Hierarchical Generalized Additive Models: an introduction with mgcv

Abstract

In this paper, we discuss an extension to two popular approaches to modelling complex structures in ecological data: the generalized additive model (GAM) and the hierarchical model (HGLM). The hierarchical GAM (HGAM), allows modelling of nonlinear functional relationships between covariates and outcomes where the shape of the function itself varies between different grouping levels. We describe the theoretical connection between these models, HGLMs and GAMs, explain how to model different assumptions about the degree of inter-group variability in functional response, and show how HGAMs can be readily fitted 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 example data.

I: Introduction

Two of the most popular and powerful modelling techniques currently in use by ecologists are generalized additive models (GAMs; Wood, 2006c) 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 used to solve different problems. GAMs are used to estimate smooth functional relationships between predictor variables and the response. Examples of such relationships would be the distribution of abundance of a population as a function of depth (Stanley, Pedersen & Snelgrove, 2016) or the swimming speed of snakes as function of temperature (Vickers, Aubret & 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 population counts, or

to model individual level heterogeneity in a study with repeated observations of multiple individuals.

However, the connection between HGLMs and GAMs is quite deep, both conceptually and mathematically (Verbyla et al., 1999). HGLMs and GAMs fit highly variable models by "pooling" parameter estimates towards one another. In an HGLM, this occurs as group-level effects are pulled towards global effects. In a GAM, this occurs in the enforcement of a smoothness criterion on the variability of a functional relationship, pulling parameters towards a straight line.

Given this connection, a natural extension to the standard GAM framework is to allow smooth functional relationships between predictor and response to vary between groups, but in such a way that the different functions are in some sense pooled toward a common shape. We often want to know both how functional relationships vary between groups, and if a relationship holds 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, we can use HGAMS 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 separate 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 a smooth that did not predict any specific group well. In our example, using a single global temperature-size relationship would miss that the three species have 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.

This paper discusses several approaches to group-level smoothing, and corresponding trade-offs. We focus on fitting HGAMs with popular the *mgcv* package for the R statistical programming language, which allows for a variety of HGAM model structures and fitting strategies. We discuss options available to the modeller and practical and theoretical reasons for choosing them. We 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 HGAM formulations, 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 work through example analyses using this approach, to demonstrate the modelling process and how HGAMs can be incorporated into the ecologist's quantitative toolbox. Finally, in part IV, we discuss some of the computational and statistical issues involved in fitting HGAMs in *mgcv*. We have also included all the code needed to make the figures for this document in supplemental code (online), and on the GitHub repository associated with this paper github.com/noamross/mixed-effects-gams.

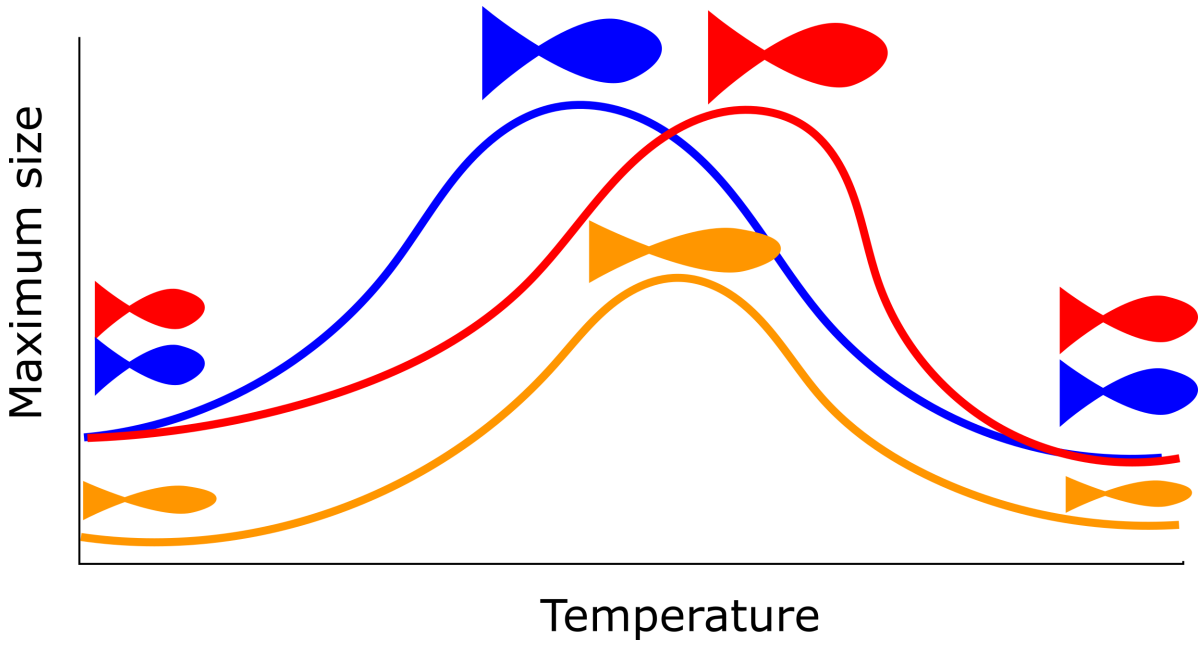

Figure 1: Hypothetical example of functional variability between different group levels. Each line indicates how the maximum possible body size for different species of fish in a community might vary as a function of average water temperature. The orange species shows lower maximum size at all temperatures,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

II: A review of Generalized Additive Models

The generalized linear model (GLM; McCullagh & Nelder, 1989) relates a response (y) to a linear combination of explanatory variables. The response is assumed to be conditionally distributed according to some exponential family distribution (e.g., binomial, Poisson or Gamma distributions for trial, count or strictly positive real response, respectively). The generalized additive model (GAM; Hastie & Tibshirani, 1990; Ruppert, Wand & Carroll, 2003; Wood, 2006c) allows the relationships between the explanatory variables (henceforth covariates) and the response to be described by smooth terms (usually *splines* (de Boor, 1978), but potentially other structures). In general we have models of the form:

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

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 in this example, though it is possible to construct smooths of multiple variables.

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

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

K, referred to as "basis size", "basis complexity" or "basis richness", determines the maximum complexity of each smooth.

It would seem that large basis size could lead to overfitting, but this counteracted by a *smoothing penalty* that influences basis function coefficients so as to prevent excess wiggliness and ensure appropriate complexity of each smooth term. For each term, a *penalty matrix* ($\mathbf{S}$), specific to the form of the basis functions, is pre- and post-multiplied by the parameter vector $\boldsymbol{\beta}$ to calculate the penalty ($\mathbf{S}^T\boldsymbol{\beta}\mathbf{S}$). We then penalize the model likelihood by the penalty term, controlling the trade-off via a *smoothing parameter* (λ). The penalized likelihood used to fit the model is thus

$$L - \boldsymbol{\lambda}\mathbf{S}^T\boldsymbol{\beta}\mathbf{S}$$

Figure 2 shows the results of setting different smoothing parameters for a simple one-dimensional smoothing problem: optimal smoothing in the first plot; the second plot shows what happens when the smoothing parameter is set to zero: interpolation (the penalty has no effect); the right plot shows when the smoothing parameter is set to a very large value, resulting in a straight line.

To measure the complexity of an penalized smooth terms 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 (2006c, Section 4.4) for further details).

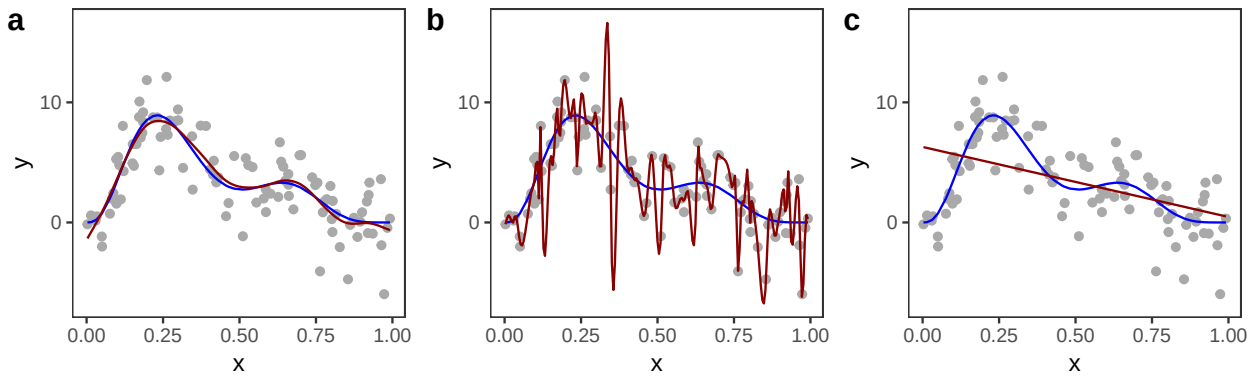

Figure 2: Examples of how different choices of the smoothing parameter effect the resulting function. Data (points) were generated from the blue function and noise added to them. In the left plot the smoothing parameter was estimated using REstricted Maximum Likelihood to give a good fit to the data, 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.

Basis types and penalty matrices

Different types of smooths are useful for different needs, and have different associated penalty matrices for their basis function coefficients. In the examples in this paper, we will use three types of smooths: thin plate regression splines, cyclic cubic smooths, and random effects.

Thin plate regression splines (TPRS; Wood, 2003), are a general purpose spline basis which can be use for problems in any number of dimensions, provided one can assume that the amount of smoothing in any of the covariates is the same (so called isotropy or rotational invariance). TPRS, like many splines, use a penalty matrix made up of terms based on the the squared derivatives of basis functions across their range. Models that overfit the data will tend to have large derivatives, so this penalization reduces wiggliness. Typically, TPRS are second-order ($m = 2$), meaning that squared second derivatives are used. However, TPRS may be of lower order ($m = 1$), or higher order ($m > 2$). We will see in section III how lower-order TPRS smooths are useful in fitting HGAMs. 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 smoother that penalizes the squared second derivative of the smooth across the function. In these, though, start and end of the smoother are constrained to match in value and first derivative. These are useful for fitting models with cyclic components such as seasonal effects. We will use these smoothers to demonstrate how to fit HGAMs to cyclic data.

