Hierarchical Generalized Additive Models: an introduction with mgcv

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 2006) for modelling flexible regression functions, and generalized linear mixed models ("hierarchical generalized linear models" HGLMs or simply "hierarchical models"; Bolker et al. 2009, A. 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. An example of such a relationship would be a marine population's distribution as a function of depth [ADD CITE] or how plant species grow rates vary with temperature around its optimum [ADD CITE]. 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 or both slope and intercept 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 a 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*.

There are many potential uses for hierarchical GAMs. For example, to estimate how the maximum size different fish species reach varies along a common temperature gradient (fig. 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 fit a single global temperature-size curve plus species-specific curves that were penalized to be close to the mean function.

The capability to fit hierarchical GAMs 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 cover the different approaches to group-level smoothing, the options for each one and why a user might choose it, and demonstrate the different approaches across a range of case studies.

This paper is divided into six sections. Part II is a brief (and friendly) review of how generalized additive models work and their relation to hierarchical models. In part III, we discuss different ways of modelling hierarchical additive models, what assumptions each model makes about how information is shared between

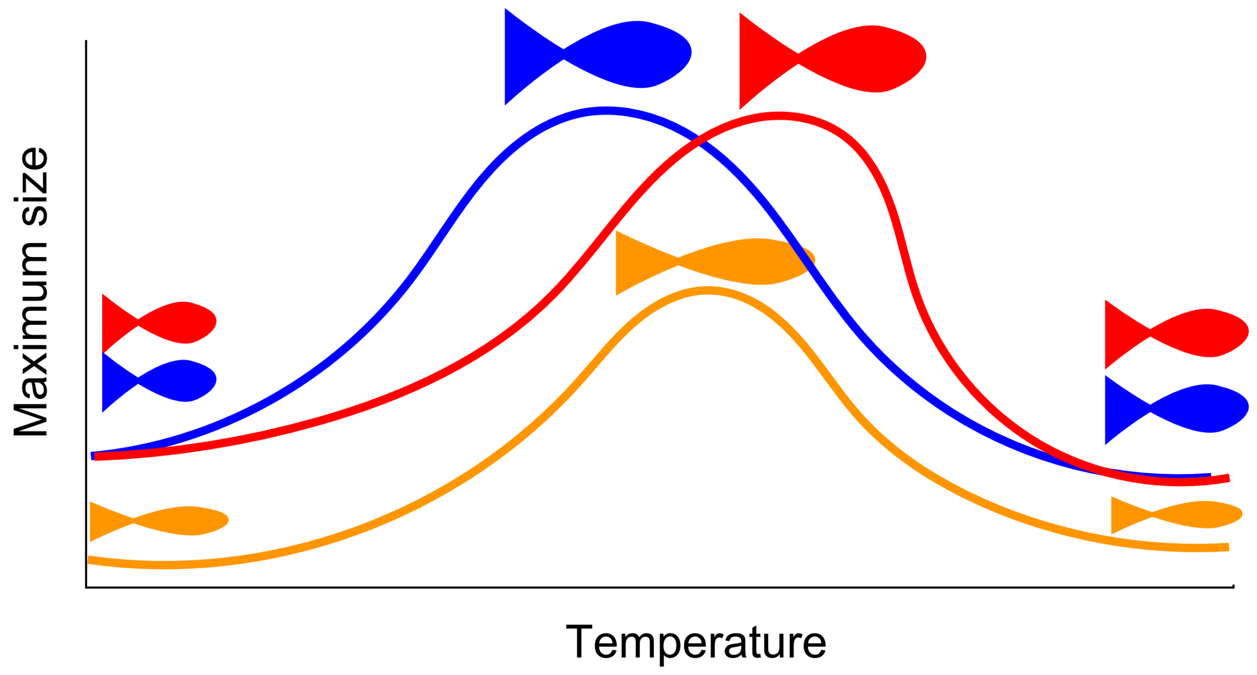


Figure 1:

groups, and different ways of specifying these models in *mgcv*. In part IV, we discuss some of the tools available for plotting model output and assessing model goodness of fit. In part V, we discuss some of the computational and statistical issues involved in fitting hierarchical GAMs in *mgcv*. Finally, in part VI, we work through a few examples of analyses using this approach, to demonstrate the modelling process and how hierarchical GAMs can be incorporated into the quantitative ecologist's toolbox.

II: and introduction to Generalized Additive Models

One of the most common model formulations in statistics is the generalized linear model (McCullagh and Nelder 1989) — models that relate some response (y) to linear combinations of explanatory variables. We may allow allow the response to be distributed according to some exponential family distribution (e.g., letting the response be a trial, a count or a strictly positive number – binomial, Poisson or Gamma distributions, respectively). The generalized additive modelling (GAM) framework (Hastie and Tibshirani 1990; Ruppert, Wand, and Carroll 2003; Simon N Wood 2006) 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 are then talking about models of the form:

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

where y is the response (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_js is represented by a sum of simpler *basis functions* (b_k) multiplied by corresponding coefficients (β_k), which need to be estimated to be estimated:

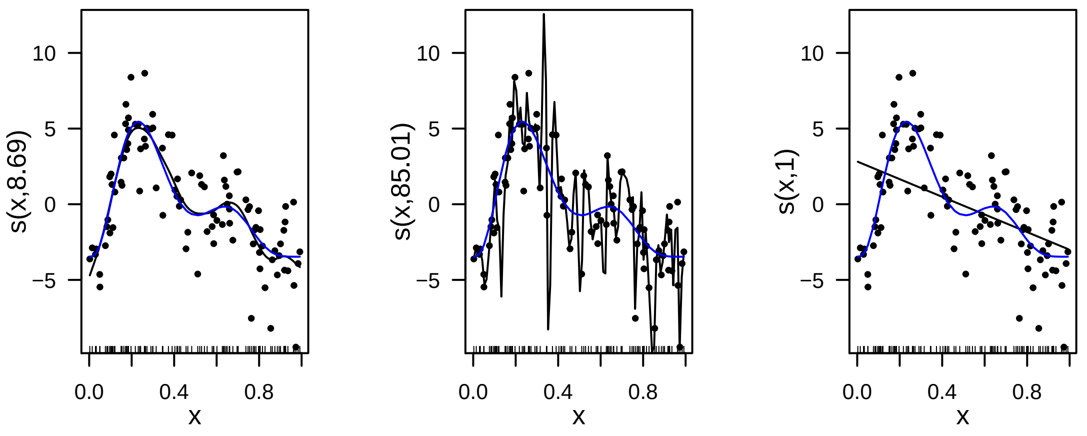


Figure 2: Examples of how different choices of the smoothing parameter effect the resulting function. Data (points) were generated from the blue function and noise added to them. In the left plot the smoothing parameter was estimated to give a good fit to the data, in the middle plot the smoothing parameter 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 wigglyness, giving a straight line. Numbers in the y axis labels show the estimated degrees of freedom for the term.

$$f_j(x_j) = \sum_{k=1}^{K} \beta_k b_k(x_j),$$

The size of K of each smooth will determine the flexibility 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 the functions complexity is appropriate; hence the basis only need to be "large enough" and we let the penalty deal with excess wigglyness.

The penalty for a term is usually based on derivatives of that term – as the derivatives give the wigglyness of the function and hence its flexibility. We trade-off the fit of the model against the wigglyness penalty to obtain a model that both fits the data well but does not overfit. To control this trade-off we estimate a *smoothing parameter*. Figure 2 shows optimal smoothing (where the smoothing parameter is estimated to give a parsimonious model) 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*.

say something about knots!

say something about penalty matrices and also smoothing parameters

The number of basis functions, K, limits the maximum basis complexity for a given smooth term. To measure the wigglyness of a given 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 a larger value indicates a more wiggly term. See Simon N Wood (2006) Section 4.4 for further details.

There are many possible basis functions that can be used to model the b_ks. Here we'll use thin plate regression splines, which have the appealing property that knot choice is somewhat automatic (the best approximation to including knots at each data poiint is used; Simon N. Wood (2003)).

