilty, and generalized additive models (GAMs) to model nonlinear relationships between covariates and outcomes. In this paper, we discuss an extension to both those ideas, the hierarchical GAM (HGAM), that allows the user to model nonlinear functional relationships between covariates and outcomes where the shape of the function itself varies between different grouping levels (say between different species in a community, or between different locations on a landscape). We describe the theoretical connection between these models and standard HGLMs and GAMs, how to model different assumptions about the degree of inter-group variability in functional response, and show how HGAMs can be readily fit using existing GAM software, the mgcv package in R. We also discuss computational and statistical issues with fitting these models, and demonstrate how to fit HGAMs on several example datasets.

I: Introduction

As ecology has progressed as a quantitative discipline and the questions ecologists ask have become more complicated, the statistical techniques ecologists use have increased in their flexibility to model complex relationships. Two of the more popular and powerful techniques now in use are generalized additive models (GAMs; Simon N Wood 2006b) for modelling flexible regression functions, and generalized linear mixed models ("hierarchical generalized linear models" (HGLMs) or simply "hierarchical models"; Bolker et al. 2009; Gelman et al. 2013) for modelling between-group variability in regression relationships.

At first glance, GAMs and HGLMs are very different tools. GAMs are used to estimate smooth functional relationships between predictor variables and the response, assuming that the phenomena under investigation is not linear (in the GLM sense) but is a smooth function of the predictor variables. Examples of such relationships would be the vertical distribution of abundance of a population as a function of depth (Stanley, Pedersen, and Snelgrove 2016) or the swimming speed of snakes as function of temperature (Vickers, Aubret, and Coulon 2017). HGLMs, on the other hand, are used to estimate linear relationships between predictor variables and response, but impose a structure where predictors are organized into groups (often referred to as "blocks") and the relationships between predictor and response may differ between those groups. Either the slope or intercept, or both, may be subject to grouping. A typical example of HGLM use might be to include site-specific effects in a model of counts, or to model individual level heterogeneity in a study with repeated observations of multiple individuals.

Both GAMs and HGLMs can be used to fit potentially highly variable models by "pooling" parameter estimates towards one another. The connection between the two methods is quite deep and GAMs may be interpreted (and fitted) as HGLMs and vice-versa. Given this connection, the obvious extension to the standard GAM framework is to allow the smooth functional relationship between predictor and response to vary between different grouping levels, but in such a way that the different functions are in some sense pooled toward each other. We often want to know both how the functional relationship between varies between groups, and if there is a strong relationship on average across groups. We will refer to this type of model as a *hierarchical GAM*, or HGAM.

There are many potential uses for HGAMs. For example, to estimate how the maximum size of different fish species varies along a common temperature gradient (figure 1). Each species will typically have its own response function, but since the species overlap in range, they should have similar responses over at least some of the temperature gradient; figure 1 shows all three species reach their largest maximum sizes in the center of the temperature gradient. Estimating a seperate function for each species throws away a lot of shared information and could result in highly noisy function estimates if there were only a few data points for each species. Estimating a single average relationship could result in an average function that did not predict any specific group well. In our example, using a single global temperature-size relationship would miss the three species distinct temperature optima, and that the orange species is significantly smaller at all temperatures than the other two (figure 1). We prefer a hierarchical model that includes a global temperature-size curve plus species-specific curves that were penalized to be close to the mean function.

The ability to fit HGAMs already exists in the popular *mgcv* package for the R statistical programming language. There are many different options available respresenting different model assumptions with corresponding trade-offs. This paper will discuss the different approaches to group-level smoothing, the options for each and why a user might choose them, and demonstrate the different approaches across a range of case studies.

This paper is divided into five sections. Part II is a brief review of how GAMs work and their relation to hierarchical models. In part III, we discuss different ways of fitting HGAMS, what assumptions each model makes about how information is shared between groups, and different ways of specifying these models in *mgcv*. In part IV, we discuss some of the computational and statistical issues involved in fitting HGAMs in *mgcv*. Finally, in part V, we work through examples analyses using this approach, to demonstrate the modelling process and how HGAMs can be incorporated into the quantitative ecologist's toolbox.

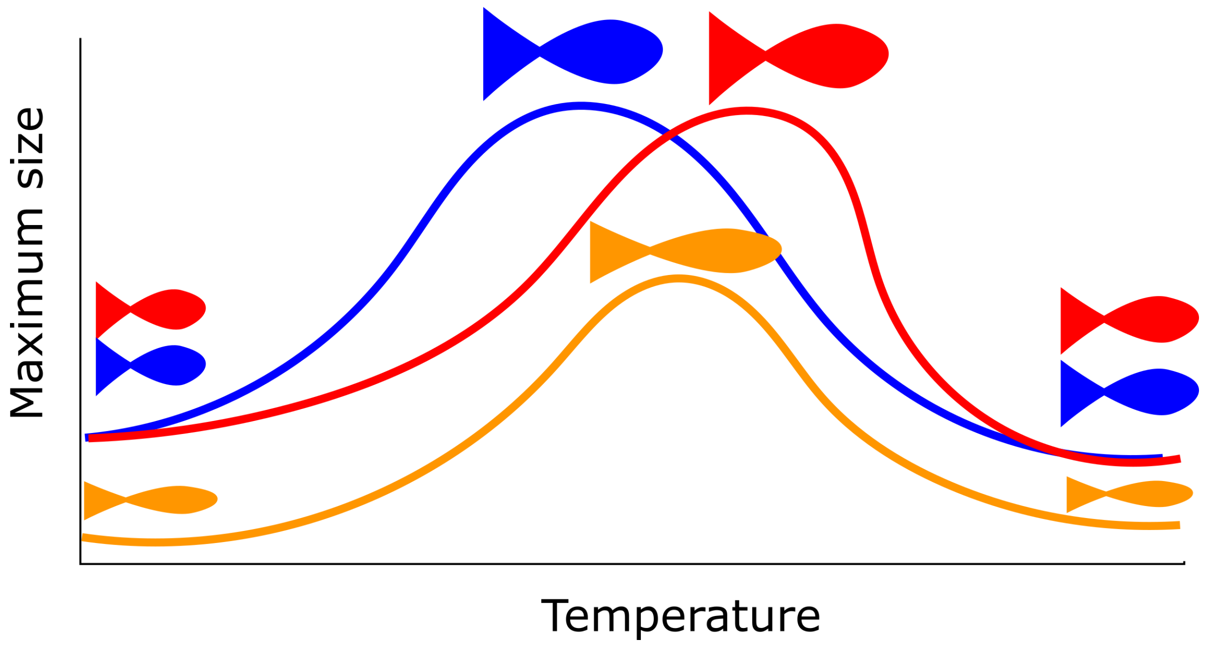

Figure 1: Hypothetical example of functional variability between different group levels. Each line indicates how the maximum possible body size for different species of fish in a community might vary as a function of average water temperature. While the orange species shows lower maximum size at all temperatures, and the red and blue species differ in what temperature they can acheive the maximum possible size, all three curves are similiarly smooth, and peak close to one another, relative to the entire range of tested temperatures.

II: an introduction to Generalized Additive Models

One of the most common model formulations in statistics is the generalized linear model (McCullagh and Nelder 1989) that relates a response (y) to a linear combination of explanatory variables. The response is assumed to be conditionally distributed according to a member of the exponential family (e.g., letting the response be a trial, a count or a strictly positive real number — binomial, Poisson or Gamma distributions, respectively). The generalized additive model (GAM; Hastie and Tibshirani 1990; Ruppert, Wand, and Carroll 2003; Simon N Wood 2006b) allows the relationships between the explanatory variables (henceforth covariates) and the response to be described by smooth terms (usually *splines* (Boor 1978), but potentially other structures). In general we have models of the form:

$$\mathbb{E}(Y) = g^{-1}\left(\beta_0 + \sum_{j=1}^{J} f_j(x_j)\right),$$

where $\mathbb{E}(Y)$ is the expected value of the response Y (with an appropriate distribution and link function g), f_j is a smooth function of the covariate x_j, β_0 is an intercept term and g^{-1} is the inverse link function. Here there are J smooths and each is a function of only one covariate, though it is possible to construct smooths of multiple variables.

Each smooth f_j is represented by a sum of simpler *basis functions* ($b_{j,k}$) multiplied by corresponding coefficients ($\beta_{j,k}$), which need to be estimated:

$$f_j(x_j) = \sum_{k=1}^{K} \beta_{j,k} b_{j,k}(x_j),$$

The size K of each smooth will determine the complexity of the resulting term (referred to as "basis size", "basis complexity" or "basis richness"). Though it seems like the basis can be overly complex ("how big should I make K?") and lead to overfitting, we need not worry about this as we use a penalty to ensure that each function's complexity is appropriate; hence the basis only need be *large enough* and we let the penalty deal with excess wiggliness.

The penalty for a term is usually based on a derivative of the smooth, as the derivative is used to measure the wiggliness of the function and hence its flexibility. Model fit is measured by its likelihood ($\mathcal{L}(\mathbf{y}|\theta)$, the likelihood of observing the data, $\mathbf{y}$, given the parameters θ). We trade-off the fit of the model against the wiggliness penalty to obtain a model that fits the data well but does not overfit. To control this trade-off we estimate a *smoothness parameter* for each smooth λ_j. In the case of a single smooth term with a single penalty, this trade-off results in the penalised likelihood $\mathcal{L}_p$ of the estimated model[1]:

$$\mathcal{L}_p = \underbrace{\mathcal{L}(\mathbf{y}|\beta)}_{\text{model fit}} + \lambda \underbrace{\beta^{\mathsf{T}}\mathbf{S}\beta}_{\text{wiggliness}}.$$

[1]The addition of more smooths (or more penalty terms for each smooth) would simply mean changing the term $\lambda\beta^{\mathsf{T}}\mathbf{S}\beta$ into a sum over several such products.

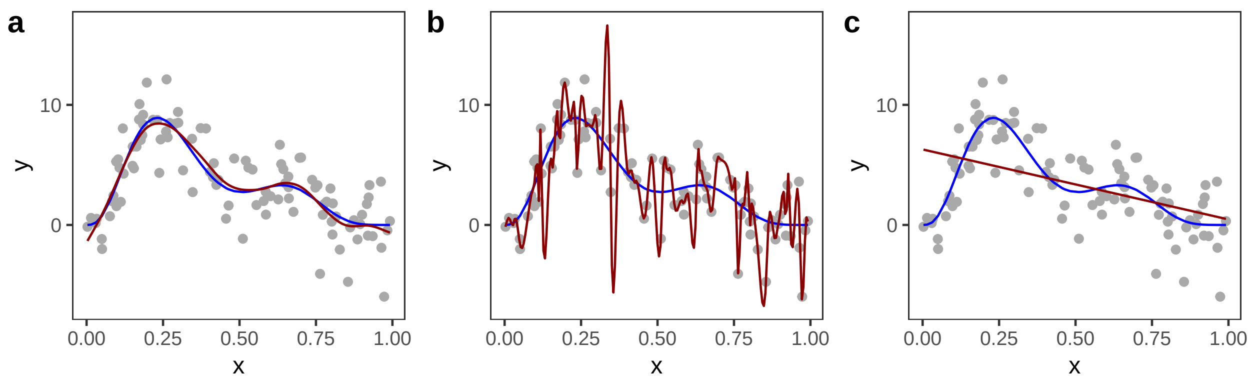

Figure 2: Examples of how different choices of the smoothing parameter effect the resulting function. Data (points) were generated from the blue function and noise added to them. In the left plot the smoothing parameter was estimated using REML to give a good fit to the data, in the middle plot the smoothing parameter was set to zero, so the penalty has no effect and the function interpolates the data, the right plot shows when the smoothing parameter is set to a very large value, so the penalty removes all terms that have any wiggliness, giving a straight line. Numbers in the y axis labels show the estimated degrees of freedom for the term.

Where $\mathbf{S}$ is the penalty matrix for the smooth, which determines how strongly to penalize different linear combinations of parameters, and depends on the type of smooth used. There are several different criteria used to estimate penalty parameters, including generalized cross validation (GCV), maximizing marginal likelihood (ML), or restricted maximum likelihood (REML). In general, REML seems to give the best performance, as ML tends to underestimate wiggliness, and GCV can overestimate wiggliness (Wood 2011).

Figure 2 shows REML-based optimal smoothing in the first plot; the second plot shows what happens when the smoothing parameter is set to zero, so the penalty has no effect (interpolation); the right plot shows when the smoothing parameter is set to a very large value, giving a straight line. Smooths of this kind are often referred to as a *basis-penalty smoothers*.

The number of basis functions, K, limits the maximum complexity for a given smooth term. To measure the complexity of a smooth term, we use the *effective degrees of freedom* (EDF), which at a maximum is the number of coefficients to be estimated in the model, minus any constraints. The EDF can take non-integer values and larger values indicate more wiggly terms. See Wood (2006b, Section 4.4) for further details.

There are many possible basis functions that can be used to construct the b_k. In the examples in this paper, we will use three types of smoother for illustration: thin plate regression splines, cyclic cubic smooths, and random effects.

Thin plate regression splines (TPRS), which have a wide range of appealing theoretical properties and are implemented in (Wood 2003)). Thin plate splines are defined based on the order of derivative that is penalized (which we will refer to as m). When $m = 1$, the penalty matrix associated with the TPRS penalizes the integral of the squared first derivative of the TPRS across the range of the data, when $m = 2$ it penalizes the squared second derivative, etc. Smooths fit with higher order TPRS are typically visually more smooth. When we refer to TPRS, we will typically be referring to the version where $m = 2$; however, we will see in section III that it can be useful to use $m = 1$ TPRS when fitting more complicated HGAMs. TPRS are also defined for any number of predictors, so multivariate smoothers can be constructed easily. The basis is *isotropic*: smoothness is treated the same in all directions. Example basis functions and penalty matrix $\mathbf{S}$ for a $m = 2$ TPRS with six basis functions for evenly spaced data are shown in figure 3.

Cyclic cubic smoothers are another continuous smoother that again penalizes the squared second derivative of the smooth across the function, but are designed so that the the value of the function at the start and end of the covariate have the same value and first derivative, and zero second derivative. We will use these smoothers to demonstrate how to fit HGAMs to cyclic data.

We can also think about random effects as "smooths" in this framework, if we take pragmatic Bayesian point of view and consider the penalty matrix S to be the inverse of the covariance matrix (i.e. a precision matrix) of the basis function coefficients (Kimeldorf and Wahba 1970; Wood 2017). For instance, to include a simple single-level random effect to account for variation in group means (intercepts) there will be one basis function for each level of the grouping variable, that takes a value of 1 for any observation in that group and 0 for any observation not in the group. The penalty matrix for these terms is a n_g by n_g identity matrix, where n_g is the number of groups. This means that each group-level coefficient will be penalized in proportion to its squared deviation from zero. This is equivalent to how random effects are estimated in standard mixed effect models. The penalty term is then proportional to the inverse of the variance of the fixed effect estimated by standard hierarchical model solvers (Verbyla et al. 1999). This connection between random effects and basis function smooths extends beyond the varying-intercept case. Any basis-function representation of a smooth can be transformed so that it can be represented as a combination of a random effect with an associated variance, and possibly one or more fixed effects, corresponding to functions in the null space of the original basis-function (see below). While this is beyond the scope of this paper, see Verbyla et al. (1999) or Wood, Scheipl, and Faraway (2013) for a more detailed discussion on the connections between these approaches.