Random effects are also "smooths" in this framework. In this case, the penalty matrix is the inverse of the covariance matrix of the basis function coefficients (Kimeldorf & Wahba,

1970; Wood, 2017a). 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 software (Verbyla et al., 1999).

This connection between random effects and splines extends beyond the varying-intercept case. Any single-penalty 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. See Verbyla et al. (1999) or Wood, Scheipl & 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 (this is also why figure 2c resulted in a linear, rather than flat, fit to the data). 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 be 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. A different basis can be used in each marginal smooth, as required for the data at hand.

There are two approaches used in *mgcv* for generating tensor products. The first approach (Wood, 2006a) 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

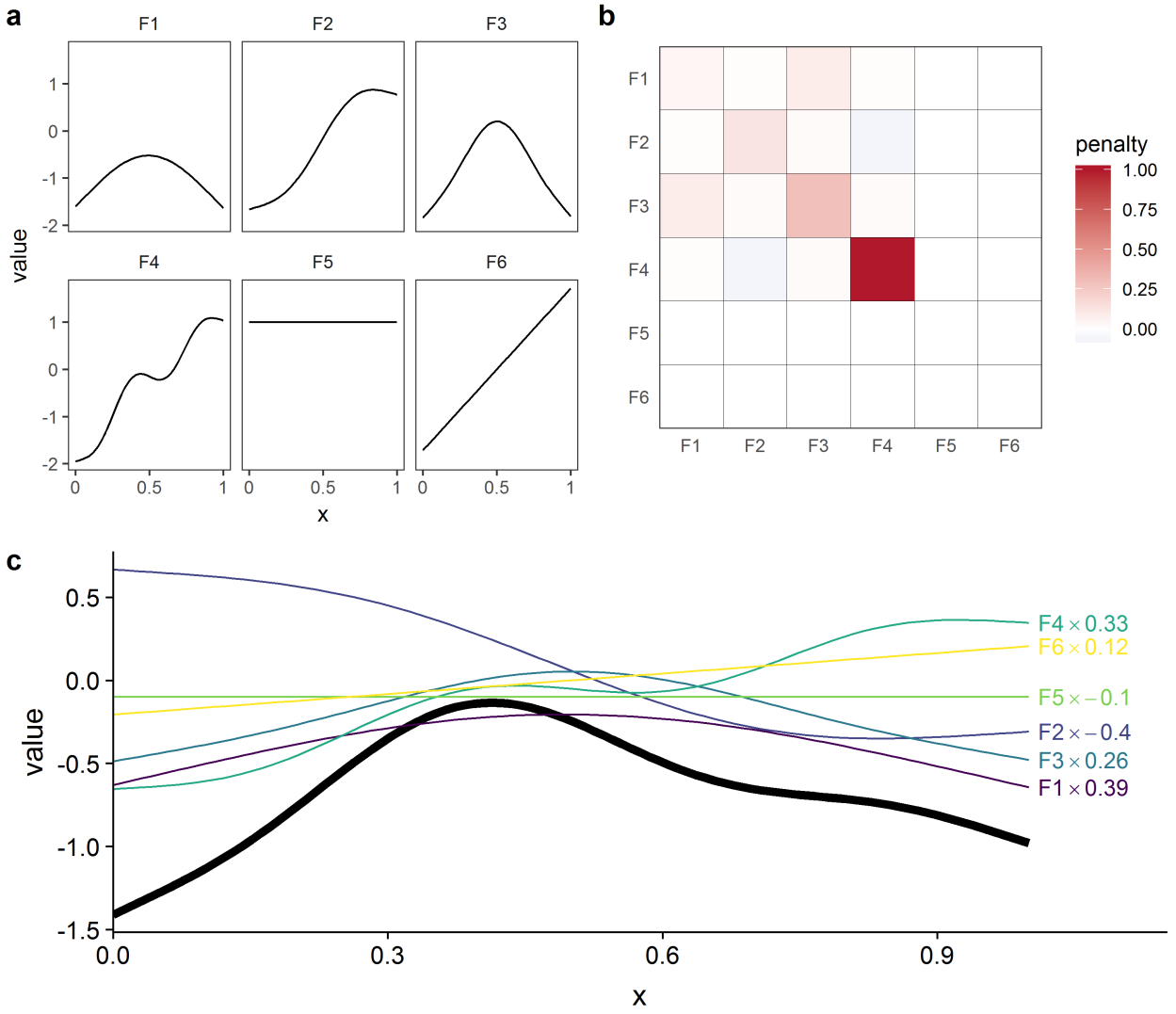

Figure 3: a) Examples of the basis functions associated with a six basis function thin plate spline (m=2), calculated for data, x, spread evenly between $x = 0$ and $x = 1$. Each line represents a single basis function. b) The smoothing penalty matrix for the thin plate smoother. Red entries indicate positive values and blue indicate negative values. For example, 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. c) An example of how the basis functions add up to create a single smooth function. Thin coloured lines represent each basis function multiplied by a coefficient, and the solid black line is the sum of those basis functions.

approach (Wood, Scheipl & Faraway, 2013) separates each penalty into penalized (rank space) and unpenalized (nullspace) components (components that don't have derivatives, such as intercept and linear terms in a one-dimensional cubic spline), 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 smoothing parameters, 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 penalty applies to all basis functions), so it can be fitted using standard mixed effect software such as *lme4* (Bates et al., 2015).

Comparison to hierarchical linear models

Generalized linear mixed effect models (Gelman, 2006; GLMMs; also referred to as hierarchical generalized linear models, multilevel models etc; e.g., Bolker et al., 2009) 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, in an empirical study, individuals may be nested within sample sites, sites are nested within forests, and forests within provinces. 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 & 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 smooth way. The next section investigates the extension of the smoothers we showed above to the case where observations are grouped and we model group-level smooths.

III: What are hierarchical GAMs?

What do we mean by hierarchical smooths?

In this section, we will describe how to model inter-group variability using smooth curves and how to fit these models using *mgcv*. Model structure is key in this framework, so we start with three model 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?

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. Wigglyness 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 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 penalties akin to those described in section II. Figure 4, model 4 illustrates the case where models have different shapes. Similarly, two curves could have very similar overall shape, but differ in their wiggliness. For instance, one function could be equal to another plus a high-frequency oscillation term. 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 V. 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 the models in Figure 4 can be coded straightforwardly in *mgcv*. We will use two example datasets to demonstrate how to code these models (see the supplemental 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 & Tardif (1990) of CO_2 uptake in grasses under varying concentrations of CO_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 HGAMs to estimate inter-plant variation in functional responses. This data set has been modified from the default version available with R, to recode the `Plant` variable as an unordered factor `Plant_uo`[1].

[1]Note that *mgcv* requires that grouping or categorical variables be coded as factors in R. It will will raise an error message if passed data coded as character. It is also important to know whether the 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,

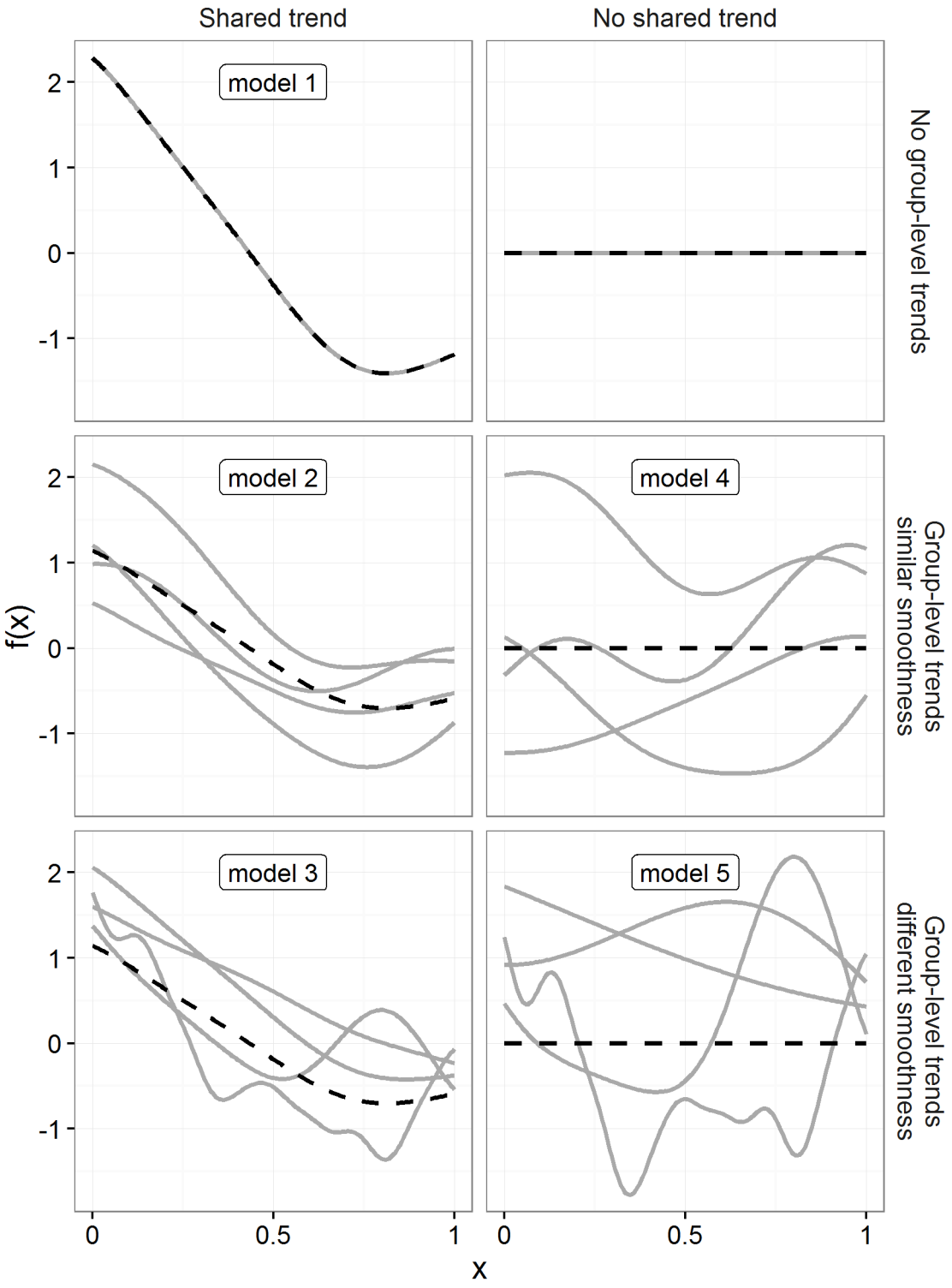

Figure 4: Alternate types of functional variation f(x) that can be fitted with HGAMs. The dashed line indicates the average function value for all groups, and each solid line indicates the functional value at a given predictor value for an individual group level. The null model (of no functional relationship between the covariate and outcome, top right), is not explicitly assigned a model number.