DLM:: put the cubic splines back in here

TPRS are also defined for any number of predictors, so multivariate smoothers can be constructed easily. The basis is *isotropic* so smoothing is treated the same in all directions. So if one had, a bivariate smooth of temperature and time, a one degree change in temperature would equate to a one second change in time, which is an odd assumption to make. In the more general case where units are not alike, we can use *tensor products* to combine two or more univariate smooths into a more complex basis. Each component can be made up from a different basis, playing to their particular strengths.

In the linear modelling literature we can specify a single interaction between terms (in R, `a:b`) or a "full interaction", which includes the marginal terms (`a*b` in R, which is equivalent to `a + b + a:b`). There are parallels for smooths too, allowing us to separate-out the main effect terms from the interactions (in R `te(a, b)` specifies the tensor product which is equivalent to `ti(a) + ti(b) + ti(a, b)`). The ability to separate out the interactions and main effects will become very useful in the next section, once we start looking at group-level smooths.

** this plot doesn't get referenced?!**

We represent the terms in our model as basis functions, which end up as additional columns in our design matrix and parameter vector, and penalties, which penalize the likelihood and stop our model from being too wiggly. Taking a pragmatic Bayesian approach to the problem, the penalty is really a prior on how we think the model should act. In which case the penalty matrix itself is a prior precision matrix (inverse variance) for the term. With that in mind, we can think about random effects as "smooths" in our model, albeit ones with with ridge penalies (CITE). For instance, to include a random effect modelling between group variation in 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 equivilent to how random effects are estimated in standard mixed effect models. The penalty term here is proportionate to the inverse of the variance of the fixed effect estimated by standard hierarchical model solvers [add citation here does this contradict what's above??].

This connection between random effects and basis function smooths extends beyond the varying-intercept case. Any basis-function representation of a smooth function can be transformed so that it can be represented as a random effect with an associated variance. While this is beyond the scope of this paper, see Simon N. Wood, Scheipl, and Faraway (2012) for a more detailed discussion on the connections between these approaches.

Smoothing penalties vs. shrinkage penalties

does this go above??

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 *nullspace*). For 1-dimensional thin plate regression splines, 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 **BLAH**. It is often useful to be able to remove nullspace functions as well, to be able to shrink them to zero if they do not contribute significantly to a given model fit. This can be done either by tweaking the penalty matrix so that it both smooths and shrinks as the single penalty term increases, or by adding a new penalty term that just penalizes the null space for the model. Figure 4 shows an example of what the basis functions (Fig. 4A), and smoothing penalties and shrinkage penalties (Fig. 4B) look like for a 6-basis function cubic spline and 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.

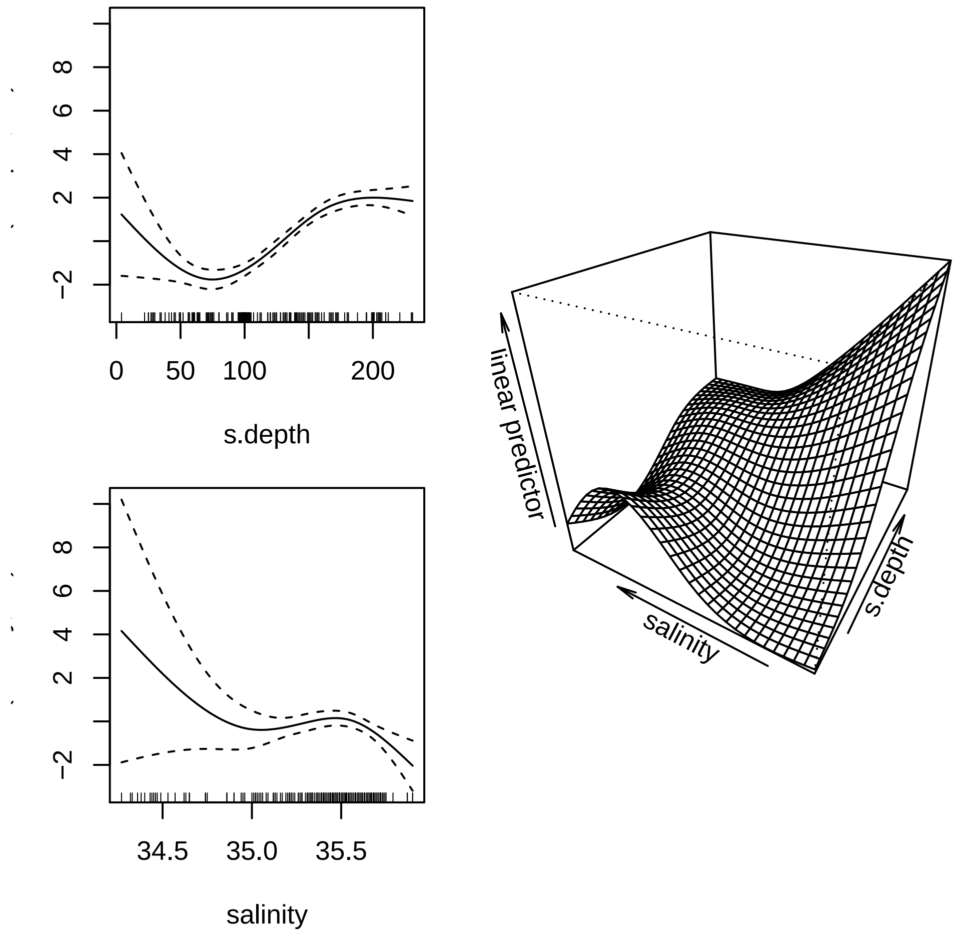


Figure 3: Tensor product of depth and salinity with data taken from a 1992 survey of mackerel eggs. The two left plots show the marginal smooths of each term in the model (`ti(s.depth)` above and `ti(salinity)` below), the right plot shows the interaction effect (`ti(s.depth, salinity)`). Data are from Simon N Wood (2006).

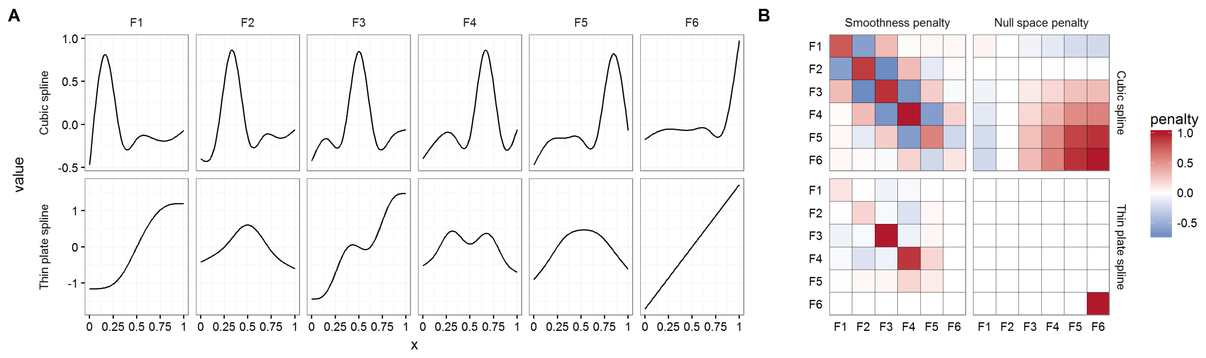


Figure 4:

DLM: say something about cs basis here

EDIT this figure

Comparison to hierarchical linear models

Generalized linear mixed effect models (GLMMs; also referred to as hiearchical generalized linear models, multilevel models etc; e.g., Bolker et al. 2009; Andrew Gelman 2006) are an extension of regression modelling that allow the modeller to include 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 groupings in the data, the groupings used in the 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.

** this last section should say something like: these are both latent gaussian thingos, the model structure is the difference – we want to be able to heirarchically structure our smoothers**

III: What are hierarchical GAMs?

What do we mean by hierarchical smooths?

The smoothers in section II allowed us to model flexible relationships between our response and predictor variables. In this section, we will describe how to model 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 smoothness, 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 questions result in five possible models (Figure 5), beyond the null model of “no relation between response and predictors”.:

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.