Smoothing penalties vs. shrinkage penalties

Penalties can have two effects on how well a model fits: they can penalize how wiggly a given term is (smoothing) and they can penalize the absolute size of the function (shrinkage). The penalty can only effect the components of the smooth that have derivatives (the *range space*), not the other parts (the *null space*). For 1-dimensional thin plate regression splines (when $m = 2$), this means that there is a linear term left in the model, even when the penalty is in full force (as $\lambda \to \infty$), as shown in figure 3[2]. Figure 3 shows an example of what the

[2]This is also why figure 2c resulted in a linear fit to the data.

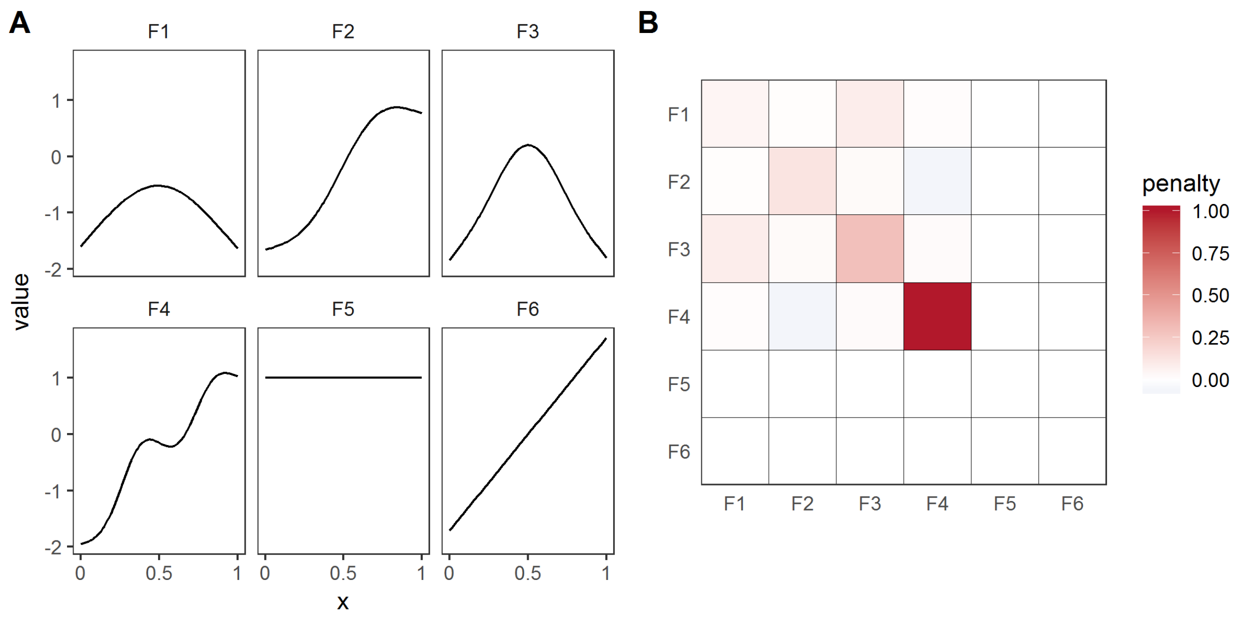

Figure 3: a) Examples of the basis functions associated with a six basis function thin plate spline (m=2), calculated for x data spread evenly between $x = 0$ and $x = 1$. Each line represents a single basis function. To generate a given smooth function, each basis function would be multiplied by its own coefficient. b) The smoothing penalty matrix for the thin plate smoother. Red entries indicate positive values and blue indicate negative values. For example, for the thin plate spline, functions F3 and F4 would have the greatest proportionate effect on the total penalty (as they have the largest values on the diagonal), whereas function F5 and F6 would not contribute to the wiggliness penalty at all (all the values in the 5th and 6th row and column of the penalty matrix are zero). This means these functions are in the null space of this basis, and are treated as completely smooth.

basis functions (Fig. 3A), and smoothing penalty (Fig. 3B) look like for a 6-basis function thin-plate spline. The random effects smoother we discussed earlier is an example of a pure shrinkage penalty; it penalizes all deviations away from zero, no matter the pattern of those deviations. This will come in useful later in section III, where we use random effect smoothers as one of the components of a HGAM.

Interactions between smooth terms

It is also possible to create interactions between covariates with different smoothers (or degrees of smoothness) assumed for each covariate, using *tensor products*. For instance, if one wanted estimate the interacting effects of temperature and time on some outcome, it would not make sense to use a two-dimensional TPRS smoother, as that would assume that a one degree change in temperature would equate to a one second change in time. Instead, a tensor product allows us to create a new set of basis functions that allow for each marginal function (here temperature and time) to have its own marginal smoothness penalty. Each marginal smooth can use a different type of basis, as wanted or dictated by the data.

There are two approaches used in mgcv for generating tensor products. The first approach (Simon N. Wood 2006) essentially creates an interaction of each pair of basis functions for each marginal term, and a penalty for each marginal term that penalizes the average average wiggliness in that term; in mgcv, these are created using the `te` function. The second approach (Wood, Scheipl, and Faraway 2013) seperates each penalty into penalized (rank-space) and unpenalized (null-space) components, then creates new basis functions and penalties for all pair-wise combinations of penalized and unpenalized components between all pairs of marginal bases; in mgcv; these are created using the `t2` function. The advantage of the first method is that it requires fewer smoothers, so is faster to estimate in most cases. The advantage of the second method is that the tensor products created this way only have a single penalty associated with each basis function (unlike the `te` approach, where each both penalties apply to all basis functions), so it can be fit using standard mixed effect software such as lme4 (Bates et al. 2015).

Comparison to hierarchical linear models

Generalized linear mixed effect models (GLMMs; also referred to as hierarchical generalized linear models, multilevel models etc; e.g., Bolker et al. 2009; Gelman 2006) are an extension of regression modelling that allow the modeller to include terms in the model that account for structure in the data — the structure is usually of the form of a nesting of the observations. For example individuals are nested within sample sites, sites are nested within forests and forests within states. The depth of the nesting is limited by the fitting procedure and number of parameters to estimate.

HGLMs are a highly flexible way to think about grouping in data; the groupings used in models often refer to the spatial or temporal scale of the data (McMahon and Diez 2007) though can be based on any useful grouping.

We would like to be able to think about the groupings in our data in a simple way, even when the covariates in our model are related to the response in a non-linear way. The next section investigates the extension of the smoothers we showed above to the case where each observation is in a group, with a group-level smooth.

III: What are hierarchical GAMs?

What do we mean by hierarchical smooths?

The smooths in section II allowed us to model flexible relationships between our response and predictor variables. In this section, we will describe how to model inter-group variability using smooth curves and how to fit these models in *mgcv*. Model structure is key in this framework, so we start with three choices:

1. Should each group have its own smooth, or will a global smooth term suffice?
2. Do all of the group-specific curves have the same wiggliness, or should each group have its own smoothing parameter?
3. Will the smooths for each group have a similar shape to one another — a shared average curve[3]?

These three choices result in five possible models (figure 4):

1. A single common smooth for all observations.
2. A single common smooth plus group-level smooths that have the same wigglyness.
3. A single common smooth plus group-level smooths with differing wigglyness.
4. Group-specific smooths without an average trend, but with all smooths having the same wigglyness.
5. Group-specific smooths with different wigglyness.

It is important to note that "similar wiggliness" and "similar shape" are two distinct concepts; functions can have very similar wiggliness but very different shapes. Wiggliness measures how quickly a function changes across its range, and it is easy to construct two functions that differ in shape but have the same wiggliness. For example, a logistic curve might have the same squared total second derivative between -1 and +1 as a sine curve, but they have very different shapes. Figure 4, model 4 illustrates this case.

Similarly, two curves could have very similar overall shape, but differ in their wiggliness. For instance, if one function was equal to the second function plus a high-frequency osscilation. Figure 4 model 3 illustrates this.

We will discuss the trade-offs between different models and guidelines about when each of these models is appropriate in section IV. The remainder of this section will focus on how to specify each of these five models using *mgcv*.

Coding hierarchical GAMs in R

Each of these models can be coded straightforwardly in *mgcv*. To help illustrate this throughout the section when describing how to set these models up, we will refer to the response variable as `y`, continuous predictor variables as `x` (or `x1` and `x2`, in the case multiple predictors), and `fac` to designate the discrete grouping factor whose variation we are interested in understanding[4].

[3]For this paper, we consider two functions to have similar shape if the average squared distance between the functions is small (assuming the functions have been scaled to have a mean value of zero across their ranges). This definition is somewhat restricted; for instance, a cyclic function would not be considered to have the same shape as a phase-shifted version of that function, nor would two normal distributions with the same mean but different standard deviations. The benefit of this definition of shape, however, is that it is straightforward to translate into quadratic penalties as we have been using.

[4]Note that it is important to know how the group-level variable `fac` is coded in R. If it is coded as a character, `mgcv` will raise an error message, as it requires a factor. It is also important to know whether the

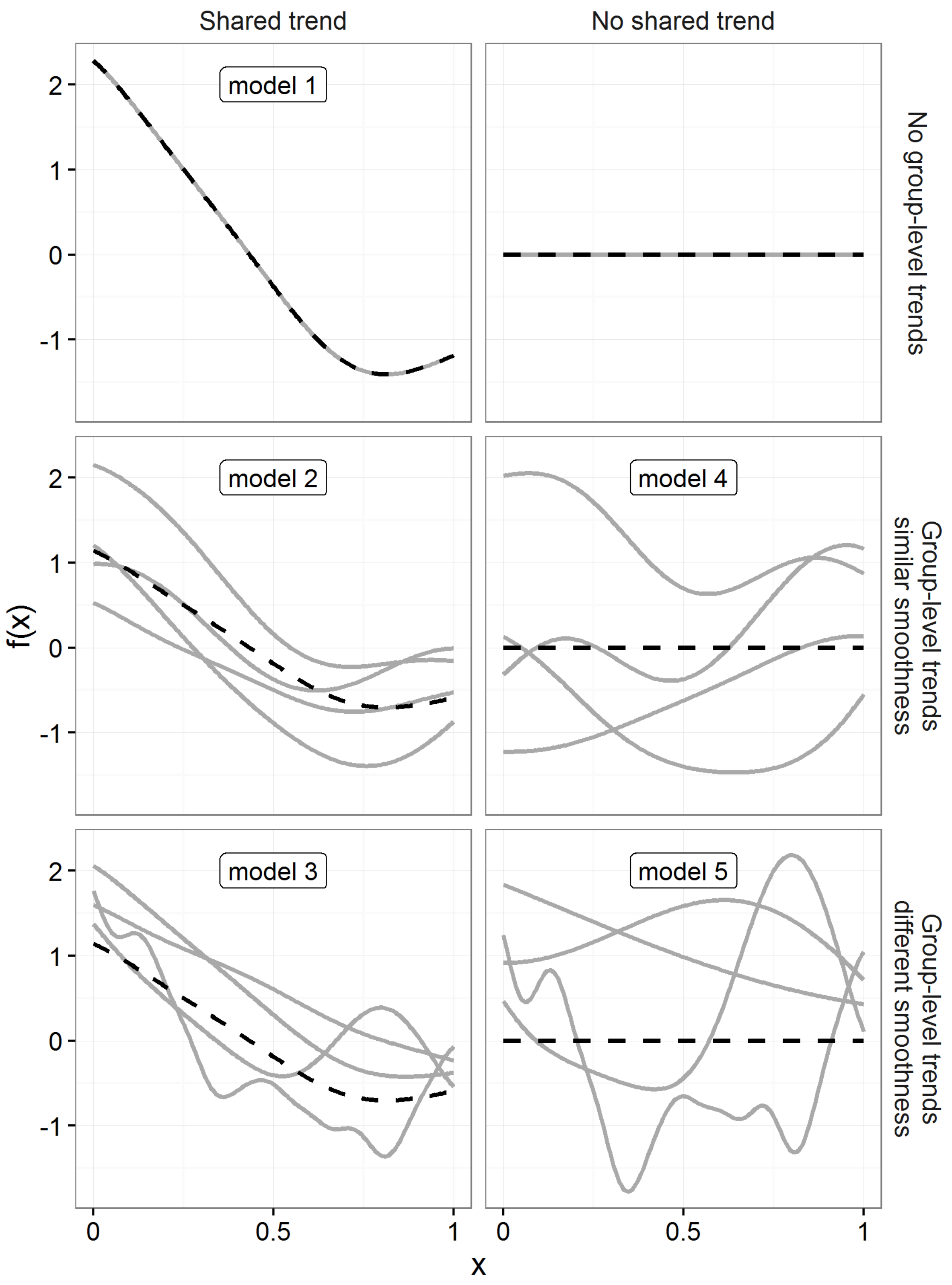

Figure 4: Alternate types of functional variation f(x) that can be fitted with HGAMs. The dashed line indicates the average function value for all groups, and each solid line indicates the functional value at a given predictor value for an individual group level.

We will also use two example datasets to demonstrate how to code these models (see the appendix for code to reproduce these examples):

A. The `CO2` dataset, available in R via the `datasets` package. This data is from an experimental study by Potvin, Lechowicz, and Tardif (1990) of CO_2 uptake in grasses under varying concentrations of C)$_2$, measuring how concentration-uptake functions varied between plants from two locations (Mississippi and Quebec) and two temperature treatments (chilled and warm). Twelve plants were used and CO_2 uptake measured at 7 CO_2 concentrations for each plant (figure 5a). Here we will focus on how to use these HGAMs to estimate inter-plant variation in functional responses.

B. A hypothetical study of what bird movement might look like along a migration corridor, sampled throughout the year. We have simulated this data for this paper (see appendix). This dataset consists of records of numbers of observed locations of 100 tagged individuals each from six species of bird, at ten locations along a latitudinal gradient, with one observation taken every four weeks. Not every bird was observed at each time point, so counts vary randomly between location and week. The data set (`bird_move`) consists of the variables `count`, `latitude`, `week` and `species` (figure 5b). This example will allow us to demonstrate how to fit these models with interactions and with non-normal (count) data.

Throughout the examples we use Restricted Maximum Likelihood (REML) to estimate model coefficients and smoothing parameters. We strongly recommend using either REML or marginal likelihood (ML) when fitting GAMs for the reasons outlined in (Wood 2011). In each case some data processing and manipulation has been done to obtain the graphics and results below. We have included all the code needed to make the figures for this document in the online appendix, and on the Github page associated with this paper (github.com/noamross/mixed-effect-gams).