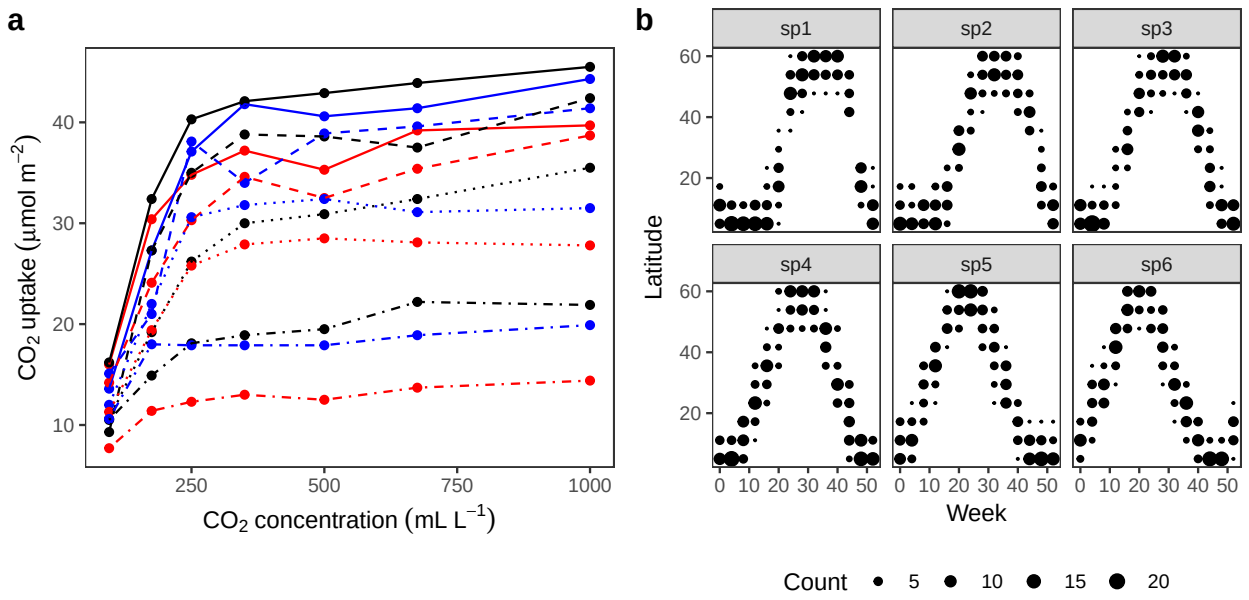

Figure 5: Example data sets used throughout section III. a) Grass CO_2 uptake versus CO_2 concentration for 12 individual plants. Color and line linetype included to distinguish individual plant trends. b) Simulated data set of bird migration, with point size corresponding to weekly counts of 6 species along a latitudinal gradient (zeros excluded for clarity).

B. Simulated bird movement data along a migration corridor, sampled throughout the year (see supplemental code). 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. The true model used to generate this data was model 2: a single global function plus species-specific deviations around that global function.

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) rather than the default GCV criteria 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. See supplemental code for details on data processing steps.

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.

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 $\log_e(\texttt{uptake})$ as a function of two smooths: a thin plate regression spline of $\log_e$-concentration, and a random effect for plant to model plant-specific intercepts.[2] Mathematically:

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

where $\zeta_{\texttt{Plant\_uo}}$ is the random effect for plant and ϵ_i is a Gaussian error term. Here 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", family = "gaussian")
```

This is a common GAM structure, 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`[3].

Figure 6 illustrates *mgcv*'s default plotting out for `CO2_mod1`: the left panel shows the estimated smooth of concentration, and the right shows a quantile-quantile plot of the estimated random effects vs Gaussian 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 from 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 supplemental code for more details on how to generate these predictions).

Examining these plots, we see that while functional responses among plants are similar, some patterns are not captured by this model. For instance, for `Qc2`, residuals are clearly all

[2]Note that we're actually modelling $\log_e(\texttt{uptake})$; 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 by simple random effects): $\log_e(f_1(x)) = \log_e(\alpha) + \log_e(f_2(x))$.

[3]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 the specified `k`. <–! I'm not sure this is at all helpful without further explanation. If we just use "maximum number of basis functions" above, would that be sufficiently accurate? –>

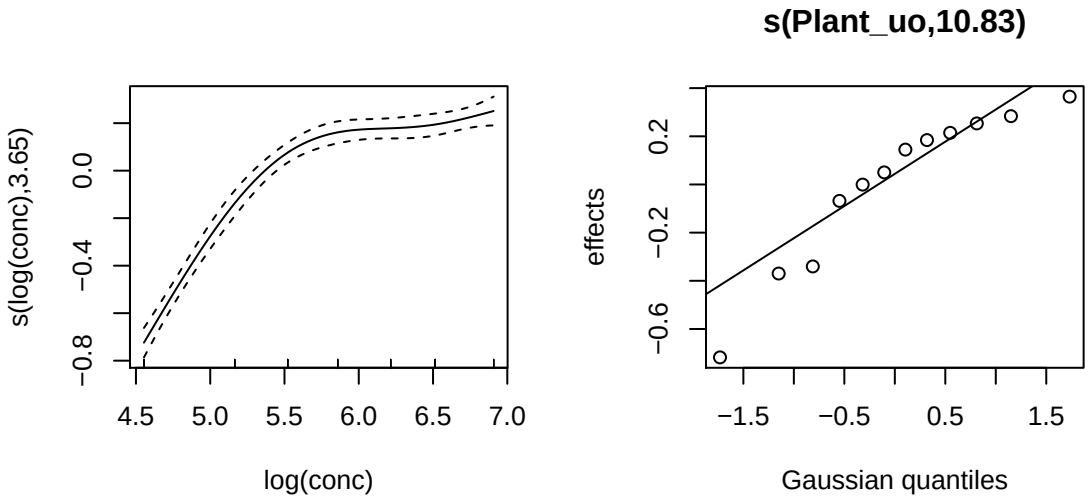

Figure 6: *mgcv* plotting output for model 1 applied to the CO2 dataset. Left shows the smooth of $\log_e$ concentration and right plot shows a quantile-quantile plot of the random effects against Gaussian quantiles, used to check the appropriateness of the normal random effect assumption. Numbers in the vertical axis of the left figure and the title of the right give the effective degrees of freedom of the smooths.

distributed on one side of the estimated curve. A model including individual differences in functional responses may better explain variation.

For our bird example, we model the count of birds as a function of location and time, including their interaction. For this we structure 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 thin plate regression spline (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. For this case we ignore species-specific variability.

```
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 (cf. figure 5b).

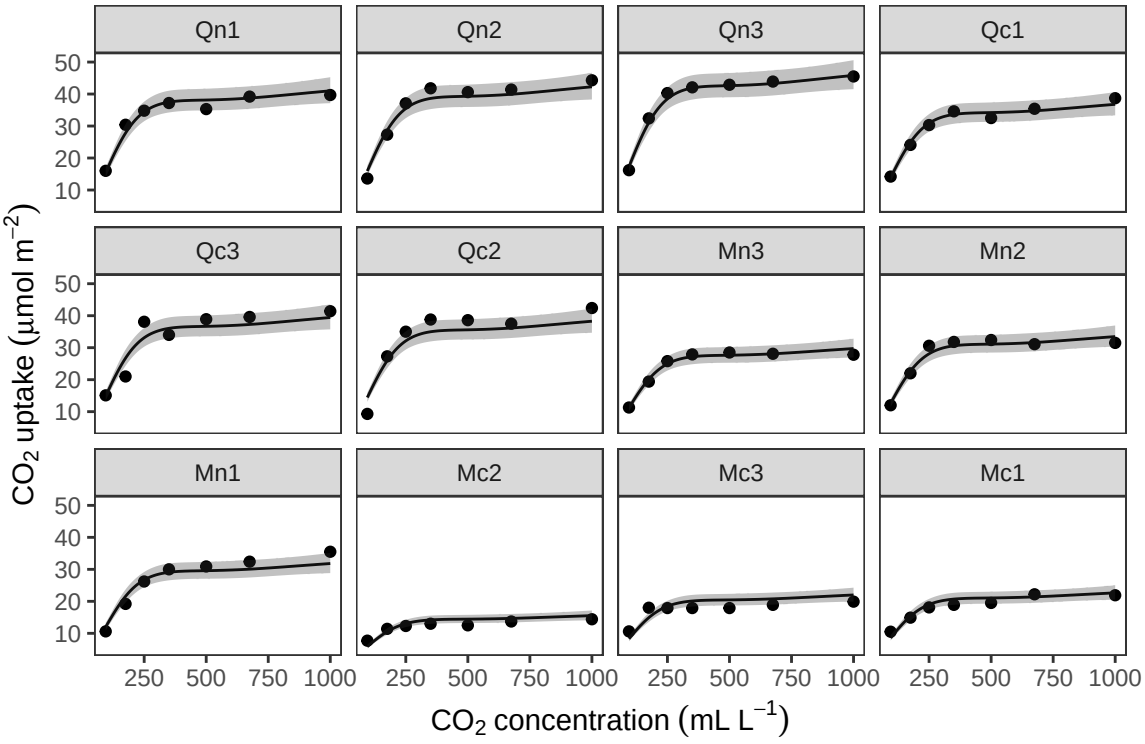

Figure 7: Predicted uptake function ($\pm$ 2 s.e.) for each plant, based on model 1 (a single global function for uptake plus a individual-level random effect intercept). Model predictions are for log-uptake, but are transformed here to show the fitted function on the original scale of the data.