DLM: Somewhere (here?) we need to say something about how same wigglyness=/=same shape?

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

EJP: Going with canned and simulated data for the examples rather than real as it’s a bit less messy

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 x_1 and x_2, in the case multiple predictors), and fac to designate the discrete grouping factor whose variation we are interested in understanding.

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

A. The `CO2` dataset, available in R in the `datasets` package. This data is from an experimental study by CITE 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). A total of 12 plants were measured, and uptake measured at 7 concentration levels for each plant. Here we will focus on how to use these techniques to estimate inter-plant variation in functional responses.

B. A hypothetical study of bird movement along a migration corridor. 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`. This example will allow us to demonstrate how to fit these models with interactions and with non-normal (count) data.

DLM: cite bird_move?

It is important to note that the grouping variable should be coded in R as an unordered factor – a character will raise an error and numeric will lead to a completely different model specification. Whether the factor is ordered or not will not matter for most of the smoothers we use here. However, for models 3&5 order will matter (see below for further details).

Throughout the examples we use Restricted Maximum Likelihood (REML) [CITE] 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 [CITE Wood 2011 JRSSB].

In each case some data processing and manipulation has been done to obtain the graphics and results below. We recommend readers take a look at the source RMarkdown [CITE] document for this paper to get the full code.

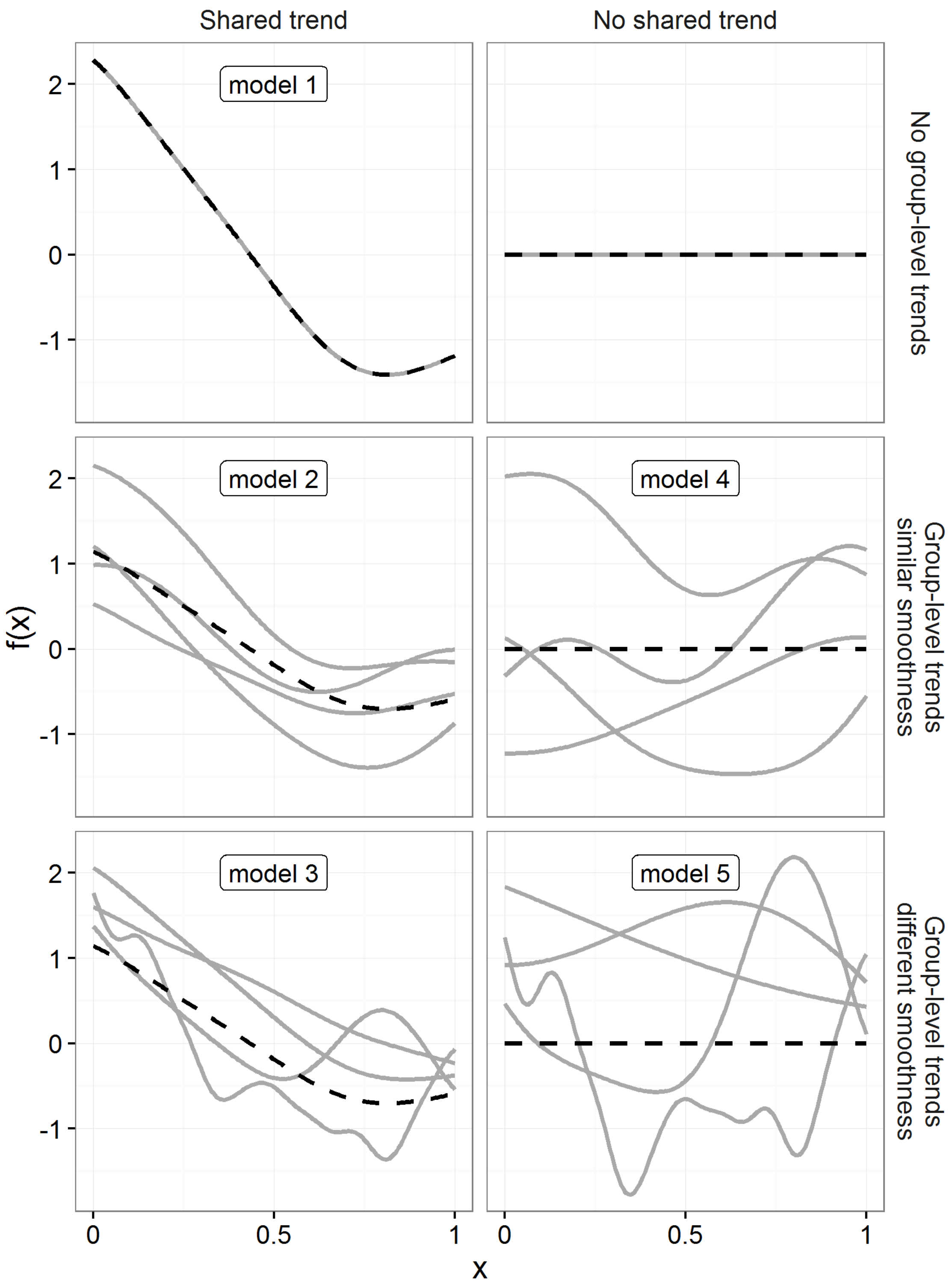

Shared trend
No shared trend
No group-level trends
Group-level trends similar smoothness
Group-level trends different smoothness
model 1
model 2
model 3
model 4
model 5
f(x)
x
2
1
0
-1
0
0.5
1

Figure 5:

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 concentration, and a random effect for species to model species-specific intercepts.[1] 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. We assume that $\log_e(\texttt{uptake}_i)$ is normally distributed.

DLM: not sure if this note is necessary...

DLM: need to justify why we use log concentration not just concentration? (could just cite?)