A single common smooth for all observations (Model 1)

We start with the simplest model we can in our framework and include many details here to ensure that readers are comfortable with the terminology and R functions we are going to use later.

For our `CO2` data set, we will model $\log_e$(`uptake`) as a function of two smooths: a thin plate regression spline of log concentration, and a random effect for species to model species-specific intercepts.[5] Mathematically:

factor is coded as ordered or unordered (see `?factor` for more details on this). This matters when fitting groupwise smooths using the `by=` argument (as is used for fitting models 3 and 5, shown below). if the factor is unordered, *mgcv* will set up a model with one smooth for each grouping level. If the factor is ordered, *mgcv* will set any basis functions for the first grouping level to zero. In model 3 the ungrouped smooth will then correspond to the first grouping level, rather than the average functional response, and the group-specific smooths will correspond to deviations from the first group. In model 5, using an ordered factor will result in the first group not having a smooth term associated with it at all.

[5]Note that we're actually modelling ln(uptake); this can be a useful approach when dealing with estimating multiple functional relationships as it means that functions that differ from each other by a multiplicative constant (so $f_1(x) = \alpha \cdot f_2(x)$ will differ by an additive constant when log-transformed (which can be estimated

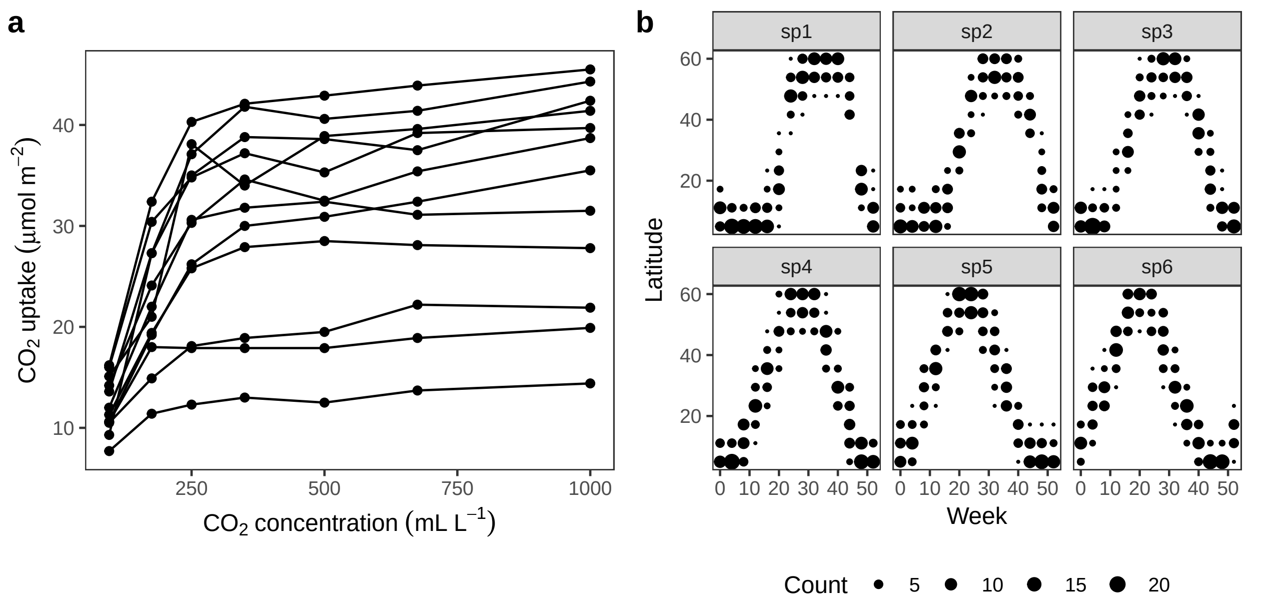

Figure 5: Example data sets used throughout section III. a) Grass CO$_2$ uptake versus CO$_2$ concentration for 12 individual plants (black lines). b) Simulated data set of bird migration, with point size corresponding to weekly counts of 6 species along a latituidinal gradient (zeros excluded for clarity).

$$\log_e(\texttt{uptake}_i) = f(\log_e(\texttt{conc}_i)) + \zeta_{\texttt{Plant\_uo}} + \epsilon_i$$

where $\zeta_{\texttt{Plant\_uo}}$ is the random effect for plant and ϵ_i is a Gaussian error term. We assume that $\log_e(\texttt{uptake}_i)$ is normally distributed.

In R we can write our model as:

```
CO2_mod1 <- gam(log(uptake) ~ s(log(conc), k=5, bs="tp") +
                              s(Plant_uo, k=12, bs="re"),
                data=CO2, method="REML")
```

This is the typical GAM setup, with a single smooth term for each variable. Specifying the model is similar to specifying a GLM in R via `glm()`, with the addition of `s()` terms to include one-dimensional or isotropic multidimensional smooths. The first argument to `s()` are the terms to be smoothed, the type of smooth to be used for the term is specified by the `bs` argument, and the number of basis functions is specified by `k`[6].

Figure 6 illustrates `mgcv`'s default plotting out for `CO2_mod1`: the left panel shows the estimated global functional relationship, and the right shows a quantile-quantile plot of the

by simple random effects): $ln(f_1(x)) = ln(\alpha) + ln(f_2(x))$.

[6]Due to identifiability or other constraints (e.g. cyclic smooths) arising from the type of smoother, the actual number of basis functions used may be less than `k`.

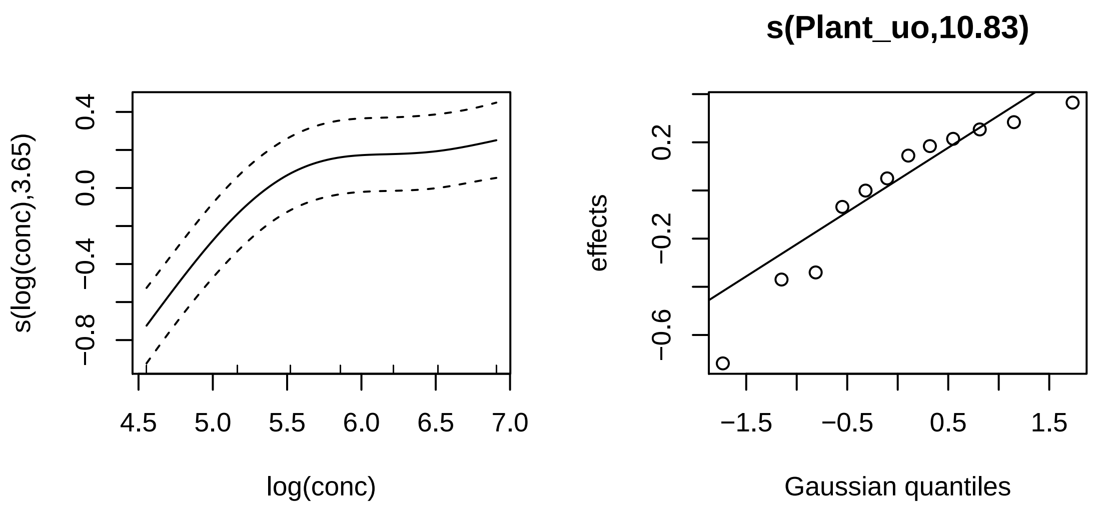

Figure 6: mgcv plotting output for model 1 applied to the CO2 dataset.

estimates effects vs Guassian quantiles, which can be used to check our model.

Looking at the effects by term is useful, but we are often interested in fitted values or predictions our models. Using the built in prediction functions with mgcv, we can estimate what the fitted function (and uncertainty around it) should look like for each level, as shown in Figure 7 (see appendix script for the code that generated this figure).

We can include interactions in an `s()` term via isotropic smooths such as TPRSs or we can use the tensor product (`te()`) function, if we don't believe the composite terms are isotropic. In this case `bs` and `k` can be specified as a single value (in which case each marginal smooth has the same basis or complexity) or as a vector of basis types or complexities. For example, `y ~ te(x1,x2, k=c(10,5), bs=c("tp","cs"))`, would specify a non-isotropic tensor product smooth of `x1` and `x2`, with the marginal basis for `x1` being a thin plate regression spline with 10 basis functions, and the smooth of `x2` being a cubic regression spline with 5 basis functions and a penalty on the null space.

For our bird example, we want to look at the interaction between location and time, so for this we setup the model as:

$$\mathbb{E}(\texttt{count}_i) = \exp(f(\texttt{week}_i, \texttt{latitude}_i))$$

where we assume that $\texttt{count}_i \sim$ Poisson. For the smooth term, f, we employ a tensor product of `latitude` and `week`, using a TPRS for the marginal latitude effects, and a cyclic cubic regression spline for the marginal week effect to account for the cyclic nature of weekly effects (we expect week 1 and week 52 to have very similar values), both splines had basis complexity (`k`) of 10. We will also assume the counts of individuals at each location in each week follow a Poisson distribution, and we will ignore species-specific variability.

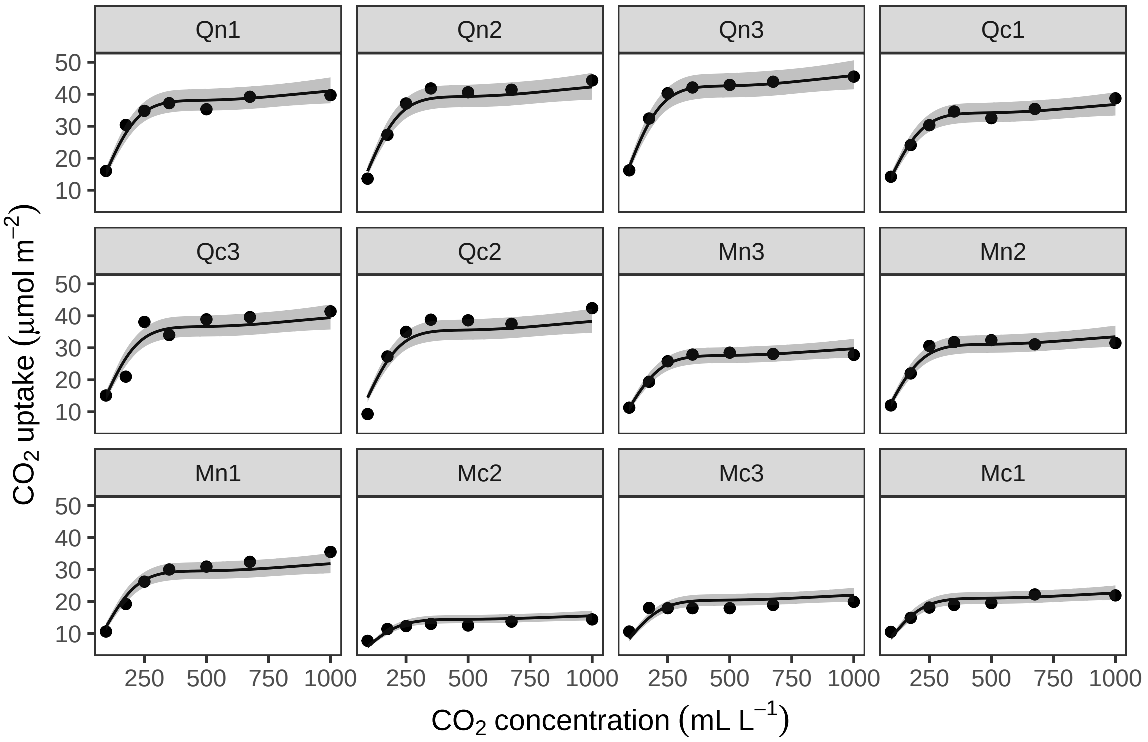

Figure 7: Predicted uptake function (± 2 s.e.) for each individual plant, based on model 1 (a single global function for uptake plus a individual-level random effect intercept). Model predictions are for log-uptake, but are inverse-log transformed here to show the fitted function on the original scale of the data.

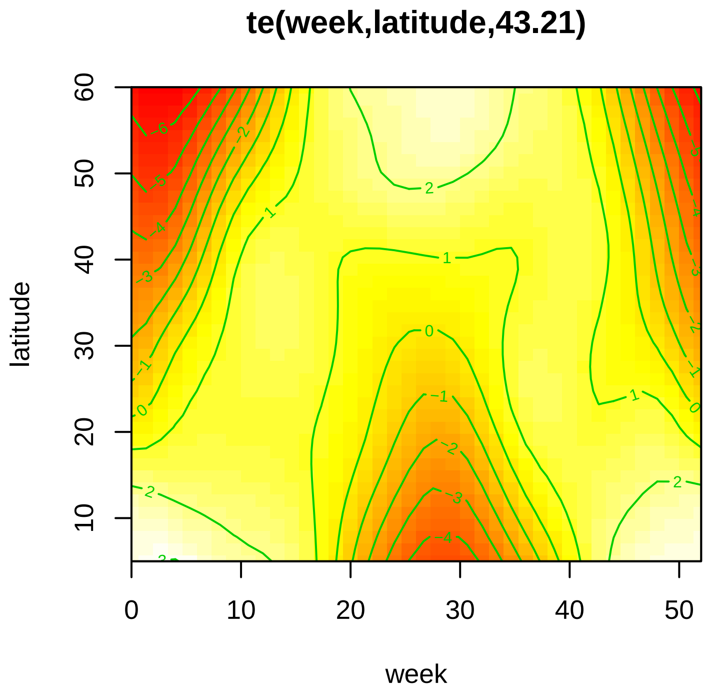

Figure 8: The default plot for this GAM illustrates the average log-abundance of all bird species at each latitude for each week, with yellow colours indicating more individuals and red colours fewer.

```
bird_mod1 <- gam(count ~ te(week, latitude, bs=c("cc", "tp"), k=c(10, 10)),
                 data=bird_move, method="REML", family=poisson,
                 knots = list(week = c(0.5, 52.5)))
```

Figure 8 shows the default plot (created by running `plot(bird_mod1, pages=1, scheme=2, rug=FALSE)`) for the week-by latitude smoother. It shows birds starting at low latitudes in the winter then migrating to high latitudes from the 10th to 20th week, staying there for 15-20 weeks, then migrating back. However, the plot also indicates a large amount of variability in the timing of migration. The source of this variability is apparent when we look at the timing of migration of each species (figure 5b).

All six species in figure 5b) show relatively precise migration patterns, but they differ in the timing of when they leave their winter grounds and the amount of time they spend at their summer grounds. Averaging over all of this variation results in a relatively imprecise (diffuse) average estimate of migration timing (figure 8, 9), and viewing species-specific plots of observed versus predicted values (figure 9), it is appearent that the model fits some of the species better than others. This model could potentially be improved by adding inter-group variation in migration timing. The rest of this section will focus on how to model this type of variation.