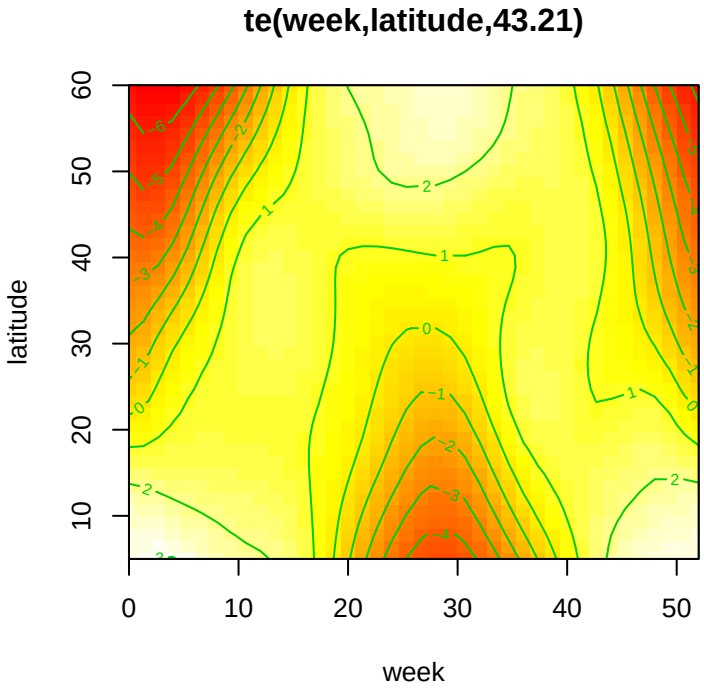

Figure 8: Plot illustrating the average log-abundance of all bird species at each latitude for each week, with yellow colours indicating more individuals and red colours fewer.

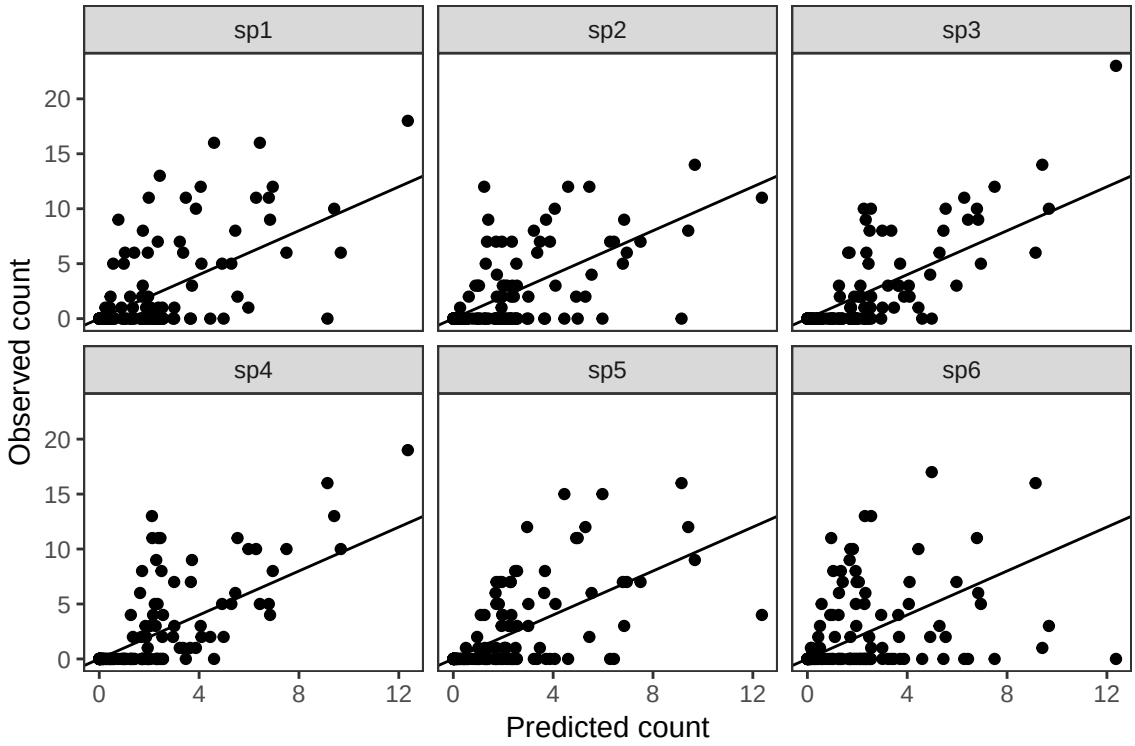

Figure 9: Observed counts by species versus predicted counts from `bird_mod1` (1-1 line added as reference). If our model fitted well we would expect that all species should show similiar patterns of dispersion around the 1-1 line (and as we are assuming the data is Poisson, the variance around the mean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

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) estimate of migration timing (figure 8), and viewing species-specific plots of observed versus predicted values (figure 9), it is apparent 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.

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

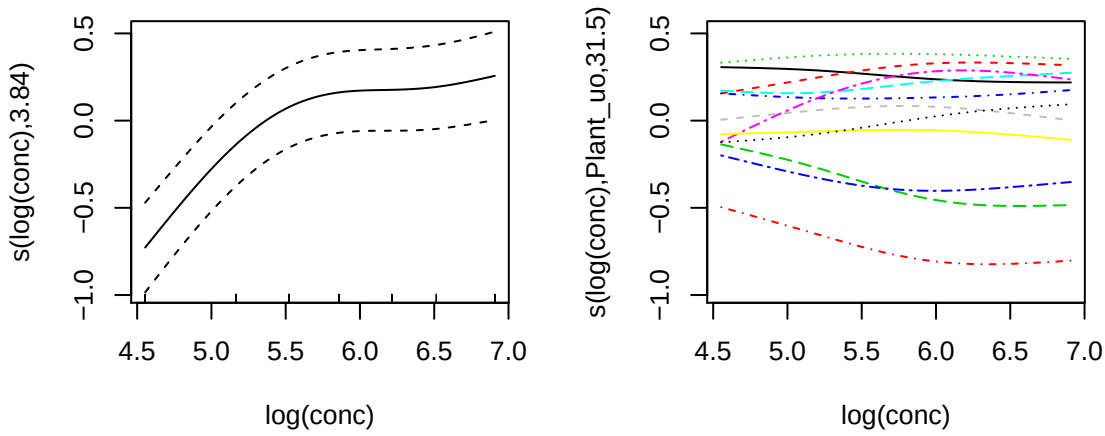

Figure 10: Global function (left) and group-specific deviations from the global function (right) for `CO2_mod2`

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 `?mgcv::smooth.construct.fs.smooth.spec` for details). 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. To ensure that all parts of the smooth can be shrunk towards zero effect, each component of the penalty nullspace is given its own penalty[4].

We modify the previous CO_2 model to incorporate group-level smooths 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:

```
CO2_mod2 <- gam(log(uptake) ~ s(log(conc), k=5, m=2) +
                              s(log(conc), Plant_uo, k=5,  bs="fs", m=2),
                data=CO2, method="REML")
```

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

[4]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.

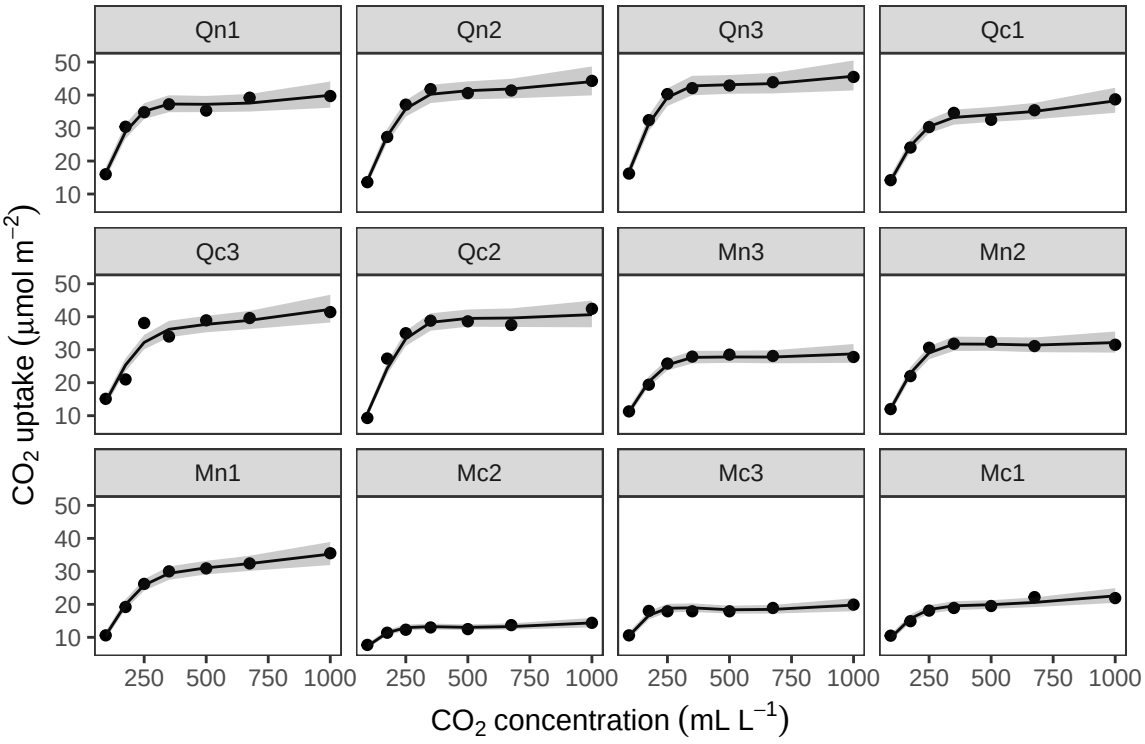

Figure 11: Predicted uptake values (lines) versus observed uptake for each plant, based on model 2.

smooths combine to predict uptake rates for individual plants. We see that, unlike in the single global smooth case above, none of the curves deviate from the data systematically.