In R we can write our model as:

```r
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, with the addition of `s()` terms to include one-dimensional or isotropic multidimensional smooths. The first argument to `s()` is 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=...`.

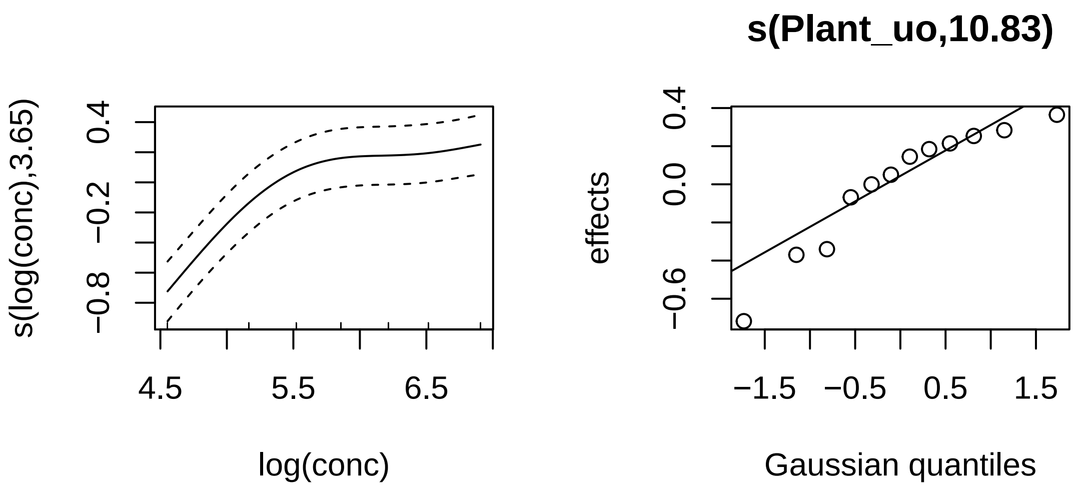


Figure [BLAH CROSSREF] has two panels: the left showing the estimated global functional relationship, and the right shows a quantile-quantile plot of the estimates effects vs Guassian quantiles, which can be used to check our model.

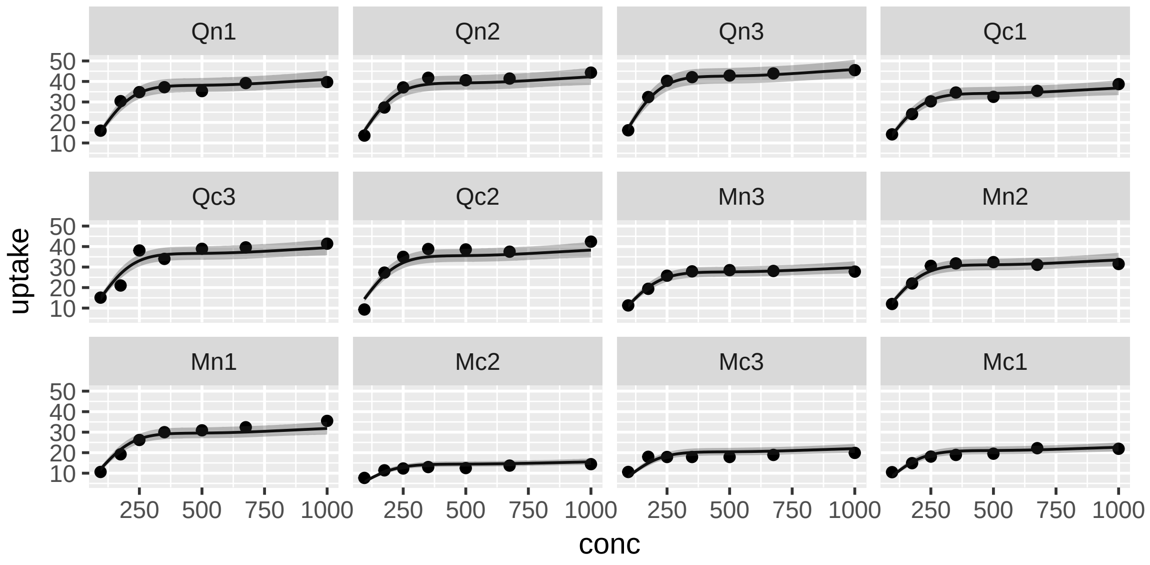
DLM: add more to plot description!

[1] 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 by simple random effects): $ln(f_1(x)) = ln(\alpha) + ln(f_2(x))$. We have also ln-transformed concentration. Since the concentration-uptake relationship changes rapidly at low concentration values and slowly at higher values, estimating the relationship without log-transforming it can lead to a small estimated penalty value, and an overly wiggly function at higher concentration values.)

Looking at the effects by term is useful but we are often interested in fitted values or predictions our models. This can be useful to construct plots (like those in Figure [XXXX]). The next block of code shows how you could plot this to illustrate inter-plant variation in the functional response with estimated functional variability, plotting untransformed uptake and concentration to make the figure easier to comprehend. You can see the effect log-transforming concentration has on model fits; even though plants Mc1 – Mc3 show a much flatter response in untransformed space, the same functional response fits relatively well after accounting for the random effect (a multiplicative difference between functional responses).

DLM: what does "estimated functional variability" mean above??

```r
# setup prediction data
CO2_mod1_pred <- with(CO2,
                      expand.grid(conc=seq(min(conc), max(conc), length=100),
                                  Plant_uo=levels(Plant_uo)))
# make the prediction, add this and a column of standard errors to the prediction
# data.frame. Predictions are on the log scale.
CO2_mod1_pred <- cbind(CO2_mod1_pred,
                       predict(CO2_mod1, CO2_mod1_pred, se.fit=TRUE))

# make the plot
ggplot(data=CO2, aes(x=conc, y=uptake, group=Plant_uo)) +
    facet_wrap(~Plant_uo) +
    geom_point() +
    geom_line(aes(y=exp(fit)), data=CO2_mod1_pred) +
    geom_ribbon(aes(ymin=exp(fit - 2*se.fit), ymax=exp(fit + 2*se.fit), x=conc),
                data=CO2_mod1_pred, alpha=0.3, inherit.aes=FALSE)
```



We can include interactions in an `s()` term via isotropic smooths such as thin plate regression splines 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 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 a penalty on the null space.

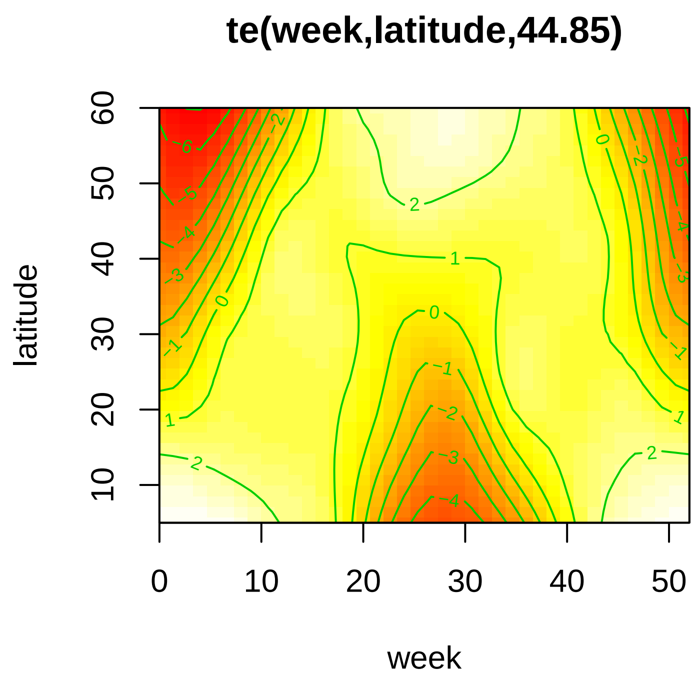
For our bird example, we want to look at the interaction between location and time, so for this we setup the model as:

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

where we assume that $\texttt{count}_i \sim$ Poisson. For the smooth term, f, we employ a tensor product of latitude and week, using thin plate regression spline for the marginal latitude effects, and a cyclic cubic 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. For simplicity of code we will exclude a species-specific random effect like the one we had in the CO_2 uptake example[2].

DLM: can we just say we're ignoring species here?

```r
library(tidyr)
library(viridis) # for color plotting

bird_move <- read.csv("../data/bird_move.csv") # load data

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

plot(bird_mod1, pages=1, scheme=2, rug=FALSE)
box()
```



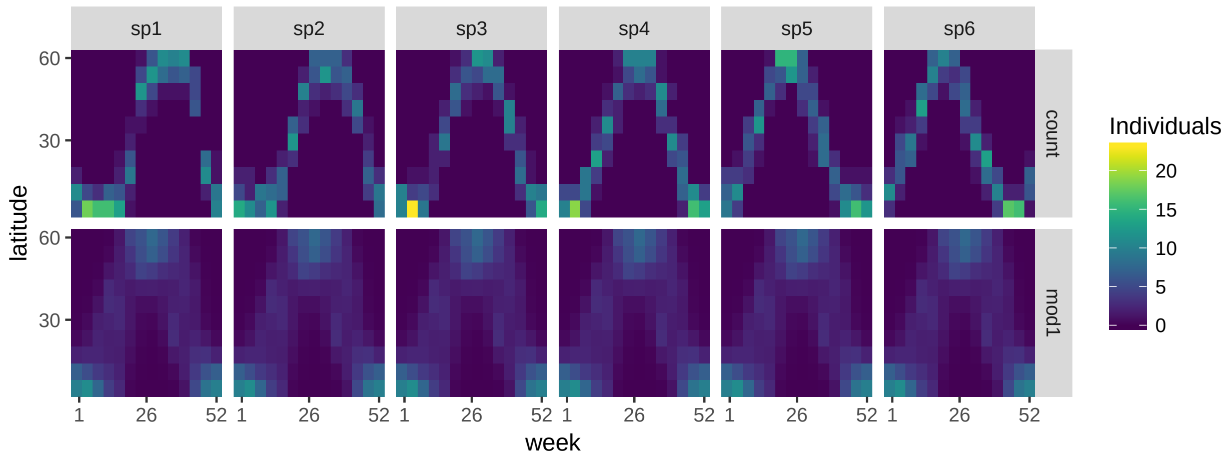
DLM: this is part of the figure caption: "The default plot for this GAM illustrates the average log-abundance of all bird species at each latitude for each week, with yellow colours indicating more individuals and red colours fewer."

Figure [BLAHXXX] 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

[2]If we included it, it could be interpreted as modelling variation in average observability between species. However, as we cheated and know what the data generating process looks like, we do not need to worry about adding it here.

large amount of variability in the timing of migration. The source of this variability is apparent when the migration timing of each species is plotted in conjunction with the model fit:

```r
bird_move <- transform(bird_move, mod1=predict(bird_mod1, type="response"))
#gets the fitted model values, at the response scale

bird_move_plot <- gather(bird_move, key=model, value=value, count, mod1)
#combines observed and fitted estimates into a single column called "value"

ggplot(bird_move_plot, aes(x=week, y=latitude, fill=value)) +
    geom_raster() +
    facet_grid(model ~ species)+
    scale_fill_viridis() +
    scale_x_continuous(expand=c(0, 0), breaks=c(1, 26, 52)) +
    scale_y_continuous(expand=c(0, 0), breaks=c(0, 30, 60)) +
    labs(fill="Individuals")
```



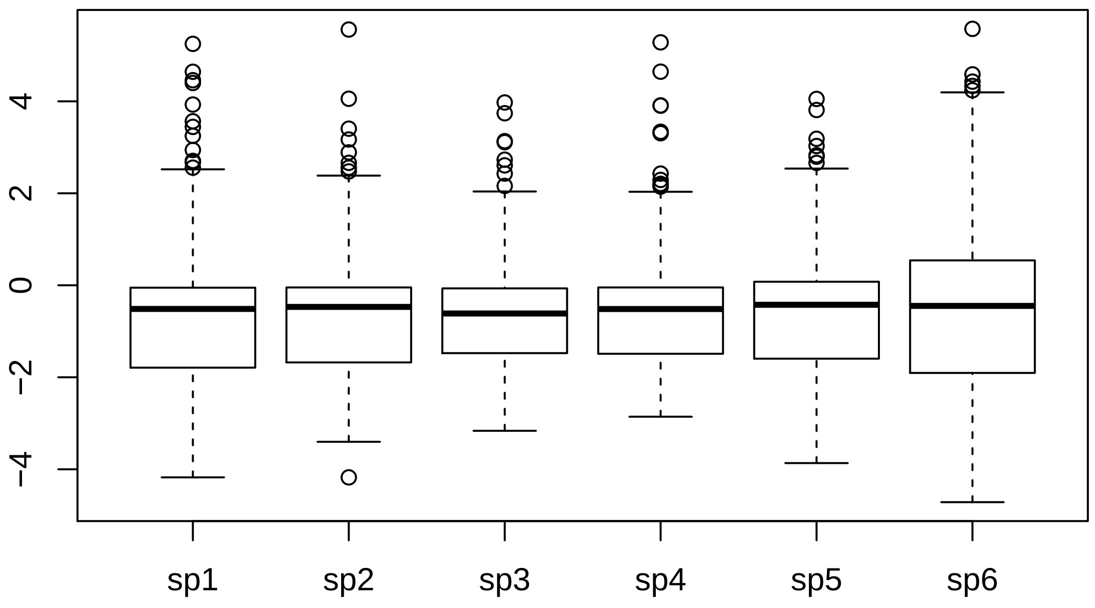
DLM: this is part of the figure caption: "Here the top row denotes the observed counts of each species (with color indicating abundance in that location in that week), and the bottom indicates the model fit."

DLM: I think that the bottom row of this graphic is a big of a waste of ink, as it's the same information as in the previous plot, but duplicated 6 times. I'd prefer to see the top row as EDA in the intro to the data.

All six species in Figure [BLAHXXX] show relatively precise migration patterns, but they differ in the timing of when they leave their winter grounds and the time they spend at their summer grounds. Averaging over all of this variation results in a relatively imprecise (diffuse) average estimate of migration timing (bottom row). 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.

DLM: not sure about this interpretation of the plots – I agree there is heterogeneity caused by species, but not sure this is exactly what is shown here. In some sense smoothers are wrong everywhere, I think the plot just shows this property... Maybe the plot below shows the heterogeneity issue?

```r
resid_plot <- bird_move
resid_plot$residuals <- residuals(bird_mod1)
boxplot(residuals~species, data=resid_plot)
```



Boxplot of residuals from the bird model. If our model fitted well we would expect that the boxplots would all be about the same width. Instead we see that the spread of residuals is much higher for species 1 and 6.

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 allows for inter-group variation in responses. 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)[3]. As there can be issues of co-linearity between the global smooth term and the group-specific terms (see section V for more details), it is generally necessary to use a smoother with a more restricted null space than the global smooth; for thin plate splines this can be done by setting m=2 for the global smooth and m=1 for the group smooth [cite Wieling paper here]. e.g.: `y~s(x,bs="tp",m=2)+s(x,fac,bs="fs",m=1,xt=list(bs="tp"))`. Appendix A illustrates another way to setup these models.

We modify our previous CO_2 model as follows:

[3]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.

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

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, bs="tp") +
                              s(log(conc), Plant_uo, k=5,  bs="fs", m=1),
                data=CO2, method="REML")
```

```
source("../code/functions.R")

CO2_mod2 <- gam(log(uptake) ~ s(log(conc), k=5, m=2, bs="tp") +
                              s(log(conc), Plant_uo, k=5,  bs="fs", m=1),
                data=CO2, method="REML")

plot(CO2_mod2, page=1, seWithMean=TRUE)
```

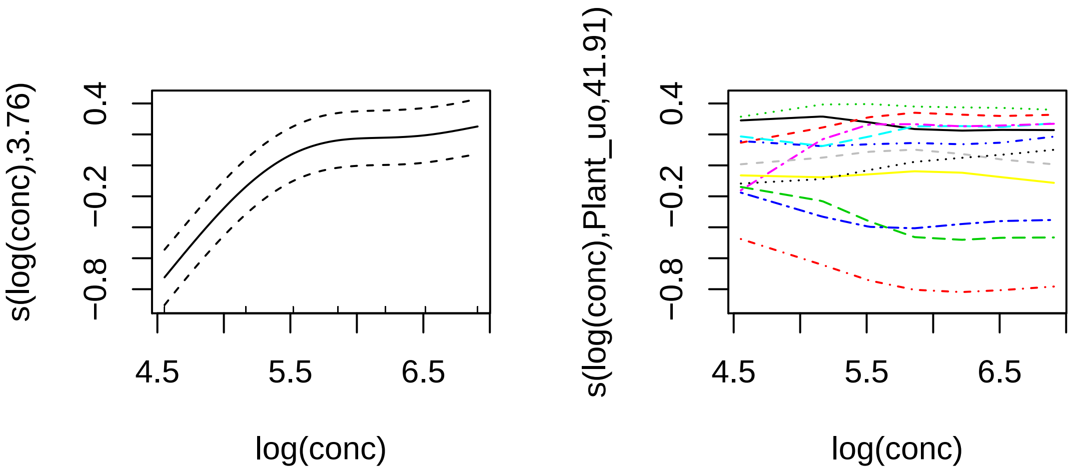
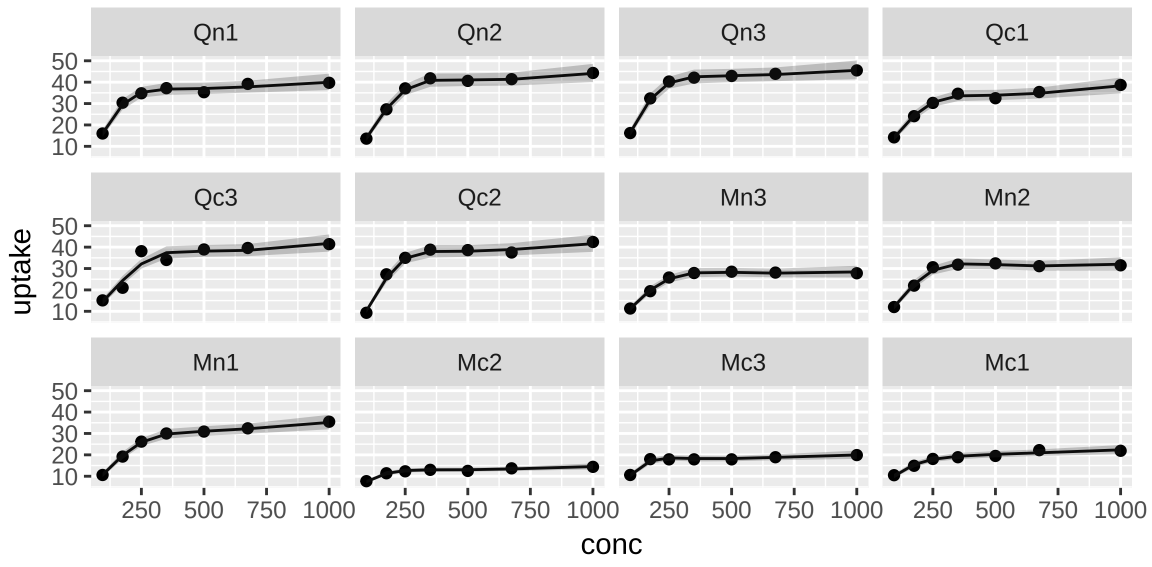