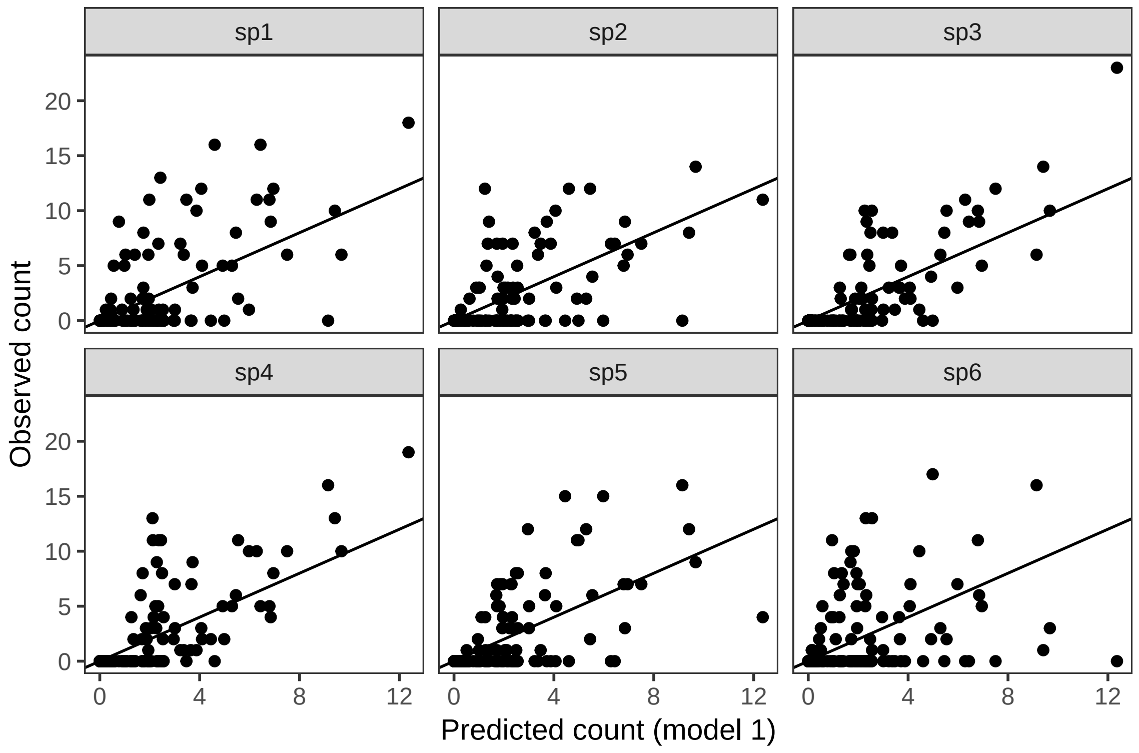

Figure 9: Observed counts by species versus predicted counts from `bird_mod1` (1-1 line added as reference). If our model fitted well we would expect that all species should show similiar patterns of dispersion around the 1-1 line (as we are assuming the data is Poisson, the variance around the mean should equal the mean). Instead we see that variance around the predicted value is much higher for species 1 and 6.

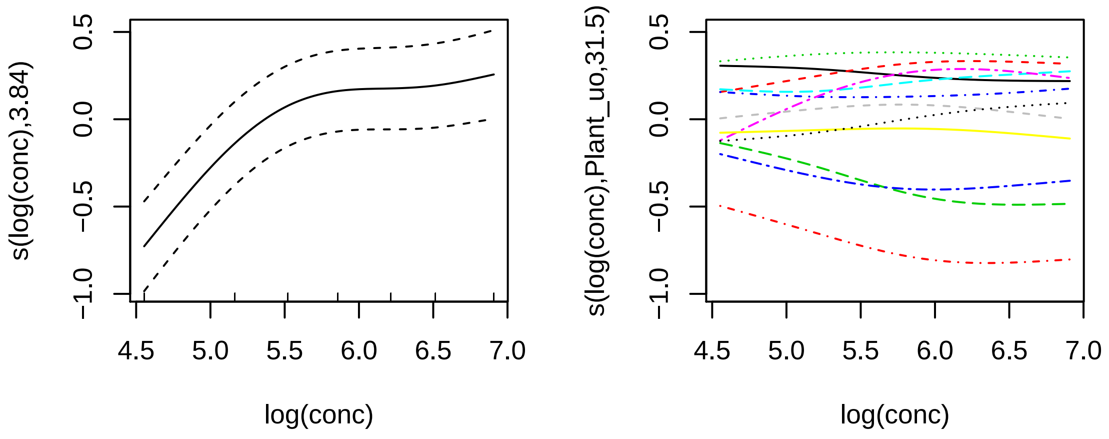

Figure 10: Global function (left) and group-specific deviations from the global function (right) for `CO2_mod2`

A single common smooth plus group-level smooths that have the same wigglyness (Model 2)

Model 2 is a close analogue to a GLMM with varying slopes: all groups have similar functional responses, but inter-group variation in responses is allowed. This approach works by allowing each grouping level to have its own functional response, but penalizing functions that are too far from the average.

This can be coded in *mgcv* by explicitly specifying one term for the global smooth (as in model 1 above) then adding a second smooth term specifying the group level smooth terms, using a penalty term that tends to draw these group-level smooths to zero. For one-dimensional smooths, *mgcv* provides an explicit basis type to do this, the factor smooth or `"fs"` basis (see `?smooth.construct.fs.smooth.spec` for detailed notes). This smoother creates a copy of each set of basis functions for each level of the grouping variable, but only estimates one set of smoothing parameters for all groups. The penalty is also set up so each component of its null space is given its own penalty (so that all components of the smooth are penalized towards zero)[7].

We modify our previous CO_2 model as follows:

$$\log_e(\texttt{uptake}_i) = f(\log_e(\texttt{conc}_i)) + f_{\texttt{Plant\_uo}_i}(\log_e(\texttt{conc}_i)) + \epsilon_i$$

where $f_{\texttt{Plant\_uo}_i}(\log_e(\texttt{conc}_i))$ is the smooth of concentration for the given plant. In R we then have:

[7]As part of the penalty construction, each group will also have its own intercept (part of the penalized null space), so there is no need to add a separate term for group specific intercepts as we did in model 1.

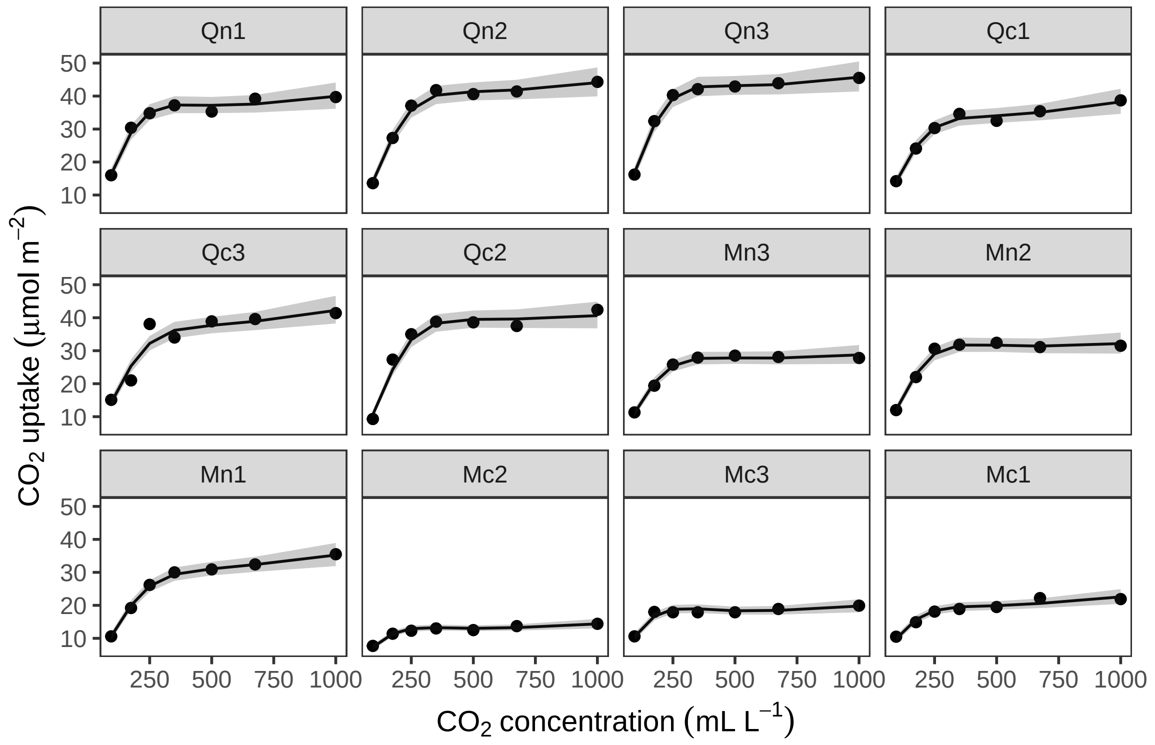

Figure 11: Predicted uptake values (lines) versus observed uptake for each plant, based on model 2.

Figure 10 shows the fitted smoothers for `CO2_mod2`. The plots of group-specific smooths indicate that plants differ not only in average log-uptake (which would correspond to each plant having a straight line at different levels for the group-level smooth), but differ slightly in the shape of their functional responses. Figure 11 shows how the global and group-specific smooths combine to predict uptake rates for individual plants:

The `"fs"`-based approach mentioned above does not work for higher-dimensional tensor product smooths (if one is willing to use thin plate regression splines for the multivariate smooth then one can use `"fs"`). Instead, the group-specific term can be specified with a tensor product of the continuous smooths and a random effect for the grouping parameter[8]. e.g.: `y ~ te(x1, x2, bs="tp", m=2) + t2(x1, x2, fac, bs=c("tp","tp","re"), m=2, full=TRUE)`. We illustrate this approach below on the bird migration data.

```
bird_mod2 <- gam(count ~ te(week, latitude, bs=c("cc", "tp"),
                              k=c(10, 10), m=c(2, 2)) +
                            t2(week, latitude, species, bs=c("cc", "tp", "re"),
                              k=c(10, 10, 6), m=c(2, 2, 2), full=TRUE),
                      data=bird_move, method="REML", family=poisson)
```

[8]As mentioned in section II, these terms can be specified either with `te()` or `t2()` terms. Using `t2` as above (with `full=TRUE`) is essentially a multivariate equivilent of the `fs` smooth; it requires more smooth terms than `te()`, but can be fit using other mixed effects software such as `lme4`, which is useful when fitting models with a large number of group levels (see Section IV for details).

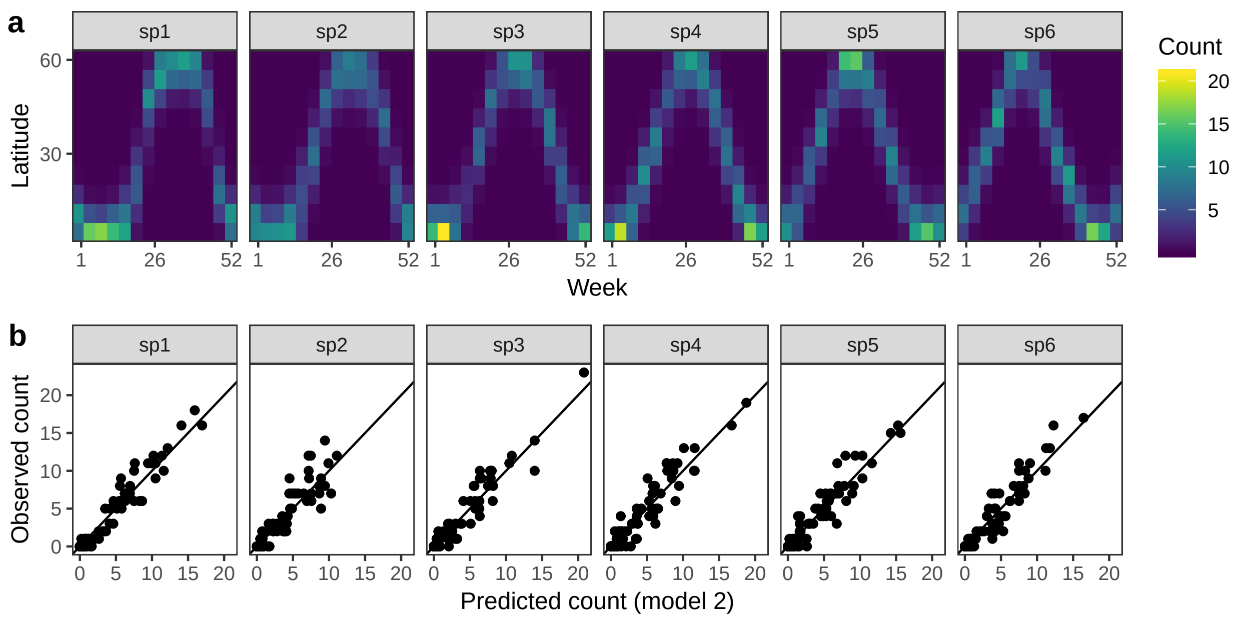

Figure 12: a) Predicted migration paths for each species based on `bird_mod2`, with lighter colors corresponding to higher predicted counts. b) Observed counts versus predictions from `bird_mod2`.

Model 2 is able to effectively capture the observed patterns of interspecific variation in migration behaviour (figure 12a), shows a much tigther fit between observed and predicted values, as well as less evidence of over-dispersion in some species compared to model 1 (figure 12b).

A single common smooth plus group-level smooths with differing wigglyness (Model 3)

This model class is very similar to model 2, but we now allow each group-specific smooth to have its own smoothing parameter and hence it's own level of wigglyness. This increases the computational cost of the model, and means that the only information shared between groups is through the global smoothing term. This is useful if different groups differ substantially in how variable they are.

Fitting a seperate smooth term (with its own penalties) can be done in *mgcv* by using the `by` argument in the `s()` and `te()` (and related) functions. Therefore, we can code this model as: `y ~ s(x,bs="tp") + s(x, by=fac, m=1, bs="ts") + s(fac, bs="re")`. Note three major differences from how model 2 was specified:

1. We explicitly include a random effect for the intercept (the `bs="re"` term), as group-specific intercepts are not incorporated into factor `by` variable smooths (as would be

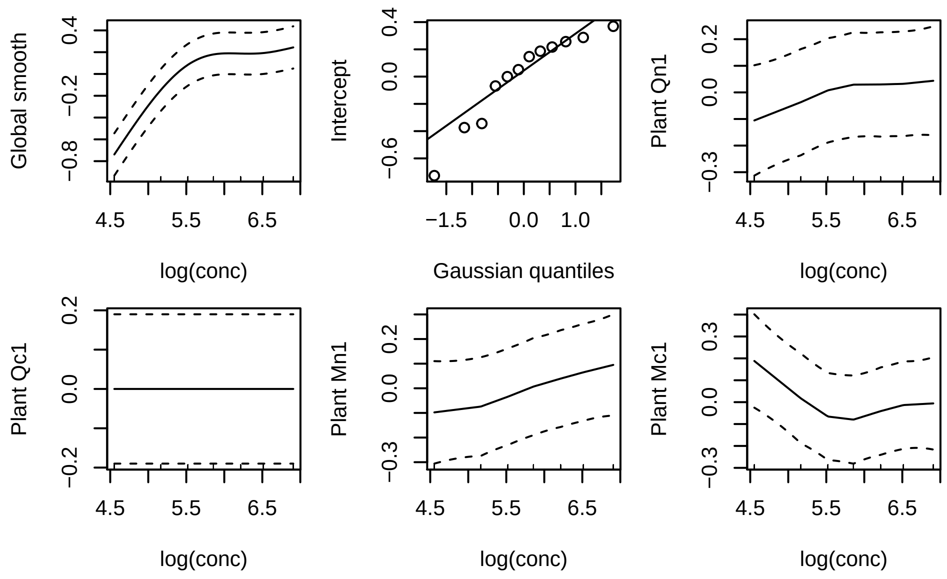

Figure 13: Functional relationships for the CO2 data estimated for model 3. Top left: the global smooth; Top middle: species-specific random effect intercepts. The remaining plots are a selected subset of the plant-specific smoothers, indicating how the functional response of that plant differs from the global smooth.

the case with `bs="fs"` or a tensor product random effect).