The `"fs"`-based approach mentioned above does not work for higher-dimensional tensor product smooths. Instead, the group-specific term can be specified with a tensor product of the continuous smooths and a random effect for the grouping parameter[5]. 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)
```

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

[5]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 equivalent 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 V on computational issues for details).

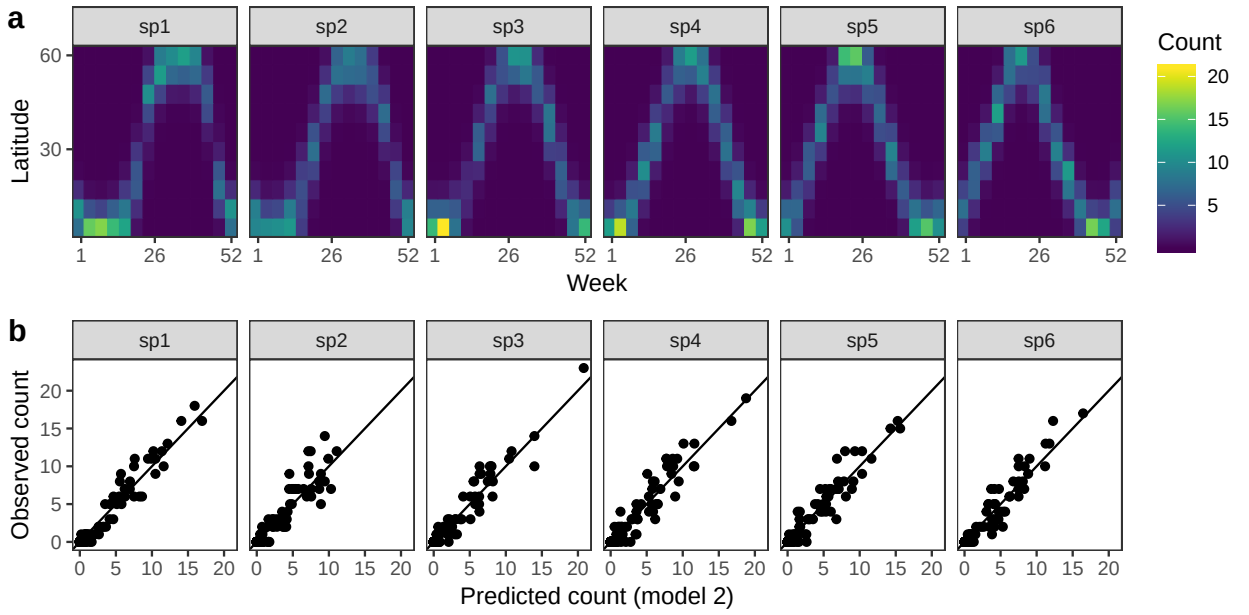

Figure 12: a) Predicted migration paths for each species based on `bird_mod2`, with lighter colors corresponding to higher predicted counts. b) Observed counts versus predictions from `bird_mod2`.

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 its own level of wigglyness. This increases the computational cost of the model (as there are more smoothing parameters to estimate), 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 separate 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 the formula for this model as:

```
y ~ s(x, bs="tp") + s(x, by=fac, m=1, bs="ts") + s(fac, bs="re")`.
```

Note three major differences here 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 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

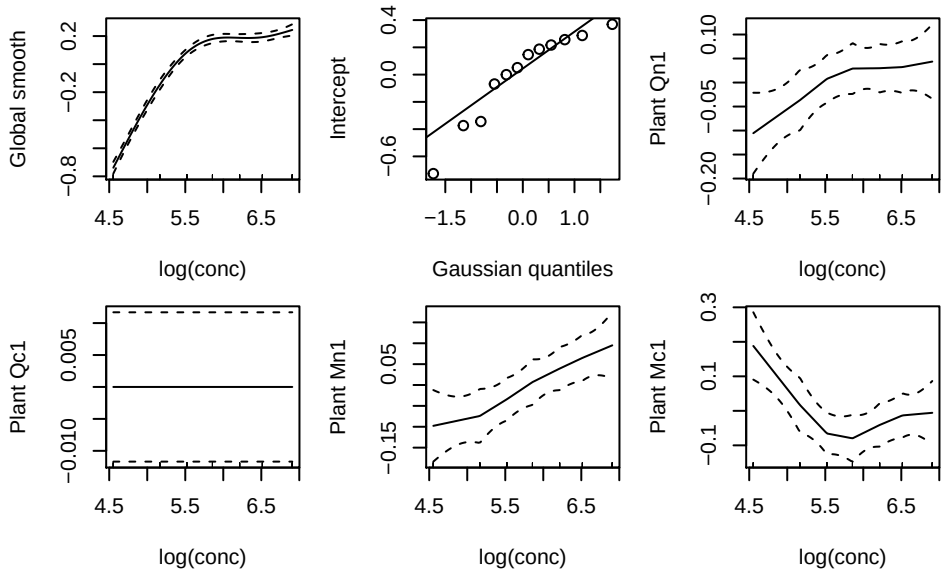

Figure 13: Functional relationships for the CO2 data estimated for model 3. Top left: the global smooth; Top middle: species-specific random effect intercepts. The remaining plots are a selected subset of the plant-specific smoothers, indicating how the functional response of that plant differs from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

(`bs="ts"`, for TPRS with shrinkage). The `by` method does not automatically penalize the null space, so there is potential for collinearity between unpenalized components of the global and group-level smoothers. Using `ts` helps reduce this issue.

3. We specify `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. This, also, reduces 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 collinear with the global smooth (**???**; Wieling et al., 2016). We have observed that this is much more of an issue when fitting model 3 compared to model 2.

We modify the `CO2` model to follow this approach like so:

```
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. It is apparent 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 isotropic 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)
```

As above, here we used a "ts" smooth for the latitude marginal effect to penalize the null space and avoid issues of collinearity between the global and groupwise smoother.

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 do not 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 way as for models 1-3 above, the plots for these datasets are very similar to the plots for model 2. This will not always be the case; if in a given study there are very few data points in each grouping level (relative to the strength of the functional relationship of interest), estimates from model 4 will typically be much more variable than from model 2, as there is no way for the model to share information on function shape between grouping levels without the global smooth term. See section V on computational issues for more on how to choose between different models.

```
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)` (as above, 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 groupwise smoothers are necessary. However, we do not currently recommend this method. There is nots sufficient theory on how accurate parametric p-values will be for comparing these models; there is uncertainty about what degrees of freedom to assign to models with varying smooths, and slightly different model specifications may not result in nested 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 model structures. There is well-developed theory of how to include effects of penalization and smoothing parameter uncertainty when estimating the model complexity penalty for AIC (Wood, Pya & Säfken, 2016). We demonstrate this approach in Table 1. Using AIC, 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 inter-group 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 we simulated the data from a model with this structure!)

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 & 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 not selecting models based purely on AIC. Instead, model selection should be based on expert subject knowledge about the system, computational time, and most importantly, the inferential goals of the study. 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 this is not possible within the framework of model 3. The next section works through two examples to show how to choose between different models, and section V discusses these and other model fitting issues

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

in more depth.

IV: Examples

We now demonstrate two worked examples on one data set to highlight how to use HGAMs 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.

For these examples, data are from a long-term study in seasonal dynamics of zooplankton, collected by the Richard Lathrop. The data were collected from a chain of lakes in Wisconsin (Mendota, Monona, Kegnonsa, and Waubesa) approximately bi-weekly from 1976 to 1994. They consist of samples of the zooplankton communities, taken from the deepest point of each lake via vertical tow. The data are provided by the Wisconsin Department of Natural Resources and their collection and processing are fully described in Lathrop (2000).

Here inferential aims are *(i)* estimate variability in seasonality among species in the community, and *(ii)* estimate between-lake variability for the most abundant taxon in the sample (*Daphnia mendotae*). As we are focusing on seasonal cycles rather than average or maximum abundances. We have log-transformed all densities, then centered and scaled them by the within year, species and lake mean; all species in all lake-years will have a mean scaled density of zero and standard deviation of one.

To enable evaluation of out-of-sample performance, we split the data into testing and training sets. As there are multiple years of data, so we use data from the even years to fit (train) models, and the odd years to test the fit:

```
zoo_train <- subset(zooplankton, year%%2==0)
zoo_test <- subset(zooplankton, year%%2==1)
```

Our step 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 and 5 (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). As the data are seasonal, we use cyclic smoothers as the basis for seasonal dynamics. Therefore we need to specify start and end points for our cycles using the `knots` argument to `gam`, as well as specify that this is smoother type to the factor-smooth using the `xt` argument:

Model 4:

```
zoo_comm_mod4 <- gam(density_scaled ~s(day, taxon,
                                       bs="fs",
                                       k=10,
                                       xt=list(bs="cc")),
                     data=zoo_train,
                     knots = list(day =c(1, 365)),
                     method = "ML")
```

Note that we use maximum likelihood (`method = "ML"`) as our smoothing selection method. This is required as we want to compare models that differ in their fixed effects.