Figure BLAHXXX shows the global function (left) and group-specific deviations from the global function (right) for `CO2_mod2`. The plots of group-specific smooths indicate that plants differ not only in average log-uptake (which would correspond to each plant having a straight line at different levels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

```
CO2_mod2_pred <- predict(CO2_mod2, se.fit=TRUE)
CO2$mod2 <- CO2_mod2_pred$fit
CO2$mod2_se <- CO2_mod2_pred$se.fit

ggplot(data=CO2, aes(x=conc, y=uptake, group=Plant_uo)) +
  facet_wrap(~Plant_uo) +
  geom_point() +
  geom_line(aes(y=exp(mod2))) +
  geom_ribbon(aes(ymin=exp(mod2-2*mod2_se),
                  ymax=exp(mod2+2*mod2_se)), alpha=0.25)
```

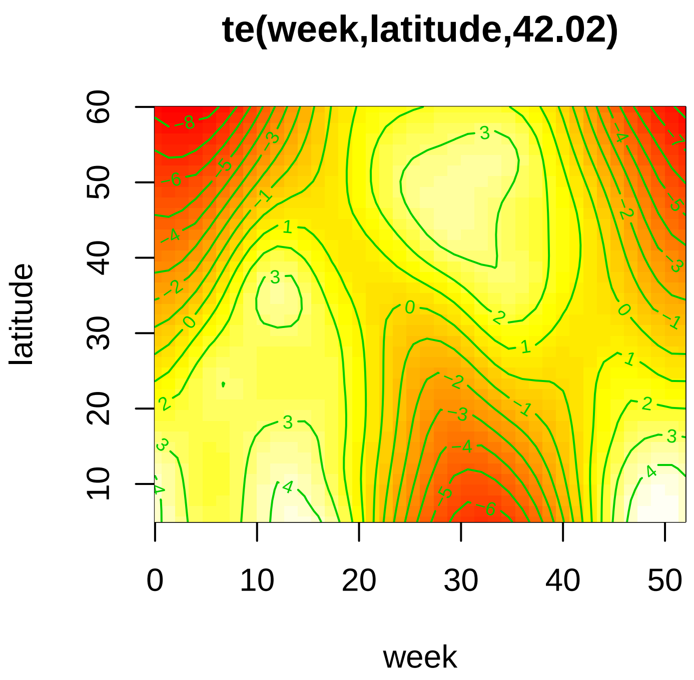


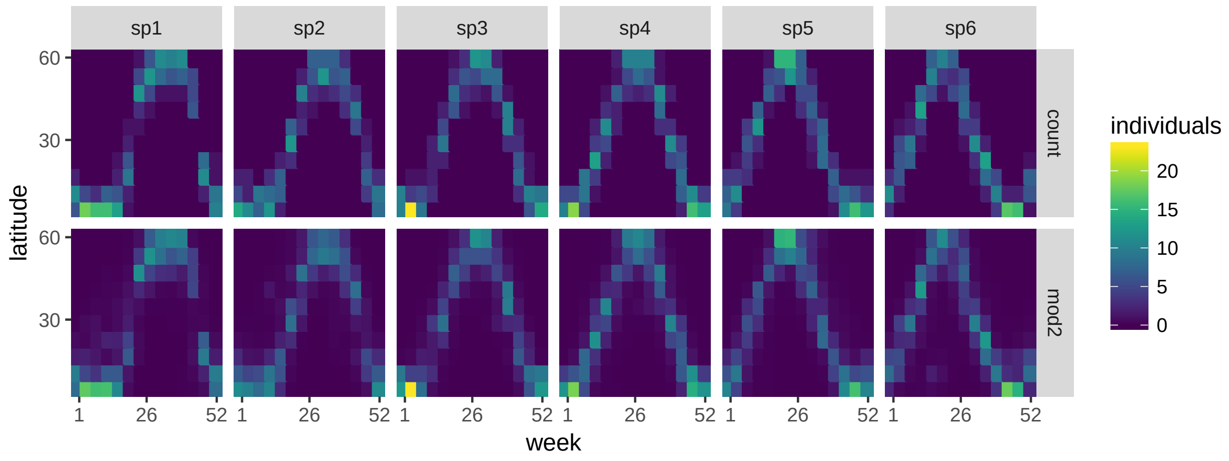
The "`fs`"-based approach mentioned above does not work for higher-dimensional tensor product smooths (if one is willing to use thin plate regression splines for the multivariate smooth then one can use "`fs`"). Instead, the group-specific term can be specified with a tensor product of the continuous smooths and a random effect for the grouping parameter. This term will again have a separate set of basis functions for each group, one penalty for the smooth term, and a second penalty drawing all basis functions toward zero[4]. e.g.: `y~te(x1,x2,bs="tp",m=2)+te(x1,x2, fac,bs=c("tp","tp","fs"),m=1)`. We illustrate this approach below on the bird migration data.

DLM: is the "fs" above supposed to be a "re"?

```
bird_mod2 <- gam(count ~ te(week, latitude, bs=c("cc", "tp"),
                               k=c(10, 10), m=c(2, 2)) +
                            te(week, latitude, species, bs=c("cc", "tp", "re"),
                               k=c(10, 10, 6), m=c(1, 1, 1)),
                    data=bird_move, method="REML", family=poisson)

plot(bird_mod2, page=1, scheme=2, rug=FALSE, seWithMean=TRUE)
```

[4]Note that this differs from the "fs" penalty, which assigned one penalty per null space term.





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=fac` argument in the `s()` function. 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 two 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 these smooth terms automatically (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 (`bs="ts"`, for “thin plate with shrinkage”), as this method does not automatically penalize the nullspace, so there is potential for co-linearity issues between unpenalized components of the global and group-level smoothers.

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, bs="ts", m=1) +
                             s(Plant_uo, bs="re", k=12),
               data= CO2, method="REML")
```

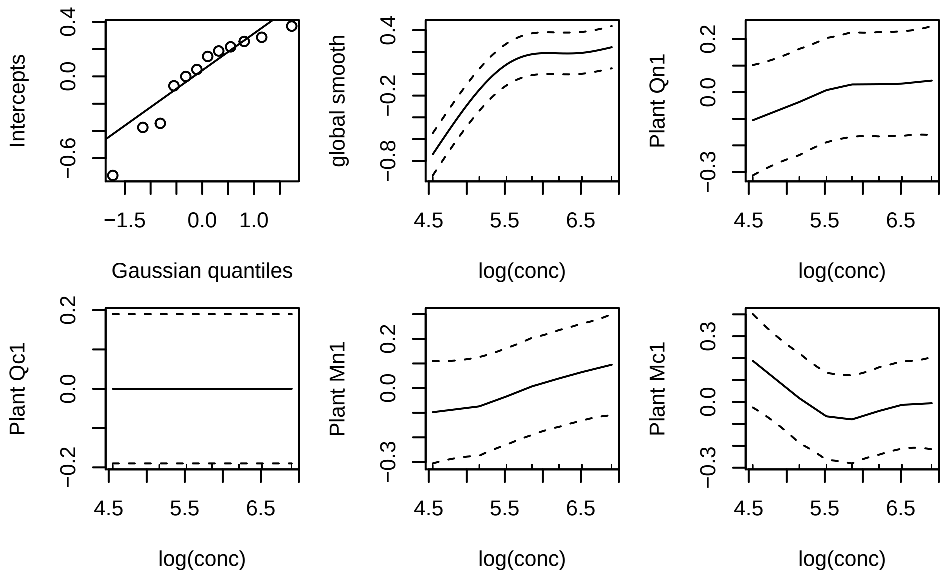


Figure BLAHXXX shows a subsample of the group-specific smooths from this model, to prevent crowding. It is appearent from this that some groups (e.g. `Qc1`) have very similar shapes to the global smooth (differing only in intercept), others do differ from the global trend, with higher uptake at low concentrations 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=fac` terms work 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, bs= c("cc", "tp"),
                              k=c(10, 10), m=c(1, 1), by=species),
                 data=bird_move, method="REML", family=poisson)
```

DLM: show some plots of this?