2. We explicitly use a basis with a fully penalized null space for the group-level smooth (`bs="ts"`, for TPRS with shrinkage), as this method does not automatically penalize the null space, so there is potential for co-linearity issues between unpenalized components of the global and group-level smoothers
3. We specific m=1 instead of m=2 for the groupwise smooths, which means the marginal TPRS basis for this term will penalize the squared 1st derivative of the function, rather than the second derivative. We do this as there can be issues of co-linearity between the global smooth term and the group-specific terms which occasionally leads to high uncertainty around the global smooth (see section V for more details). TPRS with m=1 have a more restricted null space than m=2 smoothers, so should not be as colinear with the global smooth (Baayen et al. 2016; Wieling et al. 2016). We have observed that this is much more of an issue when fitting model 3, compared to model 2.

Our `CO2` model is then modified as follows:

```
CO2_mod3 <- gam(log(uptake) ~ s(log(conc), k=5, m=2, bs="tp") +
                              s(log(conc), by=Plant_uo, k=5, m=1, bs="ts") +
                              s(Plant_uo, bs="re", k=12),
                data=CO2, method="REML")
```

Figure 13 shows a subsample of the group-specific smooths from this model, to prevent crowding. It is appearent from this that some groups (e.g. `Qc1`) have very similar shapes to the global smooth (differing only in intercept), others do differ from the global trend, with higher uptake at low concentrations and lower uptake at higher concentrations (e.g. `Mc1`, Qn1), or the reverse pattern (e.g. Mn1).

Using model 3 with higher-dimensional data is also straightforward; by terms work just as well in tensor-product smooths as they do with isotrophic smooths. We can see this with our bird model:

```
bird_mod3 <- gam(count ~ te(week, latitude, bs=c("cc", "tp"),
                            k=c(10, 10), m=c(2, 2)) +
                         te(week, latitude, by=species, bs= c("cc", "ts"),
                            k=c(10, 10), m=c(1, 1)),
                 data=bird_move, method="REML", family=poisson)
```

Note here we used a "ts" smooth for the latitude marginal effect; this is a TPRS smooth with the penalty matrix slightly tweaked so that the null space is also penalized. This is to prevent the null space of the global smoother being colinear with the null spaces of the groupwise smoothers (see section IV for more discussion about the issue of colinearity and smoother selection).

The fitted model for bird_mod3 is visually indistinguishable from bird_mod2 (figure 12) so we do not illustrate it here.

Models without global smooth terms (models 4 and 5)

We can modify the above models to exclude the global term (which is generally faster; see section V). When we don't model the global term, we are allowing each factor to be different, though there may be some similarities in the shape of the functions.

Model 4:

Model 4 (shared smooths) is simply model 2 without the global smooth term: y~s(x,fac,bs="fs") or y~te(x1,x2,fac,bs=c("tp","tp","re"). This model assumes all groups have the same smoothness, but that the individual shapes of the smooth terms are not related. Here we just show how to code these models; plotting them works in the same was as for models 1-3 above, the plots for these datasets are very similar to the plots for model 2.

```
CO2_mod4 <- gam(log(uptake) ~ s(log(conc), Plant_uo, k=5,  bs="fs", m=2),
                data=CO2, method="REML")

bird_mod4 <- gam(count ~ t2(week, latitude, species, bs=c("cc", "tp", "re"),
                            k=c(10, 10, 6), m = c(2,2,2)),
                 data=bird_move, method="REML", family=poisson)
```

Model 5:

Model 5 is simply model 3 without the first term: `y~s(x,by=fac)` or `y~te(x1,x2, by=fac)`. (Plots are very similar to model 3.)

```r
CO2_mod5 <- gam(log(uptake) ~ s(log(conc), by=Plant_uo, k=5, bs="tp", m=2) +
                              s(Plant_uo, bs="re", k=12),
                data= CO2, method="REML")


bird_mod5 <- gam(count ~ te(week, latitude, by=species, bs= c("cc", "ts"),
                            k=c(10, 10), m=c(2,2)),
                 data=bird_move, method="REML", family=poisson)
```

Comparing different HGAM specifications

These models can be compared using standard model comparison tools. Model 2 and model 3 will generally be nested in model 1 (depending on how each model is specified) so ANOVA comparisons may be used to test if the groupwise smoother is necessary. However, we do not currently recommend this method. Given the uncertainty about what degrees of freedom to assign to models with varying smooths, and the fact that slightly different model specifications may not result in nested models, we do not think there is sufficient theory on how accurate parametric p-values will be for comparing these models (see `?mgcv::anova.gam` for more discussion on ANOVA comparisons for GAMs).

Comparing models based on AIC is a more robust approach to comparing the different modelling approaches, as there is a well-developed theory of how to include effects of penalization and smoothing parameter uncertainty when estimating the model complexity penalty for AIC (Wood, Pya, and Säfken 2016). We demonstrate this approach in Table 1. Going by the AIC criteria, there is strong support for including among-group functional variability for both the CO2 dataset and the bird_move dataset (compare models 1 versus models 2-5). For the CO2 dataset (Table 1A), there is relatively strong evidence that there is more intergroup variability in smoothness than model 2 allows, and weaker evidence that model 4 or 5 (separate smooths for all plants) show the best fit. For the bird_move dataset (Table 1B), model 2 (global smooth plus group-level smooths with a shared penalty) fits the data best, which is good as that is how we simulated the data!

It is important to recognize that AIC, like any function of the data, is a random variable and should be expected to have some sampling error (Forster and Sober 2011). In cases when the goal is to select the model that has the best predictive ability, we recommend holding some fraction of the data out prior to the analysis and comparing how well different models fit that data or using k-fold cross validation as a more accurate guide to how well a given model may predict out of sample. We also strongly recommend that models are not selected based purely on AIC; instead model selection should be based on biological and ecological knowledge about the system, the goals of the study, computational time, and what outputs the user wants. For instance, while model 3 may fit a given dataset better than model 2, model 2 can be used to simulate functional variation for unobserved group levels, whereas

Table 1: AIC table comparing model fits for example datasets

Model	df	AIC
A. CO2 models		
CO2_mod1	17	-119
CO2_mod2	39	-199
CO2_mod3	42	-216
CO2_mod4	53	-219
CO2_mod5	56	-220
B. bird_move models		
bird_mod1	46	3444
bird_mod2	140	1535
bird_mod3	244	1677
bird_mod4	143	1543
bird_mod5	197	1599

this is not possible within the framework of model 3. The next section discusses these and other model fitting issues in depth.

IV: Computational and statistical issues when fitting HGAMs

Which of the five models should you choose for a given data set? There are two major trade-offs to take into account. The first is the bias-variance trade-off: more complex models can account for more fluctuations in the data, but also tend to give more variable predictions, and can overfit. The second tradeoff is model complexity versus computer time: more complex models can include more potential sources of variation and give more information about a given data set, but will generally take more time and computational resources to fit and debug. We will discuss both of these trade-offs in this section.

Bias-variance tradeoffs

The bias-variance tradeoff is a fundamental concept in classical statistical analysis. When trying to estimate any value (in the cases we are focusing on, a smooth functional relationship between predictors and data), bias measures how on average an estimate is from the true value of the thing we are trying to estimate, and the variance of an estimator corresponds to how much that estimator would fluctuate if applied to multiple different samples taken from the same population. These two properties tend to be traded off when fitting models; for instance, rather than estimating a population mean from data, we could simply use a fixed value regardless of the observed data. This estimate would have no variance (as it is always the same) but would have high bias unless the true population mean happened to equal zero.[9] The core insight into why penalization is useful is that the penalty term slightly increases the bias but can substantially decrease the variance of an estimator, relative to its unpenalized version (Efron and Morris 1977).

In GAMs and HGLMs, the bias-variance tradeoff is managed by the penalty terms (random effect variances in HGLM terminology). Larger penalties correspond to lower variance, as the estimated function is unable to wiggle a great deal, but also correspond to higher bias unless the true function is close to the null space for a given smoother (e.g. a straight line for thin-plate splines with 2nd derivative penalties, or zero for a standard random effect). The computational machinery used by mgcv to fit smooth terms is designed to find penalty terms that best trade off bias for variance to find a smooth that can effectively to predict new data.

The bias–variance tradeoff comes into play with HGAMs when choosing whether to fit separate penalties for each group level or assign a common penalty for all group levels (i.e. deciding between models 2 & 3 or models 4 & 5). If the functional relationships we are trying to estimate for different group levels actually vary in how wiggly they are, setting the penalty for all group-level smooths equal (models 2&4) will either lead to overly variable estimates for the least variable group levels, overly smoothed (biased) estimates for the most wiggly terms, or a mixture of these two, depending on how the fitting criteria used (ML, REML, or GCV) determines where the optimal smoothing parameter should be set.

We developed a simple numerical experiment to determine whether mgcv fitting criteria tend to set estimated smoothness penalties high or low in the presence of among-group variability in smoothness. We simulated data from five different groups, with all groups having the same levels of the covariate x, ranging from 0 to 2π. For each group, the true function relating x to y was a sine wave, but the frequency varied from 0.25 (equal to half a cycle across the range of x) to 4 (corresponding to 4 full cycles across the range). We added normally distributed error to all y-values, with a standard deviation of 0.2. We then fit both model 4 (where all curves were assumed to be equally smooth) and model 5 (with varying smoothness) to the entire data set, using REML criteria to estimate penalties. For this example (Fig. 14a), requiring equal smoothness for all group levels resulted mgcv underestimating the penalty for the lowest frequency (most smooth) terms, but accurately estimating the true smoothness of the highest frequency terms as measured by the squared second derivative of the smooth fit versus that of the true function (Fig. 14b).

This implies that assuming equal smoothness will tend to lead to underestimating the true smoothness of low-variability terms, and thus leading to more variable estimates of these terms. If this is a potential issue, we recommend fitting both models and using standard model evaluation criteria (e.g. AIC) to determine if there is evidence for among-group variability in smoothness. For instance, the AIC for model 4 fit to this data is -178, whereas it is -211 for model 5, implying a substantial improvement in fit by allowing smoothness to vary. However, it may be the case that there are too few data points per group to estimate seperate smoothness levels, in which case model 2 or model 4 may still be the better option even in

[9]While this example may seem contrived, this is exactly what happens when we assume a given fixed effect is equal to zero (and thus exclude it from a model).

the face of varying smoothness.

The ideal case would be to assume that among group penalties follow their own distribution (estimated from the data), to allow variation in smoothness while still getting the benefit of pooling information on smoothness between groups. However, this is currently not implemented in mgcv (and would be difficult to set up via mgcv's method of structuring penalties). It is possible to set up this type of varying penalty model in flexible Bayesian modelling software such as Stan or INLA (see below for a discussion of these tools), but how to set up this type of model has not been well studied, and is beyond the scope of this paper.

It may seem like there is also a bias–variance tradeoff between choosing to use a single global smoother (model 1) or a global smoother plus group-level terms (model 2-3), as in model 1, all the data is used to estimate a single smooth term, and thus should have lower variance than models 2-3, but higher bias for any given group in the presence of inter-group functional variability. However, in practice, this tradeoff will already be handled by mgcv via estimating penalties; if there are no average differences between functional responses, mgcv will penalize the group–specific functions toward zero, and thus toward the global model. The choice between using model 1 versus models 2-3 should generally be driven by computational costs; model 1 is typically much faster to fit than models 2-3, even in the absence of among–group differences, so if there is no need to estimate inter-group variability, model 1 will typically be more efficient.

A similar issue exists when choosing between models 2/3 and 4/5; if all group levels have very different functional shapes, the global term will get penalized toward zero in models 2/3, so they will reduce to models 4/5. Again, the choice to include a global term or not should be made based on scientific considerations (is the global term of interest to estimate) and computational considerations (which we will discuss next).