Model 5:

```
zoo_comm_mod5 <- gam(density_scaled~s(day, by=taxon,
                              k=10, bs="cc"),
                     data=zoo_train,
                     knots = list(day =c(1, 365)),
                     method = "ML")
```

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). 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 14).

Model 5's higher AIC than model 4 seems 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 2).

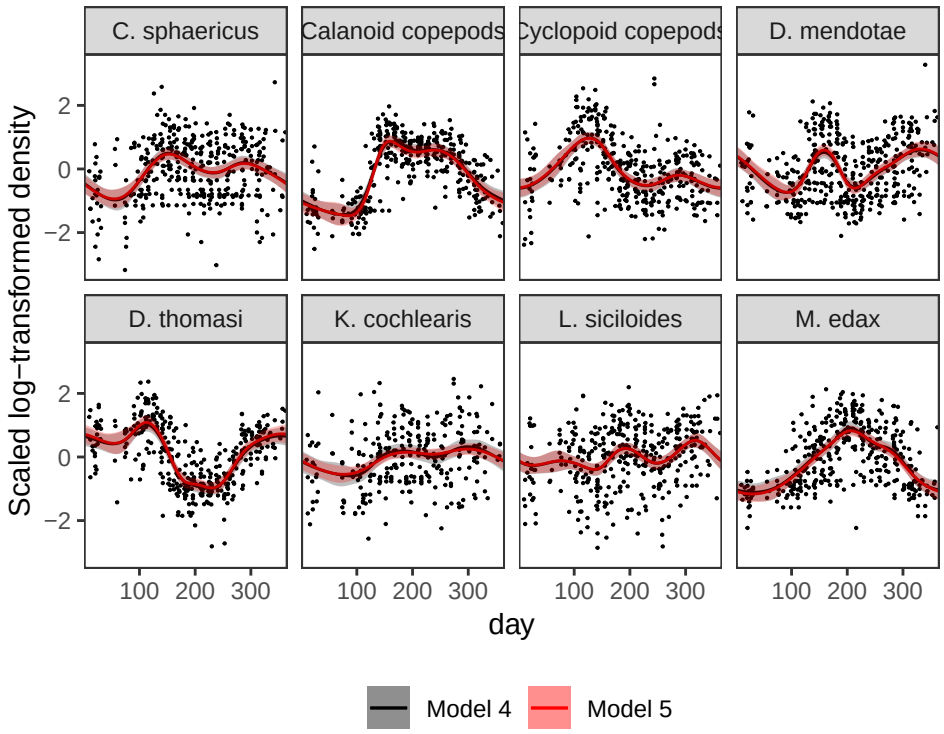

Figure 14: Species-specific seasonal dynamics for the eight zooplankon species tracked in Lake Mendota. Black points indicate individual plankton observations (after log-transformation and centering and scaling). Lines indicate predicted average values for model 4 (black) and model 5 (red). Ribbons indicate ± 2 standard errors around the mean.

Table 2: 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 test data.

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

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 more effectively model variation between lakes with a shared smooth and lake-specific smooths.

Model 1:

```
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")

printCoefmat(summary(zoo_daph_mod1)$s.table)
```

```
##           edf Ref.df      F   p-value
## s(day) 6.8235 8.0000 13.956 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Model 2:

```
zoo_daph_mod2 <- gam(density_scaled~s(day, bs="cc", k=10) +
                               s(day, lake, k=10, bs="fs",
                                 xt=list(bs="cc")),
                     data=daphnia_train,
                     knots=list(day =c(1, 365)),
                     method="ML")

printCoefmat(summary(zoo_daph_mod2)$s.table)
```

```
##                 edf  Ref.df       F   p-value
## s(day)       6.7974  8.0000 10.8812 < 2.2e-16 ***
## s(day,lake)  5.3485 35.0000  0.4319  0.003896 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Table 3: 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