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. (Plots are very similar to 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 ~ te(week, latitude, species, bs=c("cc", "tp", "re"),
                             k=c(10, 10, 6), m=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.)

```
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", "tp"),
                             k=c(10, 10), m = 2),
                 data=bird_move, method="REML", family=poisson)
```

Where group-level smooths are coded using the `by=fac` argument in the `s()` function, ; 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 not set the basis functions for the first grouping level to zero. In model 3 (with an ungrouped smooth included) 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.

V: Modelling issues

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.[5] 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 [CITE: Effron and Morris 1977].

[5]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).

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. 6a), 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. 6b).

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 the model evalution techniques discussed in Section IV 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, implying a substantial improvement in fit by allowing smoothness to vary. However, it may be the case that there is 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 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).

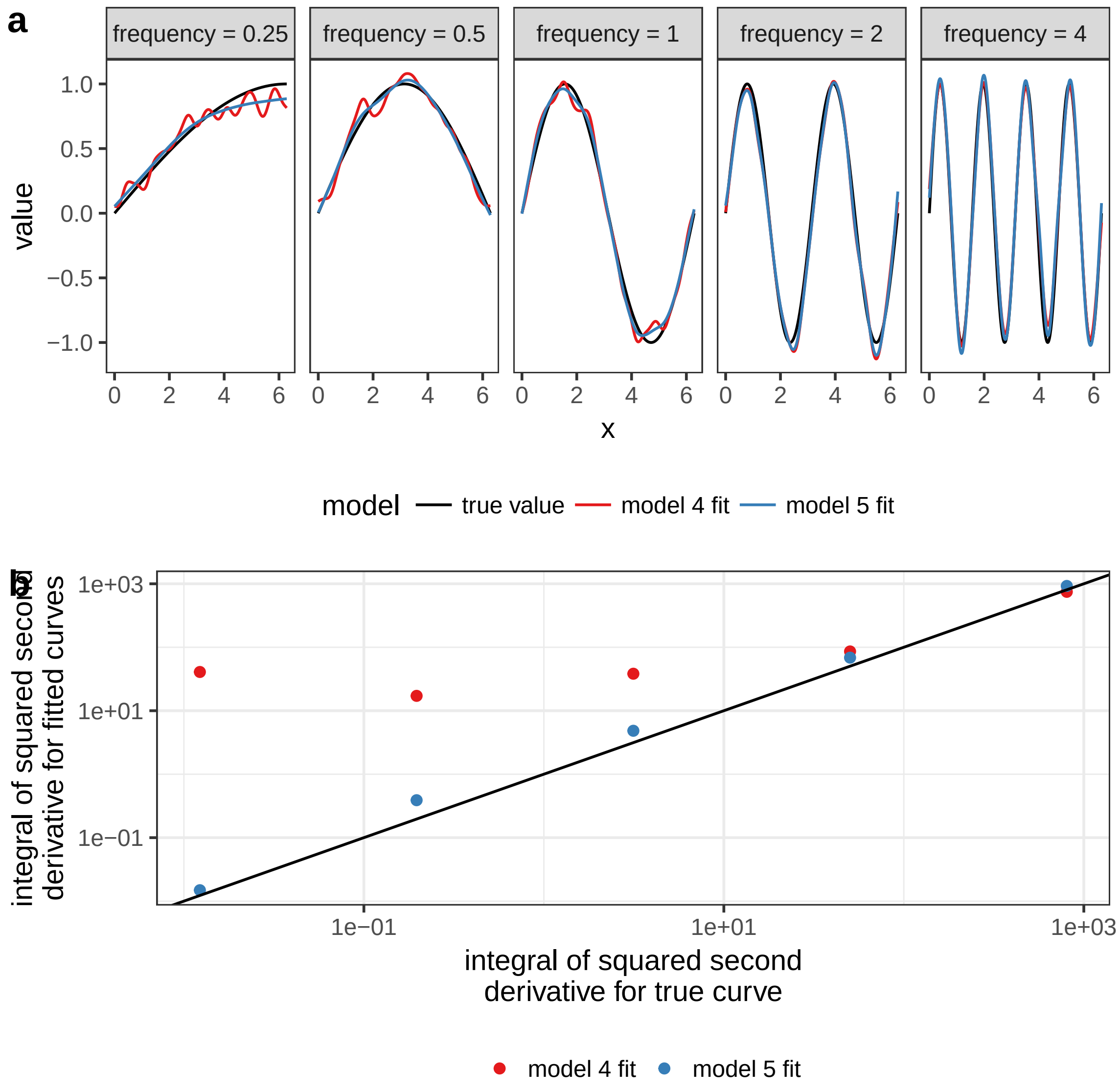


Figure 6: plotting example

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 section IV) 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 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[6] 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[7]. 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 1). 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 [CITE].

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 [CITE 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 1). 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 1B).

Alternative formulations: bam, gamm, and gamm4 (with a brief foray into Bayes)

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.

The first tool available, that requires the least changes to your code compared to the base `gam` function, is the

[6]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.

[7]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 1, 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 1:

model	relative time	# of terms	
		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	140	624	14
4	100	541	3
5	66	535	12

`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 [*CITE*]. 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 7), 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 7). 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 7)[8].

[8]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.