Complexity – computation tradeoffs

GAMs and GLMMs have substantially increased the range of flexible models available to the average researcher, and the HGAM models we discussed in section III extend on this broad base. However, the more flexible a model is, the larger an effective parameter space any fitting software has to search to find parameters that can predict the observed data. While numerical algorithms for solving complex models are always improving, it can still be surprisingly easy to use up massive computational resources trying to fit a model to even relatively small datasets. While we typically want to choose a model based on model fit (see above and our goals for what the model will be used for, computing resources can often act as an effective upper limit on possible model complexity. Fitting an HGAM means adding extra computational complexity on top of either a GAM model with only global terms or a GLMM without smooth terms. For a given data set (with a fixed number `n` data points) and assuming a fixed family and link function, the time it takes to compute a given HGAM will depend, roughly, on four factors: the number of basis functions to be estimated, the number of smooth penalties to be estimated, whether the model needs to estimate both a global smooth and groupwise smooths, and the algorithm used to estimate parameters and

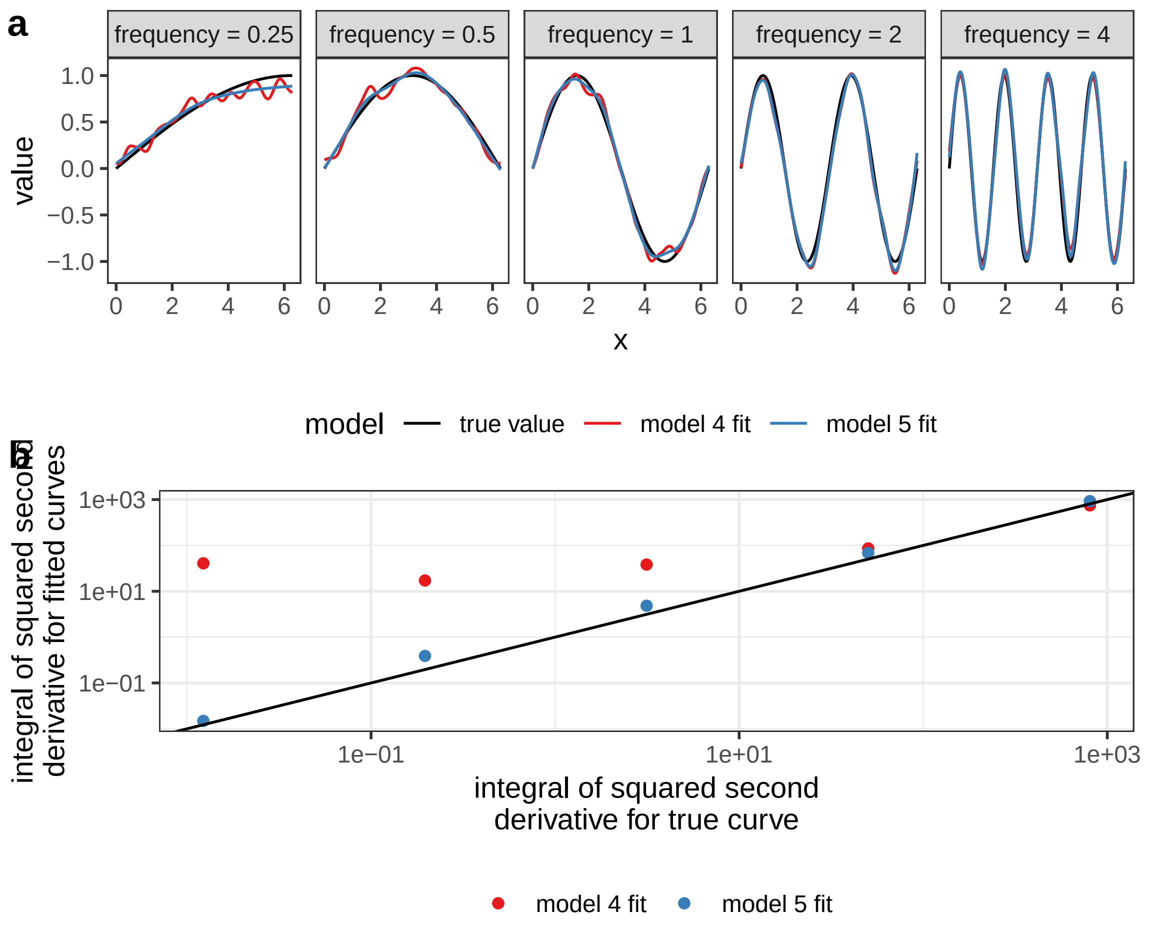

Figure 14: a) This figure demonstrates the bias that can arise from assuming equal smoothness for all group levels (model 4, red line) versus allowing for intergroup variation in smoothness (model 5, red line) when the true function (black line) shows substantial variation in smoothness between groups. b) Integrated squared 2nd derivative of the true function for each group level versus that for the functions estimated by model 4 (red) and model 5 (blue), indicating substantial undersmoothing for low-variability curves by model 4.

fitting criteria used.

The most straightforward factor that will affect the amount of computational resources is the number of parameters in the model. Adding group-level smooths (moving from model 1 to 2-5) means that there will be more regression parameters to estimate, since each grouping level needs a separate coefficient for each basis function in the smooth. For a dataset with `g` different groups and `n` data points, fitting a model will just a global smooth, `y~s(x,k=k)` will require only `k` coefficients, and takes $\mathcal{O}(nk^2)$ operations[10] to evaluate, but fitting the same data using a group-level smooth (model 4, `y~s(x,fac,bs="fs",k=k)`) will require $\mathcal{O}(nk^2g^2)$ operations to evaluate; in effect, adding a group-level smooth will increase computational time by an order of the number of groups squared[11]. The effect of this is visible in the examples we fit in section III when comparing the number of coefficients and relative time it takes to compute model 1 versus the other models (Table 2). One way to deal with this issue would be to reduce the number of basis functions (`k`) used when fitting group-level smooths when the number of groups is large; in effect, this would increase the flexibility of the model to accommodate inter-group differences, while reducing its ability to model variance within any given group. It can also make sense to use more computationally efficient basis functions when fitting large data sets, such as p-splines or cubic splines, rather than thin-plate splines, as thin-plate splines can take a substantial amount of overhead to compute the actual basis functions to use (Wood 2017).

Adding additional smoothing parameters (moving from model 2 to model 3, or moving from model 4 to 5) is even more costly than increasing the number of coefficients to estimate, as estimating smoothing parameters is computationally intensive (Wood 2011). This means that models 2 and 4 will generally be substantially faster than 3 and 5 when the number of groups is reasonably large, as models 3 and 5 fit a separate set of penalties for each group level. The effect of this is visible in comparing the time it takes to fit model 2 to model 3 (which has a smooth for each group) or models 4 and 5 for the example data (Table 2). Note that this will not hold for every model, though; for instance, model 5 takes less time to fit the bird movement data than model 4 does (Table 2B).

Alternative formulations: bam, gamm, and gamm4

When fitting models with large numbers of different group levels, it is often possible to speed up computation substantially by using one of the alternative fitting algorithms available through mgcv.

[10]To understand the effects of these terms, we will use "big-O" notation; when we say a given computation is of order $\mathcal{O}(n \log n)$, it means that, for that computation, as n gets large, the amount of time the computation will take will grown proportionally to $n \log n$, so more quickly than linearly with n, but not as fast as n squared.

[11]Including a global smooth (models 2-3) or not (models 4-5) will not generally substantially affect the number of coefficients needed to estimate (compare the number of coefficients in Table 2, model 2 vs. model 4, or model 3 versus model 5). Adding a global term will only add at most `k` extra terms, and it actually ends up being less that that, as `mgcv` drops basis functions from co-linear smooths to ensure that the model matrix is full rank.

Table 2: Relative computational time and model complexity for different HGAM formulations of the two example data sets from section III. All times are scaled relative to the length of time model 1 takes to fit to that data set. The # of coefficients measures the total number of model parameters (including intercepts). The # of smooths is the total number of unique penalty values estimated by the model.

		# of terms	
model	relative time	coefficients	penalties
A. CO2 data			
1	1	17	2
2	5	65	3
3	14	65	14
4	4	61	3
5	9	61	13
B. bird movement data			
1	1	90	2
2	110	540	5
3	150	624	14
4	110	541	3
5	67	535	12

The first tool available, that requires the least changes to your code compared to the base `gam` function, is the `bam` function. This function is designed to improve performance when fitting data sets with large amounts of data. It uses two tools to do this. First, it saves on the amount of memory needed to compute a given model by using a random subset of the data to calculate the basis functions for the smoothers, then breaking the data up into blocks and updating model fit within each block (Wood, Goude, and Shaw 2015). While this is primarily designed to reduce the amount of memory needed to fit these models, it can also substantially reduce computation time. Second, the `bam` function, when fitting using its default “fREML” (for “Fast REML”) method, you can use the `discrete` when fitting the model. This option causes bam to simplify each covariate to a set of discrete levels (instead of a continuous range), substantially reducing the amount of computation needed. Setting “discrete = TRUE” lets `bam` estimate the number of bins to use for each covariate. It is also possible to manually specify the number of bins by passing `discrete` a vector of values. See `?mgcv::bam` for more details.

It also takes more computational overhead compared to `gam` to set a `bam` model up, so for small numbers of groups, it can actually be slower than `gam` (Figure 15), however, as the number of groups increases, computational time for `bam` increases more slowly than for `gam`; in our simulation tests, when the number of groups is greater than 16, `bam` can be upward of an order of magnitude faster (Figure 15). Note that `bam` can be somewhat less computationally stable when estimating these models (i.e. less likely to converge) so it does typically make sense to still use `gam` for smaller data sets.

The second option is to fit these models using one of two dedicated mixed effect model estimation packages, `nlme` and `lme4`. The `mgcv` package includes the function `gamm` that allows you to call `nlme` to solve a given GAM, automatically handling the transformation of smooth terms into random effects (and back into basis-function representations for plotting and other statistical analyses). To use `lme4`, you will have to install the `gamm4` package, and use the `gamm4` function from this package. Using `gamm` or `gamm4` to fit models rather than `gam` can substantially speed up computation when the number of groups is large, as both `nlme` and `lme4` take advantage of the sparse structure of the random effects, where most basis functions will be zero for most groups (i.e. any group-specific basis function will only take a non-zero value for observations in that group level). As with `bam`, `gamm` and `gamm4` are generally slower than `gam` for fitting HGAMs when the number of group levels is small (in our simulations, <8 group levels), however they do show substantial speed improvements even with a moderate number of groups, and were as fast as or faster to calculate than *bam* for all numbers of grouping levels we tested (Figure 15)[12]. For large numbers of group levels and bigger data sets, it may be necessary to use full functional regression methods such as those implemented in the `refund` package (Scheipl, Staicu, and Greven 2014). See below for a discussion of what functional regression is and its connections to HGAMs.

Setting up models 1-5 in `bam` uses the same code as we have previously covered; the only difference is that you use the `bam` instead of `gam` function, and have the additional option of discretizing your covariates. The advantage of this approach is that `bam` allows you to use almost all of the same families available to the `gam` function, and `bam` model output can be evaluated using the same functions (e.g. `summary`, `AIC`, `plot`, etc.) so it is simple to substitute for `gam` if you need to speed a model up.

Both `gamm` and `gamm4` require at least a few changes to how you code models. First, there are a few limitations on how you are able to specify models 1-5 in both frameworks. Factor smooth (`bs="fs"`) basis setups work in both `gamm` and `gamm4`. However, as the `nlme` package does not support crossed random effects, it is not possible to have two "fs" terms for the same grouping variable in `gamm` models (e.g. `y~s(x1, grp,bs="fs"+s(x2, grp, bs="fs")`. These type of crossed random effects are allowed in `gamm4`. The use of `te` and `ti` terms are not possible in `gamm4` however, due to issues with how random effects are specified in the `lme4` package, making it impossible to code models where multiple penalties apply to a single basis function. Instead, for multidimensional group-level smooths, the alternate function `t2` needs to be used to generate these terms, as it creates tensor products with only a single penalty for each basis function (see `?mgcv::t2` for details on these smoothers, and Wood, Scheipl, and Faraway (2013) for the theoretical basis behind this type of tensor product). So for instance, model 2 for the bird movement data we discussed in section III would need to be coded as:

```
bird_mod4_gamm4 = gamm4(count ~ t2(week,latitude,species,
                                    bs= c("cc", "tp","re"),
```

[12]It is also possible to speed up both `gam` and `bam` by using multiple processors in parallel, whereas this is not currently possible for `gamm` and `gamm4`. For large numbers of grouping levels, this should speed up computation as well, at the cost of using more memory. However, computation time will likely not decline linearly with the number of cores used, since not all model fitting sets are parallizable, and performance of the cores can vary. As parallel processing can be complicated and dependent on the type of computer you are using to configure properly, we do not go into how to use these methods here. The help file `?mgcv::mgcv.parallel` explains how to use parallel computations for `gam` and `bam` in detail.

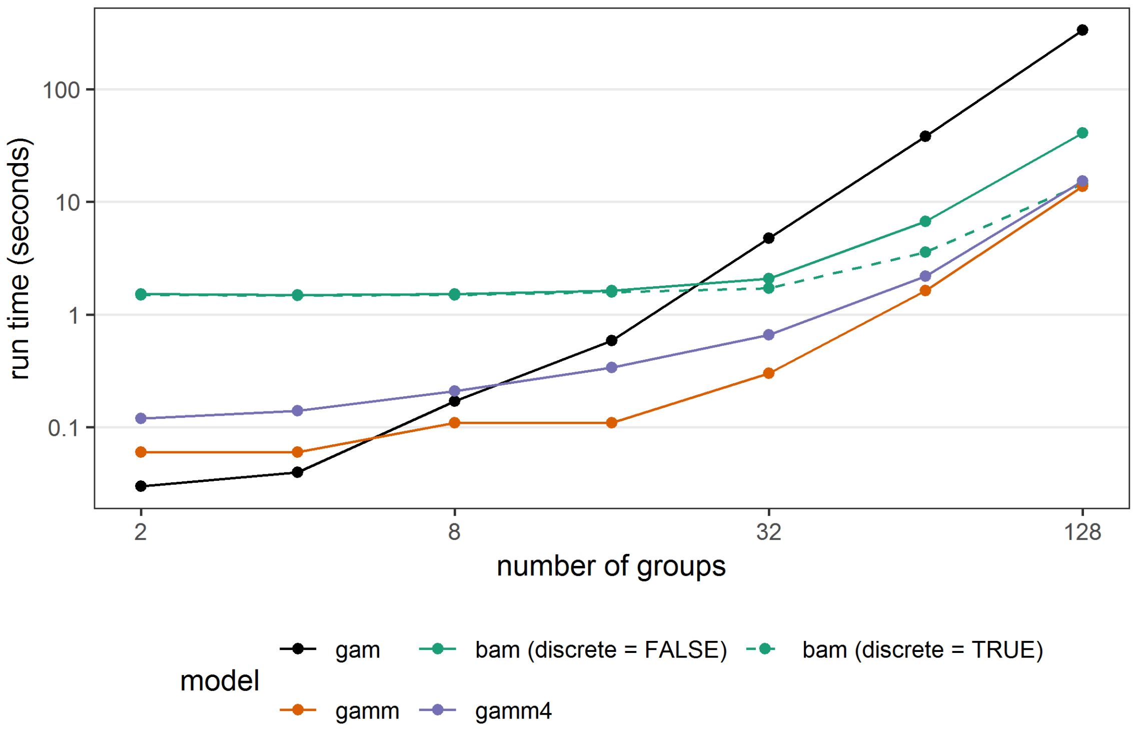

Figure 15: Elapsed time to estimate the same model using each of the four approaches. Each data set was generated with 20 observations per group using a unimodal global function and random group-specific functions consisting of an intercept, a quadratic term, and logistic trend for each group. Observation error was normally distributed. Models were fit using model 2: y~s(x, k=10, bs='cp') + s(x,fac, k=10, bs='fs', xt=list(bs='cp'), m=1). All models were run on a single core.

```
                          k=c(10,10,6),
                          m = 2),
             data= bird_move, family= poisson)
```

These packages also do not support the same range of families for the dependent variable; `gamm` only supports non-Gaussian families by using a fitting method called penalized quasi-likelihood (PQL) that is slower and not as numerically stable as the methods used in `gam`, `bam`, and `gamm4`. Non-Gaussian families are well supported by `lme4` (and thus `gamm4`), but can only fit them using marginal likelihood (ML) rather than REML, so may tend to over-smooth relative to `gam` using REML estimation. Further, neither `gamm` nor `gamm4` supports several of the extended families available through `mgcv`, such as zero-inflated, negative binomial, or ordered categorical and multinomial distributions.