Model 3:

```
zoo_daph_mod3 <- gam(density_scaled~s(day, bs="cc", k=10) +
                                   s(day, by=lake, k=10, bs="cc"),
                     data=daphnia_train,
                     knots=list(day =c(1, 365)),
                     method="ML")

printCoefmat(summary(zoo_daph_mod3)$s.table)
```

```
##                            edf     Ref.df       F   p-value
## s(day)              6.85523608 8.00000000 13.9850 < 2.2e-16 ***
## s(day):lakeKegonsa 0.04880150 8.00000000  0.0062  0.345995
## s(day):lakeMendota 0.00056926 8.00000000  0.0000  0.738127
## s(day):lakeMenona  0.92709308 8.00000000  0.1979  0.161287
## s(day):lakeWaubesa 2.23534077 8.00000000  1.4538  0.001164 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The AIC values indicate that both model 2 (1017.21) and 3 (1016.79) are better fits than model 1 (1025.13), but models 2 and 3 have similar fits. 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). Model 3 indicates that only Lake Waubesa deviates substantially from the overall dynamics. Plots for all three models (Figure 15) show that Mendota and Monona 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 Kegonsa and Waubesa show evidence of a more pronounced spring bloom and lower winter abundances. While this is stronger in Lake Waubesa, it is still detectable with simpler model 2 in Lake Kegonsa (Figure 15, black line).

Model 2 is able to predict as well or better out of sample as model 1 or 3 (Table 3), indicating that jointly smoothing the lakes together improved model prediction. 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), indicating that this model may be be missing substantial year-to-year variability in *D. mendotae* dynamics in this lake.

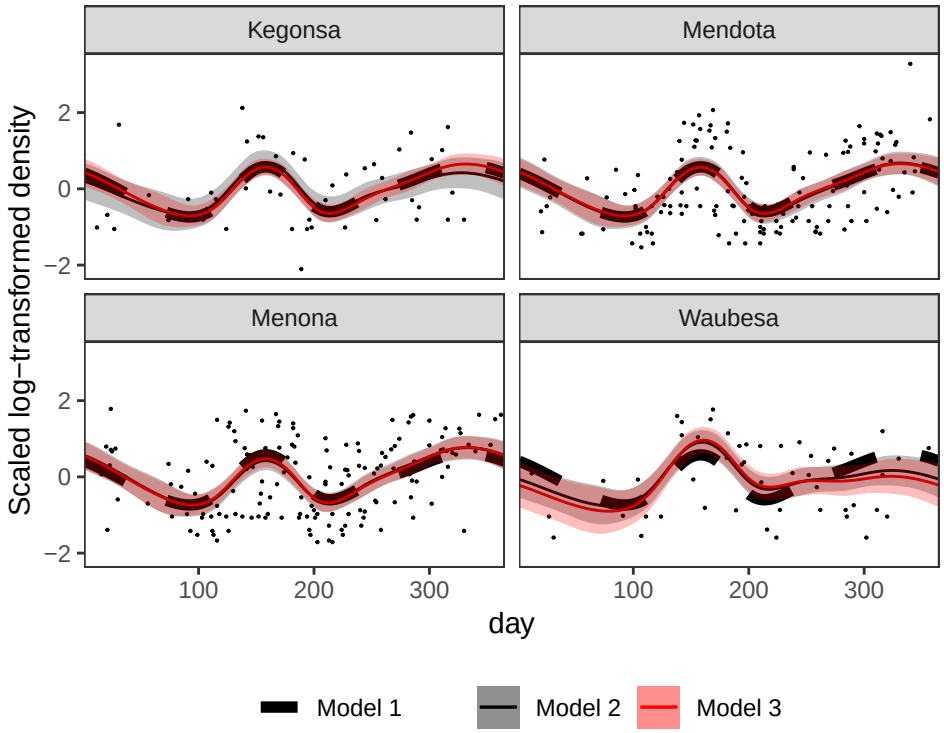

Figure 15: Raw data (points) and fitted models (lines) for extit{D. mendota} data. Dashed black line: model 1 (no inter-lake variation in dynamics); solid black line: model 2 (interlake variation with similar smoothness); red line: model 3 (varying smooths among lakes). Red and black ribbons indicate ± 2 standard errors around each model.

V: Computational and statistical issues when fitting HGAMs

Which of the five model formulations should you choose for a given data set? There are two major trade-offs to consider. 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 trade-off is model complexity versus computational cost: 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 discuss both of these trade-offs in this section. We also discuss how to extend the HGAM framework to fit more complex models.

Bias-variance trade-offs

The bias-variance trade-off is a fundamental concept in statistics. When trying to estimate any relationship (in the case of GAMs, a smooth relationship between predictors and data) bias measures how far, on average, an estimate is from the true value. The variance of an estimator corresponds to how much that estimator would fluctuate if applied to multiple different samples of the same size 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.[6] Penalization is useful because that a penalty term slightly increases model bias, but can substantially decrease variance (Efron & Morris, 1977).

In GAMs, the bias-variance trade-off is managed by the terms of the penalty matrix, and equivalently random effect variances in HGLMs. 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 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 predict new data.

The bias-variance trade-off 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, over-smoothed (biased) estimates for the most wiggly terms, or a mixture of these two, depending on the fitting criteria.

We developed a simple numerical experiment to determine whether *mgcv* fitting criteria tend

[6] 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).

to set estimated smoothness penalties high or low in the presence of among-group variability in smoothness when fitting model 2/4 HGAMs. 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 the response, 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. 16a), requiring equal smoothness for all group levels resulted in *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. 16b). This implies that assuming equal smoothness will result in underestimating the true smoothness of low-variability terms, and thus lead to more variable estimates of these terms. If this is a potential issue, we recommend fitting both models 4 and 5 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 separate smoothness levels, in which case model 2 or model 4 may still be the better option even in 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. This is currently not implemented in *mgcv*. It is possible to set up this type of varying penalty model in flexible Bayesian modelling software such as *Stan* (see below for a discussion of how to fit HGAMs using these tools), but how to this type of model has not been well studied.

It may seem there is also a bias-variance trade-off between choosing to use a single global smoother (model 1) or a global smoother plus group-level terms (models 2 and 3). In model 1, all the data is used to estimate a single smooth term, and thus should have lower variance than models 2 and 3, but higher bias for any given group in the presence of inter-group functional variability. However, in practice, this trade-off will be handled via penalization; 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 and 3 should generally be driven by computational costs. Model 1 is typically much faster to fit than models 2 and 3, even in the absence of among–group differences. 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 and 3 and models 4 and 5. If all group levels have very different functional shapes, the global term will get penalized toward zero in models 2 and 3, so they will reduce to models 4 and 5. The choice to include a global term should be made based on scientific considerations (is the global term of interest?) and computational considerations.

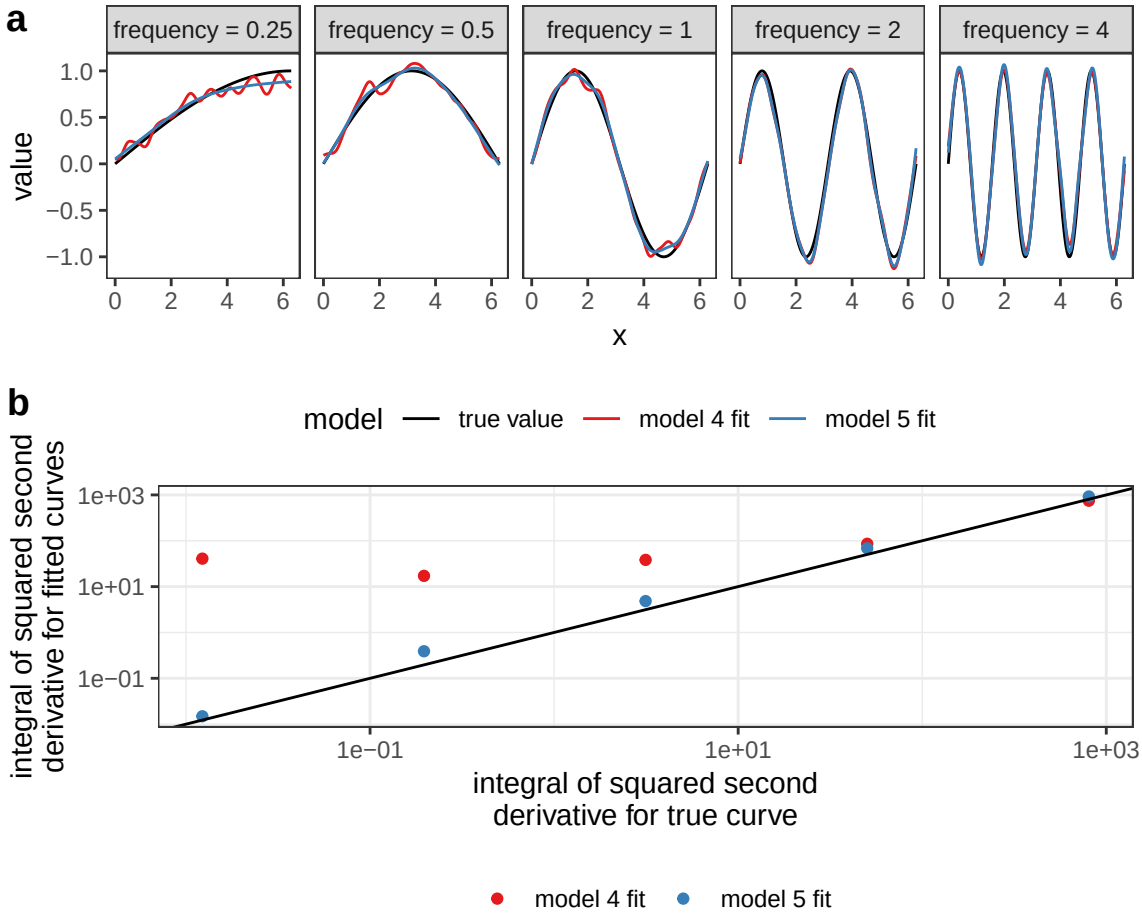

Figure 16: a) Illustration of bias that can arise from assuming equal smoothness for all group levels (model 4, red line) versus allowing for intergroup variation in smoothness (model 5, red line) when the true function (black line) shows substantial variation in smoothness between groups. b) Estimated wiggliness (as measured by the integral of the 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.

Complexity–computation trade-offs

The more flexible a model is, the larger an effective parameter space any fitting software has to search. It can be surprisingly easy to use massive computational resources trying to fit models to even small datasets. While we typically want to select models based on their fit and our inferential goals, computing resources can often act as an effective upper bound on model complexity. For a given data set, assuming a fixed family and link function, the time taken to estimate an HGAM will depend (roughly) on four factors: *(i)* the number of basis functions to be estimated, *(ii)* the number of smoothing parameters to be estimated, *(iii)* whether the model needs to estimate both a global smooth and groupwise smooths, and *(iv)* the algorithm and fitting criteria used to estimate parameters.

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. For a dataset with n_g different groups and n data, fitting a model with just a global smooth, `y~s(x,k=k)` will require k coefficients, and takes $\mathcal{O}(nk^2)$ operations to evaluate. 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 cost by an order of the number of groups squared. The effect of this is visible in the examples we fit in section III. Table 4 compares the relative time it takes to compute model 1 versus the other models.

One way to deal with this issue would be to reduce the number of basis functions used when fitting group-level smooths when the number of groups is large, limiting the flexibility of the model. It can also make sense to use more computationally–efficient basis functions when fitting large data sets, such as P-splines (Wood, 2017b) or cubic splines. Thin-plate splines entail greater of computational costs. (Wood, 2017a).

Including a global smooth (models 2 and 3 compared to models 4 and 5) will not generally substantially affect the number of coefficients needed to estimate (Table 4). Adding a global term will add at most `k` extra terms. It can be substantially less that that, as `mgcv` drops basis functions from co-linear smooths to ensure that the model matrix is full rank.

Adding additional smoothing parameters (moving from model 2 to model 3, or moving from model 4 to 5) is 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 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 4). Note that this will not hold in all cases. For instance, model 5 takes less time to fit the bird movement data than model 4 does (Table 4B).

Table 4: 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 number of coefficients measures the total number of model parameters (including intercepts). The number 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	7	65	3
3	14	65	14
4	4	61	3
5	14	61	13
B. bird movement data			
1	1	90	2
2	120	540	5
3	140	624	14
4	110	541	3
5	67	535	12

Alternative formulations: `bam`, `gamm`, and `gamm4`

When fitting models with large numbers of groups, it is often possible to speed up computation substantially by using one of the alternative fitting routines available through *mgcv*.

The first option is the funcion `bam`, this requires the least changes to existing code written using the `gam` function. `bam` is designed to improve performance when fitting large data sets via two mechanisms. First, it saves on memory needed to compute a given model by using a random subset of the data to calculate the basis functions. It then blocks the data and updates model fit within each block (Wood, Goude & Shaw, 2015). While this is primarily designed to reduce memory usage, it can also substantially reduce computation time. Second, when using `bam`'s default fREML ("Fast REML") method, you can use the `discrete=TRUE` option: this discretizes each covariate, substantially reducing the amount of computation needed (see `?mgcv::bam` for more details).

`bam` has a larger computational overhead than `gam`, so for small numbers of groups, it can be slower than `gam` (Figure 17). 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 17). Note that `bam` can be somewhat less computationally stable when estimating these models (i.e., less likely to converge).

The second option is to fit models using one of two dedicated mixed effect model estimation

packages, *nlme* and *lme4*. The `mgcv` package includes the function `gamm` uses the *nlme* package to estimate a GAM, automatically handling the transformation of smooth terms into random effects (and back into basis function representations for plotting and other statistical analyses). The `gamm4` function, in the separate *gamm4* package, uses the *lme4* in a similar way. 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 17)[7].

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 a few changes to model code. 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`. 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*, 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 & 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"),
                                    k=c(10, 10, 6),
                                    m=2),
                         data=bird_move, family=poisson)
```

[7]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 parallelizable, and performance of cores can vary. As parallel processing can be complicated and dependent on the type of computer you are using to configure, 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.

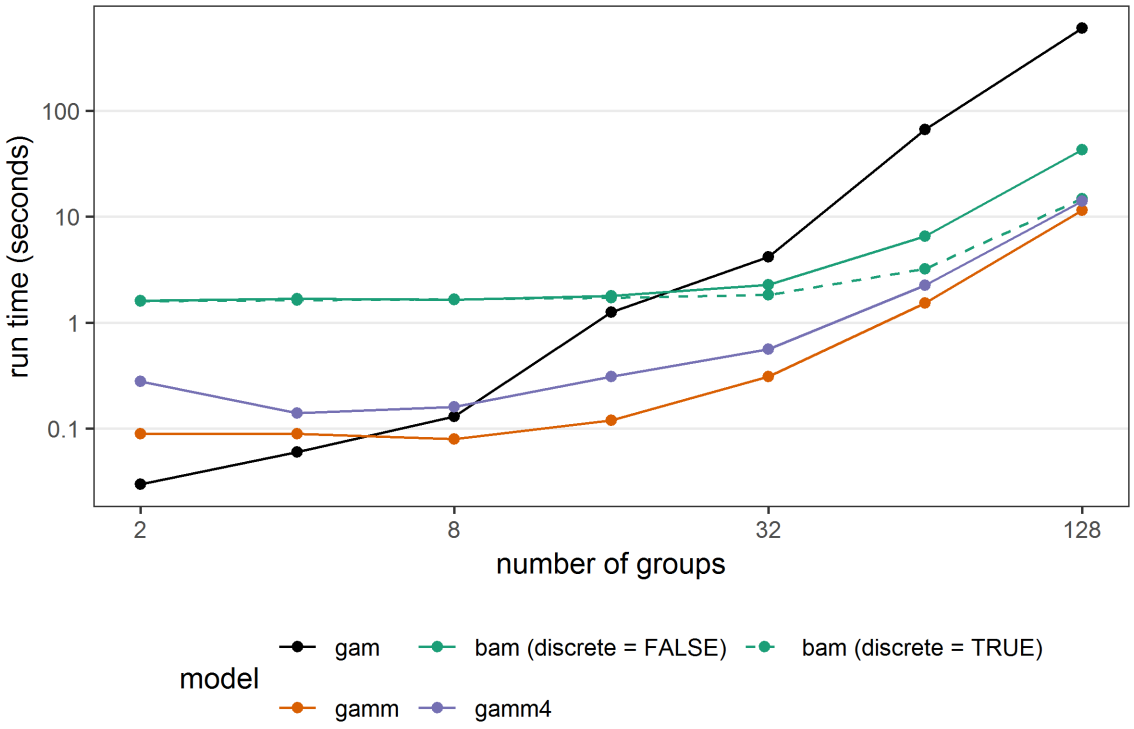

Figure 17: Elapsed time to estimate the same model using each of the four approaches. Each data set was generated with 20 observations per group using a unimodal global function and random group-specific functions consisting of an intercept, a quadratic term, and logistic trend for each group. Observation 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.