Estimation issues when fitting both global and groupwise smooths

When fitting models with separate global and groupwise smooths (models 2 and 3), one issue to be aware of is concurvity between the global smooth and groupwise terms. Concurvity measures how well one smooth term can be approximated by some combination of the other smooth terms in the model (see `?mgcv::concurvity` for details). For models 2 and 3, the global term is entirely concurve with the groupwise smooths. This is because, in the absence of the global smooth term, it would be possible to recreate that average effect by shifting all the groupwise smooths so they were centered around the global mean. In practical terms, this has the consequence of increasing uncertainty around the global mean relative to a model with only a global smooth. In some cases, it can result in the estimated global smooth being close to flat, even in simulated examples with a known strong global effect. This concurvity issue may also increase the time it takes to fit these models (for example, compare the time it takes to fit models 3 and 5 in Table 2). That these models can still be estimated is because of the penalty terms; all of the methods we have discussed for fitting model 2 ("fs" terms or random effect tensor products) automatically create a penalty for the null space of the group-level terms, so that only the global term has its own unpenalized null space, and both the REML and ML criteria work to balance penalties between nested smooth terms (this is why nested random effects can be fitted). We have noted, however, that `mgcv` still occasionally finds degenerate solutions with simulated data where the fitted global term ends up over-smoothed.

What we recommend to avoid this issue is to use a combination of smoother choice and setting model degrees of freedom so that the groupwise terms are either slightly less flexible or have a smaller null space. For instance, in the examples in section III, we used smoothers with an unpenalized null space (standard thin-plate splines) for the global smooth and ones with no null space for the groupwise terms[13]. When using thin-plate splines, it may also help to use splines with a lower order of derivative penalized in the groupwise smooths than the global smooths, as lower-order "tp" splines have fewer basis functions in the null space. For

[13]For model 2, the "fs" smoother, and tensor products of random effect ("re") and other smooth terms do not have a penalized null space by construction (they are full rank), as noted above. For model 3 groupwise terms, we used basis types that had a penalty added to the null space: bs="tp", "cs", or "ps" have this property.

example, we used `m=2` (penalizing squared second derivatives) for the global smooth, and `m=1` (penalizing squared first derivatives) for groupwise smooths in models 2 and 3. Another option would be to use a lower number of basis functions (`k`) for groupwise relative to global terms, as this will reduce the maximum flexibility possible in the groupwise terms. We do caution that these are just rules of thumb. As of this writing, there is no published work looking what the effect of adding groupwise smooths has on the statistical properties of estimating a global smooth. In cases where an accurately estimated global smooth is essential, we recommend either fitting model 1, or using Markov Random Fields (Appendix A) and calculate the global smooth by averaging across grouping levels.

A brief foray into the land of Bayes

So far we have focused on uisng HGAMs in a frequentist framework, where the goal is to estimate single curves that best approximate some underlying true curve. However, both HGLMs and GAMs, and consquently HGAMs, fit comfortably into a Bayesian regression framework, where the goal of inference is to quantify uncertainty from all model parameters, as well as incorporate prior information on model parameters into the estimation process. By moving to a Bayesian framework, it is possible to extend HGAMs to model a wider range of types of data, include background information

The key connection here is, as we mentioned in section II, that the penalty matrix can also be treated as the inverse of a prior covariance matrix for model parameters β. REML estimation in mgcv then can be viewed as an empirical Bayesian estimate of the smooth model, where any terms in the null space of the smooth are treated as fixed effects with an improper flat prior (i.e. any value for these terms are considered equally likely), any terms in the range space are treated as having a multivariate normal distribution, and the penalty terms are treated as having an improper flat prior (see Wood (2017) for more details on this connection). The Bayesian covariance matrix for model parameters can be extracted from any fitting `gam`/`bam` model with `model$Vp` or `vcov(model)`. This can in turn be used to generate predictions from the posterior distribution of the model, as the Bayesian covariance matrix already incorporates the uncertainty from having to estimate the covariance matrix into it (the standard confidence intervals used in mgcv are in fact Bayesian posterior CIs; Simon N Wood 2006a). See `?predict.gam` for more on how to simulate empirical Bayes posterior estimates.

This also means that HGAMs can also be included as components in a more complex fully Bayesian model. The `mgcv` package includes a function `jagam` that can take a specied model formula and automatically convert it into code for the JAGS (or BUGS) Bayesian statistical packages, which can be adapted by the user to their own needs. Similiarly, the `brms` package (Bürkner 2017), which can fit complex statistical models using the Bayesian software stan (Carpenter et al. 2017) allows for the inclusion of smooth terms as part of the model specification. The `brms` package does not currently support `te()` tensor products or factor-smooth basis terms (`bs="fs"`), but does support `t2`-style tensor products, which means all of the models fitted in this paper can be fit by `brms`. For instance, `CO2_mod2` could be coded like this:

```r
library(brms)

CO2_mod2_brms <- brm(
  bf(log(uptake) ~
       s(log(conc), k=5, m=2) +
       t2(log(conc), Plant_uo, k=c(5,12),
                                  bs=c("tp","re"), m=2, full =TRUE)),
  data = CO2,
  family = gaussian(),
  chains = 2,
  control = list(adapt_delta = 0.95)
)
```

Which uses two chains of stan's Hamiltonian Monte Carlo sampler to generate estimates from the model posterior. model fits can be plotted from this as:

```r
plot(marginal_effects(CO2_mod2_brms), points=TRUE, ask=FALSE, plot=TRUE)
```

and the raw stan code used to estimate the model (which can be then modified) can be extracted with:

```r
stancode(CO2_mod2_brms)
```

should we mention gretagam at this point? and does INLAbru have any capacity for HGAMs? I'm not familiar enough with INLAbru's inner workings. I do know that glmmTMB cannot currently include mgcv-style smooths.

Beyond HGAMs: functional regression

The HGAM models we have discussed are actually a type of *functional regression*, which is an extension of standard regression models to cases where the outcome variable y_i and/or the predictor variables x_i for a given outcome are functions, rather than single variables (Ramsay and Silverman 2005). HGAMs as we have described them are a form of function-on-scalar regression (Ramsay and Silverman 2005; Reiss, Huang, and Mennes 2010), where we are trying to estimate a smooth function that varies between grouping levels.

We have deliberately focused our paper on these simplier classes of functional regression model, and chosen to use the term HGAM rather than functional regression, as we believe that this more clearly connects these models to modelling approaches already familiar to ecologists. Further, we consider the unit of analysis to still be individual observations, as compared to functional regression where the the unit of analysis is whole functions (for instance, we are interested in cases like species distribution modelling, where the presence of a given species may be predicted from a sum of several species-specific functions of different environmental variables). However, there is an extensive literature dedicated to how to fit more complex functional regression models for any interested reader (see Ramsay and Silverman (2005) for a good introduction, and Scheipl, Gertheiss, and Greven (2016) for more recent work in this field). The `refund` package (Reiss, Huang, and Mennes 2010; Scheipl, Staicu, and Greven 2014; Scheipl, Gertheiss, and Greven 2016) uses the statistical machinery from `mgcv` to fit these models, and should be usable by anyone familiar with standard R and `mgcv` modelling syntax.

V: Examples

In this final section, we will go through two worked examples to highlight how to use these models in practice, and to illustrate how to fit, test, and visualize each model. We will demonstrate how to use these models to fit community data, to show when using a global trend may or may not be justified, and to illustrate how to use these models to fit seasonal time series.

Here, we are using data from the Wisconsin Department of Natural Resources collected by Richard Lathrop from a chain of lakes (Mendota, Menona, Kegnonsa, and Waubesa) in Wisconsin, to study long-term patterns in the seasonal dynamics of zooplankton. This data consists of roughly bi-weekly samples (during open-water conditions) of the zooplankton communities, taken from the deepest point of each lake via vertical tow collected every year from 1976 to 1994 (the collection and processing of this data is fully described in Lathrop (2000)). We will use this data estimate variability in seasonality among species in the community, and between lakes for the most abundant taxon in the sample (*Daphnia mendotae*). As we are focusing on seasonal cycles rather than average or maximum abudances, we have scaled all densities by log-transforming them then scaling by the within year species- and lake-specific mean and standard deviation (so all species in all lake-years will have a mean scaled density of zero and standard deviation of one).

We will split the data into testing and training sets, so we can evaluate how well our models fit out of sample. As there are multiple years of data here, we will use data from the even years to fit (train) models, and that from the odd years to test the fit:

```r
zoo_train = subset(zooplankton, year%%2==0)
#the modulus (%%) finds the remainder after division by  the right.
#here we use it to find even numbers

zoo_test = subset(zooplankton, year%%2==1)
```

Our first exercise here will be to demonstrate how to model community-level variability in seasonality, by regressing scaled density on day of year, with species-specific curves. As we are not interested here in average seasonal dynamics, we will focus on models 4&5[14]. As this is seasonal data, we will use cyclic smoothers as the basis for seasonal dynamics.

[14]Here we are focusing on only the most common species in the data set. If we wanted to estimate the seasonal dynamics for rarer species, adding a global smooth term might be useful, so we could could borrow information from the more common species.

Model 4:

```
zoo_comm_mod4 = gam(density_scaled~s(day, taxon,
                                     bs="fs",
                                     k=10,
                                     xt=list(bs="cc")),
           data=zoo_train,
           #we need to specify the start and end knots for day
           knots = list(day =c(1,365)),
           #We'll use ML as we are comparing models that differ
           #in fixed effects
           method = "ML"
           )

#Print the summary table of just the smooth terms
print(round(summary(zoo_comm_mod4)$s.table,2))
```

```
##                 edf Ref.df     F p-value
## s(day,taxon) 54.52     71 18.44       0
```

Model 5:

```
# as all of the model features except the formula are the same in model 4&5,
# we just use the update function to refit the model with the new formula
zoo_comm_mod5 = update(zoo_comm_mod4,
                       formula = density_scaled~s(day,
                                                  by=taxon,
                                                  k=10,
                                                  bs="cc"))

print(round(summary(zoo_comm_mod5)$s.table,2))
```

```
##                                  edf Ref.df     F p-value
## s(day):taxonC. sphaericus       5.60      8  8.93       0
## s(day):taxonCalanoid copepods   6.97      8 44.25       0
## s(day):taxonCyclopoid copepods  5.81      8 21.01       0
## s(day):taxonD. mendotae         6.94      8 15.98       0
## s(day):taxonD. thomasi          6.66      8 38.30       0
## s(day):taxonK. cochlearis       3.90      8  3.53       0
## s(day):taxonL. siciloides       6.22      8  5.89       0
## s(day):taxonM. edax             5.14      8 27.44       0
```

Both models have very similar fits, with a mean squared error of 0.63 for model 4 and 0.63 for model 5 (the mean squared error for the original data equals 1 because of the scaling).

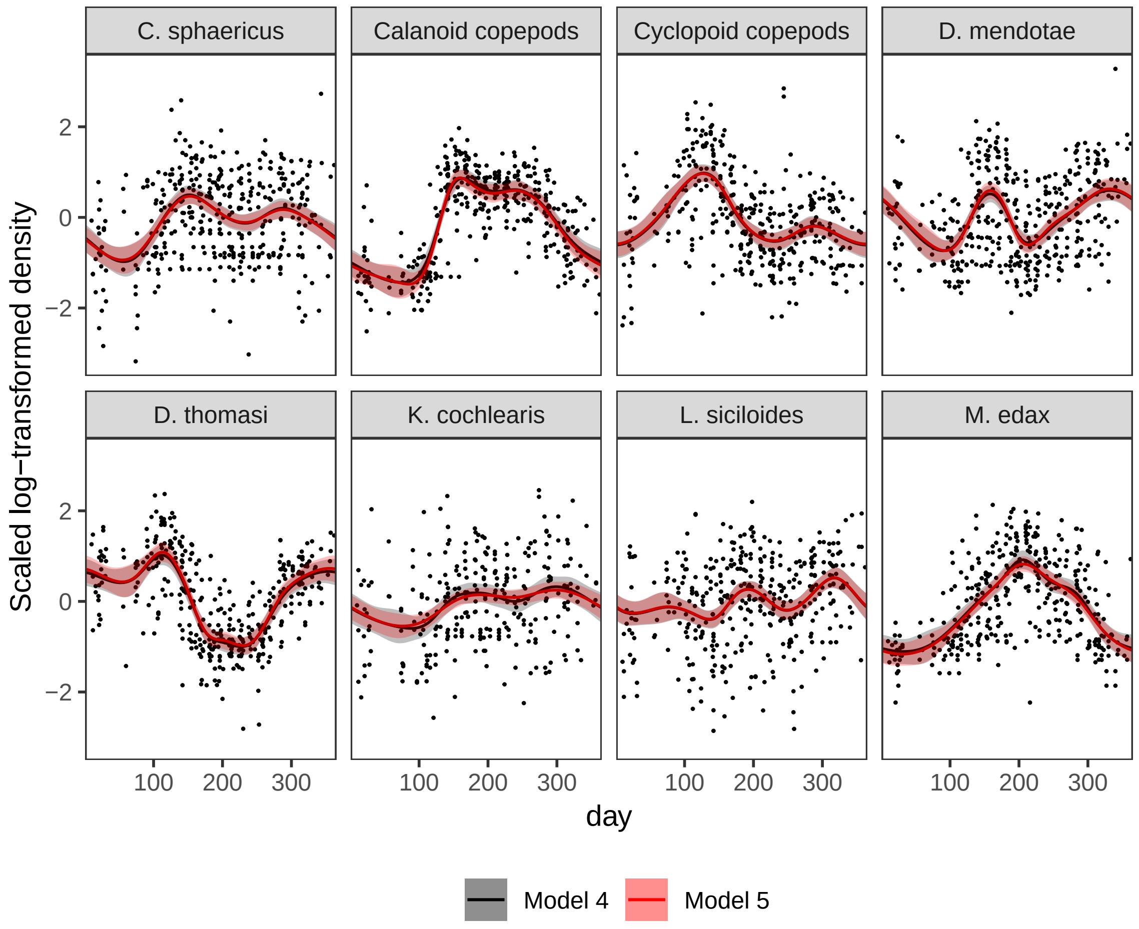

Figure 16: Species-species specific seasonal dynamics for the eight zooplankon species tracked in Lake Mendota. Black points indicate individual plankton observations (after log-transformation and centering and scaling). Lines indicate predicted average values for model 4 (black) and model 5 (red). Ribbons indicate ± 2 standard errors around the mean.

Model 5 has a somewhat lower AIC (`AIC(zoo_comm_mod4)` = 7115, `AIC(zoo_comm_mod5)` = 7104), implying a better overall fit. However, the two models are almost indistinguishable when plotted on top of each other (Figure 16).

The two curves are very close for all species, but the differences in smoothness that resulted in model 5 having an higher AIC than model 4 seem to be driven by the low seasonality of *Keratella cochlearis* and *Leptodiaptomus siciloides* relative to the other species. Still, both models show very similar fits to the training data, model 5 is only slightly better at predicting out of sample fits for *K. cochlearis*, and not at all better for *L. siciloides* (Table 3).

Now let's look at how to fit inter-lake variability in dynamics for just *Daphnia mendotae*. Here, we will compare models 1,2, and 3, to determine if a single global function is appropriate for all four lakes, or if we can effectively model variation between lakes with a shared smooth or lake-specific smooths.

Table 3: Out-of-sample predictive ability for model 4 and 5 applied to the zooplankton community dataset. MSE values represent the average squared difference between model predictions and observations for held-out data (zero predictive ability would correspond to a MSE of one).

taxon	model 4 MSE	model 5 MSE
C. sphaericus	0.81	0.81
Calanoid copepods	0.50	0.49
Cyclopoid copepods	0.66	0.67
D. mendotae	0.83	0.83
D. thomasi	0.32	0.32
K. cochlearis	0.88	0.89
L. siciloides	0.83	0.83
M. edax	0.63	0.63

Model 1:

```r
daphnia_train = subset(zoo_train,taxon=="D. mendotae")
daphnia_test = subset(zoo_test,taxon=="D. mendotae")

zoo_daph_mod1 = gam(density_scaled~s(day, bs="cc",k=10),
           data=daphnia_train,
           knots = list(day =c(1,365)),
           method = "ML"
           )

print(round(summary(zoo_daph_mod1)$s.table,2))
```

```
##         edf Ref.df     F p-value
## s(day) 6.82      8 13.96       0
```

Model 2:

```r
zoo_daph_mod2 = update(zoo_daph_mod1,
                       formula = density_scaled~s(day, bs="cc",k=10) +
                                                s(day,lake, k=10, bs="fs",
                                                  xt=list(bs="cc")))


print(round(summary(zoo_daph_mod2)$s.table,2))
```

```
##              edf Ref.df     F p-value
## s(day)      6.80      8 10.88       0
## s(day,lake) 5.35     35  0.43       0
```

Model 3:

```
zoo_daph_mod3 = update(zoo_daph_mod1,
                       formula = density_scaled~s(day, bs="cc",k=10) +
                                                s(day,by=lake, k=10,
                                                  bs="cc"))

print(round(summary(zoo_daph_mod3)$s.table,2))
```

```
##                       edf Ref.df     F p-value
## s(day)               6.86      8 13.99    0.00
## s(day):lakeKegonsa 0.05      8  0.01    0.35
## s(day):lakeMendota 0.00      8  0.00    0.74
## s(day):lakeMenona  0.93      8  0.20    0.16
## s(day):lakeWaubesa 2.24      8  1.45    0.00
```

The AIC values indicate that both model 2 (1017) and 3 (1017) are better fits than model 1 (1025), but models 2 and 3 have similar fits to one another. There does not seem to be a large amount of inter-lake variability (the effective degrees of freedom per lake are low in models 2&3), and model 3 indicates that only Lake Waubesa deviates substantially from the overall dynamics. The plots for all three models (Figure \ref{fig:daph_smooth) show that Mendota and Menona lakes are very close to the average and to one another for both models (which is unsurprising,as they are very closely connected by a short river) but both Kegons and Waubesa show evidence of a more pronouced spring bloom and lower winter abundances. While this is stronger in Lake Waubesa, model 2 (Figure \ref{fig:daph_smooth, black line) shows that it is still detectable in Lake Kegonsa if we do not need to fit a separate penalty for each lake.

In this case, model 2 is able to predict as well or better out of sample as model 1 or 3 (Table 4), indicating that jointly smoothing the lake together improved model prediction. However, None of the models did well in terms of predicting Lake Kegonsa dynamics out of sample (with a MSE of between 0.95-0.99, compared to a MSE of the original data of 1), indiciating that this model may be be missing substantial year-to-year variability in *D. mendotae* dynamics in this lake.

Bibliography

Baayen, R. Harald, Jacolien van Rij, Cecile de Cat, and Simon N. Wood. 2016. “Autocorrelated Errors in Experimental Data in the Language Sciences: Some Solutions Offered by Generalized Additive Mixed Models.” *arXiv:1601.02043 [Stat]*, January.

Bates, Douglas, Martin Mächler, Ben Bolker, and Steve Walker. 2015. “Fitting Linear

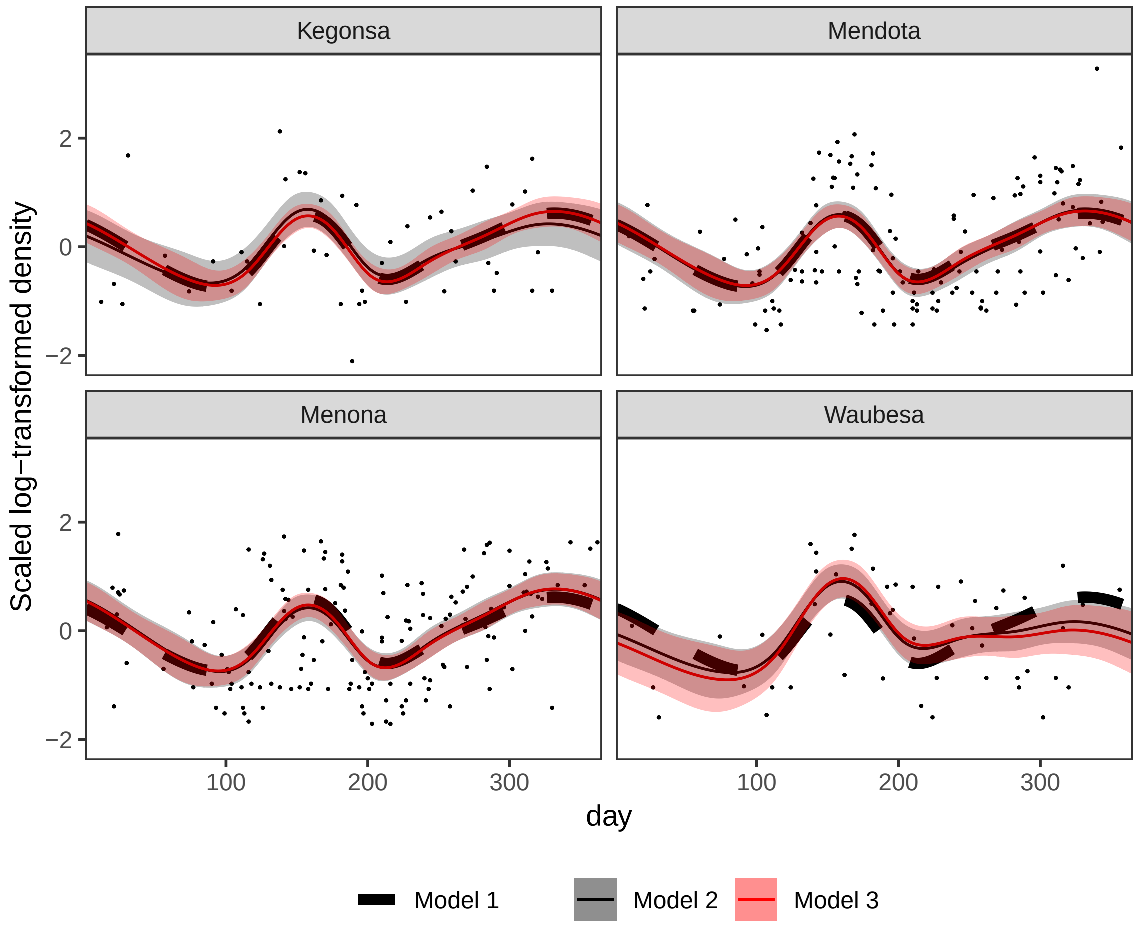

Figure 17: Raw data (points) and fitted models (lines) for *D. mendota* data. Dashed black line: model 1 (no inter-lake variation in dynamics); solid black line: model 2 (interlake variation with similar smoothness); red line: model 3 (varying smooths among lakes). Red and black ribbons indicate ± 2 standard errors around each model.

Table 4: Out-of-sample predictive ability for model 1-3 applied to the *D. mendotae* dataset. MSE values represent the average squared difference between model predictions and observations for held-out data (zero predictive ability would correspond to a MSE of one).

lake	model 1 MSE	model 2 MSE	model 3 MSE
Kegonsa	0.99	0.96	0.99
Mendota	0.78	0.77	0.77
Menona	0.83	0.80	0.80
Waubesa	0.84	0.85	0.90

Mixed-Effects Models Using Lme4." *Journal of Statistical Software* 67 (1): 1–48.

Bolker, Benjamin M, Mollie E Brooks, Connie J Clark, Shane W Geange, John R Poulsen, M Henry H Stevens, and Jada-Simone S White. 2009. "Generalized linear mixed models: a practical guide for ecology and evolution." *Trends in Ecology & Evolution* 24 (3): 127–35.

Boor, Carl de. 1978. *A Practical Guide to Splines*. Springer.

Bürkner, Paul-Christian. 2017. "Brms: An R Package for Bayesian Multilevel Models Using Stan." *Journal of Statistical Software* 80 (1): 1–28.

Carpenter, Bob, Andrew Gelman, Matthew D Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. 2017. "Stan: A Probabilistic Programming Language." *Journal of Statistical Software* 76 (1).

Efron, Bradley, and Carl Morris. 1977. "Stein's Paradox in Statistics." *Scientific American* 236 (5): 119–27.

Forster, Malcolm, and Elliott Sober. 2011. "AIC Scores as Evidence: A Bayesian Interpretation." In *Philosophy of Statistics*, edited by Prasanta S. Bandyopadhyay and Malcolm R. Forster, 535–49. Handbook of the Philosophy of Science 7. Boston, MA: Elsevier B.V.

Gelman, A., J.B. Carlin, H.S. Stern, D.B. Dunson, A. Vehtari, and D.B. Rubin. 2013. *Bayesian Data Analysis, Third Edition*. Chapman & Hall/Crc Texts in Statistical Science. Taylor & Francis.

Gelman, Andrew. 2006. "Multilevel (Hierarchical) Modeling: What It Can and Cannot Do." *Technometrics* 48 (3): 432–35.

Hastie, T J, and Robert J Tibshirani. 1990. *Generalized Additive Models*. Monographs on Statistics and Applied Probability. Taylor & Francis.

Kimeldorf, George S., and Grace Wahba. 1970. "A Correspondence Between Bayesian Estimation on Stochastic Processes and Smoothing by Splines." *The Annals of Mathematical Statistics* 41 (2): 495–502. doi:10.1214/aoms/1177697089.

Lathrop, R. C. 2000. "Madison Wisonsin Lakes Zooplankton 1976 - 1994." *Environmental Data Initiative*. http://dx.doi.org/10.6073/pasta/ec3d0186753985147d4f283252388e05.

McCullagh, P, and John A Nelder. 1989. *Generalized Linear Models, Second Edition*. CRC Press.

McMahon, Sean M, and Jeffrey M Diez. 2007. "Scales of association: hierarchical linear models and the measurement of ecological systems." *Ecology Letters* 10 (6): 437–52.

Potvin, C, M.J. Lechowicz, and S. Tardif. 1990. "The Statistical Analysis of Ecophysiological Response Curves Obtained from Experiments Involving Repeated Measures." *Ecology*, 1389–1400.

Ramsay, James, and B.W. Silverman. 2005. *Functional Data Analysis*. 2nd ed. Springer Series in Statistics. New York, NY: Springer Science+Business Media, Inc.

Reiss, Philip T., Lei Huang, and Maarten Mennes. 2010. "Fast Function-on-Scalar Regression with Penalized Basis Expansions." *The International Journal of Biostatistics* 6 (1). doi:10.2202/1557-4679.1246.

Ruppert, David, M P Wand, and R J Carroll. 2003. *Semiparametric Regression.* Cambridge University Press.

Scheipl, Fabian, Jan Gertheiss, and Sonja Greven. 2016. "Generalized Functional Additive Mixed Models." *Electronic Journal of Statistics* 10 (1): 1455–92. doi:10.1214/16-EJS1145.

Scheipl, Fabian, Ana-Maria Staicu, and Sonja Greven. 2014. "Functional Additive Mixed Models." *Journal of Computational and Graphical Statistics*, April. http://www.tandfonline.com/doi/suppl/10.1080/10618600.2014.901914?scroll=top.

Stanley, R.R.E., E. J. Pedersen, and P.V.R. Snelgrove. 2016. "Biogeographic, Ontogenetic, and Environmental Variability in Larval Behaviour of American Lobster (*Homarus Americanus*)." *Marine Ecology Progress Series* 553: 125–46.

Verbyla, Arũnas P., Brian R. Cullis, Michael G. Kenward, and Sue J. Welham. 1999. "The Analysis of Designed Experiments and Longitudinal Data by Using Smoothing Splines." *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 48 (3): 269–311. doi:10.1111/1467-9876.00154.

Vickers, Mathew J., Fabien Aubret, and Aurélie Coulon. 2017. "Using GAMM to Examine Inter-Individual Heterogeneity in Thermal Performance Curves for *Natrix Natrix* Indicates Bet Hedging Strategy by Mothers." *Journal of Thermal Biology* 63 (January): 16–23. doi:10.1016/j.jtherbio.2016.11.003.

Wieling, Martijn, Fabian Tomaschek, Denis Arnold, Mark Tiede, Franziska Bröker, Samuel Thiele, Simon N Wood, and R Harald Baayen. 2016. "Investigating Dialectal Differences Using Articulography." *Journal of Phonetics* 59: 122–43.

Wood, Simon N. 2006a. "On Confidence Intervals for Generalized Additive Models Based on Penalized Regression Splines." *Australian & New Zealand Journal of Statistics* 48 (4): 445–64.

———. 2006b. *Generalized Additive Models.* An Introduction with R. CRC Press.

Wood, Simon N. 2003. "Thin Plate Regression Splines." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 65 (1): 95–114.

———. 2006. "Low-Rank Scale-Invariant Tensor Product Smooths for Generalized Additive Mixed Models." *Biometrics* 62 (4): 1025–36. doi:10.1111/j.1541-0420.2006.00574.x.

———. 2011. "Fast Stable Restricted Maximum Likelihood and Marginal Likelihood Estimation of Semiparametric Generalized Linear Models." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 73 (1): 3–36. doi:10.1111/j.1467-9868.2010.00749.x.

———. 2017. *Generalized Additive Models: An Introduction with R, 2nd Edition.* 2nd ed. Boco Raton, FL: CRC Press.

Wood, Simon N., Yannig Goude, and Simon Shaw. 2015. "Generalized Additive Models for Large Data Sets." *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 64

(1): 139–55. doi:10.1111/rssc.12068.

Wood, Simon N., Natalya Pya, and Benjamin Säfken. 2016. "Smoothing Parameter and Model Selection for General Smooth Models." *Journal of the American Statistical Association* 111 (516): 1548–63. doi:10.1080/01621459.2016.1180986.

Wood, Simon N., Fabian Scheipl, and Julian J. Faraway. 2013. "Straightforward Intermediate Rank Tensor Product Smoothing in Mixed Models." *Statistics and Computing* 23 (3): 341–60. doi:10.1007/s11222-012-9314-z.