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 4). These models can still be estimated because of 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. 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 observed that *mgcv* still occasionally finds solutions with simulated data where the global term is over-smoothed.

To avoid this issue, we recommend both careful choice of basis and setting model degrees of freedom so that groupwise terms are either slightly less flexible than the global term or have a smaller null space. 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[8]. 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 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 is to use a lower number of basis functions (`k`) for groupwise relative to global terms. This will

[8]For model 2, the "fs" smoother, and tensor products of random effect ("re") and other smooth terms do not have a penalized nullspace 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 nullspace, so called "shrinkage" methods: `bs="tp"`, `"cs"`, or `"ps"` have this property.

reduce the maximum flexibility possible in the groupwise terms. We do caution that these are just rules of thumb. In cases where an accurately estimated global smooth is essential, we recommend either fitting model 1 or using specialized functional regression software such as the *refund* package (Scheipl, Staicu & Greven, 2014), which enforces constraints on the groupwise smooths so that they always sum to zero at any given point (avoiding the collinearity issue). Also, see below for more information on functional regression.

A brief foray into the land of Bayes

As mentioned in section II, the penalty matrix can also be treated as the inverse of a prior covariance matrix for model parameters β. Intuitively, the basis functions and penalty we use form a prior (in the informal sense) on how we'd like our model term to behave. REML gives an empirical Bayes estimate of the smooth model (Laird & Ware, 1982), where terms in the null space of the smooth have improper, flat priors (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 (2017a) for more details on this connection). The posterior Bayesian covariance matrix for model parameters can be extracted from any fitted `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 credible intervals, which happen to have good frequentist properties; Wood, 2006b). Viewing our GAM as Bayesian is a somewhat unavoidable consequence of the equivalence of random effects and splines: if we think that there is some true smooth that we wish to estimate, we must take a Bayesian view of our random effects (splines) as we do not think that the true smooth changes each time we collect data (Wood, 2017a, Section 5.8).

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 specified 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.

Similarly, 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 *mgcv*-style 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*.

Beyond HGAMs: functional regression

The HGAMs 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

& Silverman, 2005). HGAMs as we have described them are a form of function-on-scalar regression (Ramsay & Silverman, 2005; Reiss, Huang & Mennes, 2010), where we are trying to estimate a smooth function that varies between grouping levels.

We have deliberately focused our paper on these simpler 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 applications such as 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 & Silverman (2005) for a good introduction, and Scheipl, Gertheiss & Greven (2016) for more recent work in this field). The `refund` package (Reiss, Huang & Mennes, 2010; Scheipl, Staicu & Greven, 2014; Scheipl, Gertheiss & Greven, 2016) uses the statistical machinery from `mgcv` to fit these models, and should be usable by anyone familiar with `mgcv` modelling syntax. Functional regression is also a major area of study in Bayesian statistics (e.g., Kaufman, Sain & others (2010)).

Conclusion

HGAMs are a powerful tool to model intergroup variability, and we have attempted to illustrate some of the range and possibilities that these models are capable of, how to fit them, and some issues that may arise during model fitting and testing. Specifying these models and techniques for fitting them are active areas statistical research, so this paper should be viewed as a jumping-off point for these models, rather than an end-point; we refer the reader to the rich literature on GAMs (e.g. Wood, 2017a) and functional regression (Ramsay & Silverman, 2005; Kaufman, Sain & others, 2010; Scheipl, Staicu & Greven, 2014) for more on these ideas.

Bibliography

Bates D., Mächler M., Bolker B., Walker S. 2015. Fitting linear mixed-effects models using lme4. *Journal of Statistical Software* 67:1–48.

Bolker BM., Brooks ME., Clark CJ., Geange SW., Poulsen JR., Stevens MHH., White J-SS. 2009. Generalized linear mixed models: a practical guide for ecology and evolution. *Trends in Ecology & Evolution* 24:127–135.

Bürkner P-C. 2017. `brms`: An R package for Bayesian multilevel models using Stan. *Journal of Statistical Software* 80:1–28.

Carpenter B., Gelman A., Hoffman MD., Lee D., Goodrich B., Betancourt M., Brubaker M., Guo J., Li P., Riddell A. 2017. Stan: A probabilistic programming language. *Journal of*

statistical software 76.

de Boor C. 1978. *A Practical Guide to Splines*. Springer.

Efron B., Morris C. 1977. Stein's paradox in statistics. *Scientific American* 236:119–127.

Forster M., Sober E. 2011. AIC scores as evidence: A Bayesian interpretation. In: Bandyopadhyay PS, Forster MR eds. *Philosophy of Statistics*. Handbook of the Philosophy of Science. Boston, MA: Elsevier B.V., 535–549.

Gelman A. 2006. Multilevel (hierarchical) modeling: what it can and cannot do. *Technometrics* 48:432–435.

Gelman A., Carlin J., Stern H., Dunson D., Vehtari A., Rubin D. 2013. *Bayesian Data Analysis, third edition*. Taylor & Francis.

Hastie TJ., Tibshirani RJ. 1990. *Generalized Additive Models*. Taylor & Francis.

Kaufman CG., Sain SR., others. 2010. Bayesian functional ANOVA modeling using gaussian process prior distributions. *Bayesian Analysis* 5:123–149.

Kimeldorf GS., Wahba G. 1970. A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. *The Annals of Mathematical Statistics* 41:495–502. DOI: 10.1214/aoms/1177697089.

Laird NM., Ware JH. 1982. Random-effects models for longitudinal data. *Biometrics* 38:963–974.

Lathrop RC. 2000. Madison Wisonsin Lakes Zooplankton 1976 - 1994. *Environmental Data Initiative*.

McCullagh P., Nelder JA. 1989. *Generalized Linear Models, Second Edition*. CRC Press.

McMahon SM., Diez JM. 2007. Scales of association: hierarchical linear models and the measurement of ecological systems. *Ecology Letters* 10:437–452.

Potvin C., Lechowicz M., Tardif S. 1990. The statistical analysis of ecophysiological response curves obtained from experiments involving repeated measures. *Ecology*:1389–1400.

Ramsay J., Silverman B. 2005. *Functional Data Analysis*. New York, NY: Springer Science+Business Media, Inc.

Reiss PT., Huang L., Mennes M. 2010. Fast function-on-scalar regression with penalized basis expansions. *The International Journal of Biostatistics* 6. DOI: 10.2202/1557-4679.1246.

Ruppert D., Wand MP., Carroll RJ. 2003. *Semiparametric Regression*. Cambridge University Press.

Scheipl F., Gertheiss J., Greven S. 2016. Generalized functional additive mixed models. *Electronic Journal of Statistics* 10:1455–1492. DOI: 10.1214/16-EJS1145.

Scheipl F., Staicu A-M., Greven S. 2014. Functional additive mixed models. *Journal of*

Computational and Graphical Statistics 24:477–501.

Stanley R., Pedersen EJ., Snelgrove P. 2016. Biogeographic, ontogenetic, and environmental variability in larval behaviour of American lobster (*Homarus americanus*). *Marine Ecology Progress Series* 553:125–146.

Verbyla AP., Cullis BR., Kenward MG., Welham SJ. 1999. The analysis of designed experiments and longitudinal data by using smoothing splines. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 48:269–311. DOI: 10.1111/1467-9876.00154.

Vickers MJ., Aubret F., Coulon A. 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* 63:16–23. DOI: 10.1016/j.jtherbio.2016.11.003.

Wieling M., Tomaschek F., Arnold D., Tiede M., Bröker F., Thiele S., Wood SN., Baayen RH. 2016. Investigating dialectal differences using articulography. *Journal of Phonetics* 59:122–143.

Wood SN. 2003. Thin plate regression splines. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 65:95–114.

Wood SN. 2006a. Low-rank scale-invariant tensor product smooths for generalized additive mixed models. *Biometrics* 62:1025–1036. DOI: 10.1111/j.1541-0420.2006.00574.x.

Wood SN. 2006b. On confidence intervals for generalized additive models based on penalized regression splines. *Australian & New Zealand Journal of Statistics* 48:445–464.

Wood SN. 2006c. *Generalized Additive Models*. CRC Press.

Wood SN. 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)* 73:3–36. DOI: 10.1111/j.1467-9868.2010.00749.x.

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

Wood SN. 2017b. P-splines with derivative based penalties and tensor product smoothing of unevenly distributed data. *Statistics and Computing* 27:985–989. DOI: 10.1007/s11222-016-9666-x.

Wood SN., Goude Y., Shaw S. 2015. Generalized additive models for large data sets. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 64:139–155. DOI: 10.1111/rssc.12068.

Wood SN., Pya N., Säfken B. 2016. Smoothing parameter and model selection for general smooth models. *Journal of the American Statistical Association* 111:1548–1563. DOI: 10.1080/01621459.2016.1180986.

Wood SN., Scheipl F., Faraway JJ. 2013. Straightforward intermediate rank tensor product smoothing in mixed models. *Statistics and Computing* 23:341–360. DOI: 10.1007/s11222-012-9314-z.