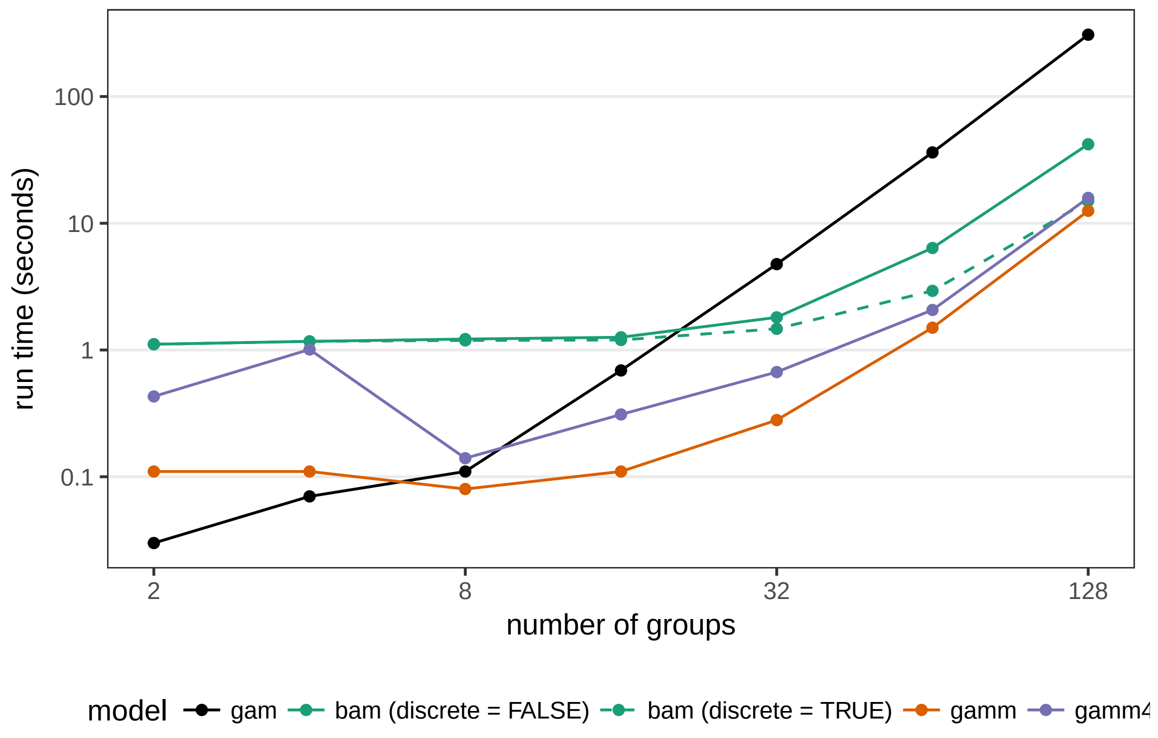


Figure 7: Elapsed time to estimate the same model using each of the four approaches. Each data set was generated with 20 observations per group using a unimodal global function and random group-specific functions consisting of an intercept, a quadratic term, and logistic trend for each group. Observation 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.

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 [*CITE*] 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)
```

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 1). 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[9]. When using thin-plate

[9]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

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 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

VI: Examples

EJP: I think we should aim for 3-4 good examples here, highlighting different aspects of the model fitting problem. The example I'll be showing is a set of zooplankton community time series data, where multiple species were tracked throughout the year for a period of roughtly 20 years. This example can highlight both testing models 4/5 (for comparing different species' season cycles) and models 2/3 (for testing for differences between years for a single species). I think we need at least one example showing how to use these methods for multivariate regression (e.g. spatial analysis), and potentially an example showing how to these models work for non-normal data, and for including other covariates. In all examples, I think we should focus on how to fit each data set, visualize models, and compare different model fits.

In this final section, we will go through a few example analyses, to highlight how to use these models in practice, and to illustrate how to fit, test, and visualize each model.

Example 1: Inter- and intra-specific variation in zooplankton seasonal cycles over time

This first example 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 here [*CITE*]). 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).

This is what the data looks like:

```r
zooplankton = read.csv("../data/zooplankton_example.csv")

str(zooplankton)
```

```
## 'data.frame':    5848 obs. of  6 variables:
```

penalty added to the null space: bs=“tp”, “cs”, or “ps” have this property.

```
##  $ day          : int  10 10 10 10 10 10 10 10 12 12 ...
##  $ year         : int  1980 1980 1980 1980 1980 1980 1980 1980 1984 1984 ...
##  $ lake         : Factor w/ 4 levels "Kegonsa","Mendota",..: 2 2 2 2 2 2 2 2 1 1 ...
##  $ taxon        : Factor w/ 8 levels "Calanoida copepodites",..: 1 2 3 4 5 6 7 8 1 2 ...
##  $ density      : num  5000 28000 3000 6000 1867000 ...
##  $ density_scaled: num  -1.4173 -0.0714 -2.3837 0.1945 1.2613 ...
```

```
levels(zooplankton$taxon)
```

```
## [1] "Calanoida copepodites"     "Chydorus sphaericus"
## [3] "Cyclopoida copepodites"    "Daphnia mendotae"
## [5] "Diacyclops thomasi"        "Keratella cochlearis"
## [7] "Leptodiaptomus siciloides" "Mesocyclops edax"
```

```
levels(zooplankton$lake)
```

```
## [1] "Kegonsa" "Mendota" "Menona"  "Waubesa"
```

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:

```
library(mgcv)
zoo_train = subset(zooplankton, year%%2==0) #the modulus (%%) finds the remainder after division by  th

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[10]. As this is seasonal data, we will use cyclic smoothers as the basis for seasonal dynamics.

```
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)), #we need to specify the start and end knots for day
           method = "ML" #We'll use ML as we are comparing models that differ in fixed effects
           )

summary(zoo_comm_mod4)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## density_scaled ~ s(day, taxon, bs = "fs", k = 10, xt = list(bs = "cc"))
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.02597    0.02621  -0.991    0.322
##
## Approximate significance of smooth terms:
##                edf Ref.df     F p-value
## s(day,taxon) 54.52     71 18.44  <2e-16 ***
```

[10]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.

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.308   Deviance explained = 32.1%
## -ML = 3615.4  Scale est. = 0.64168    n = 2947
```

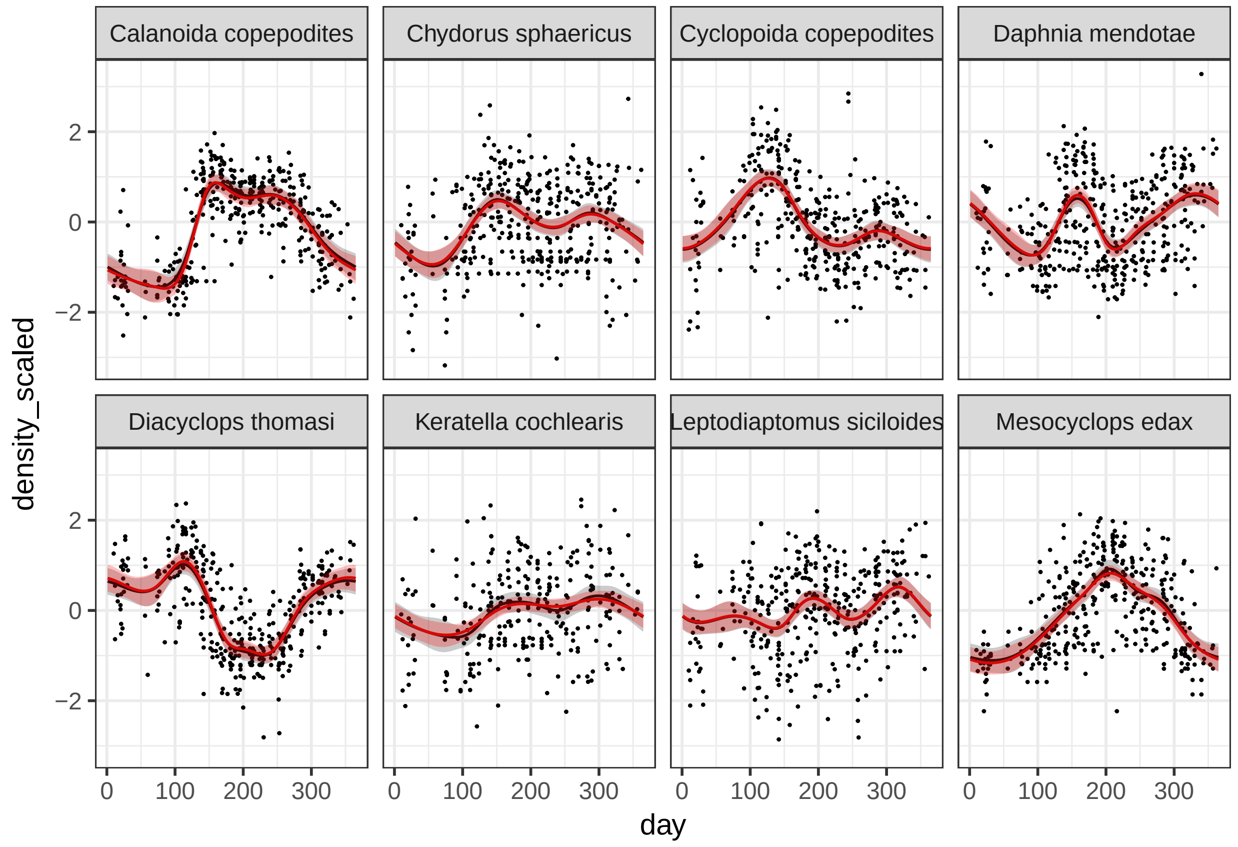
```
# as all of the model features except the formula are the same in model 4&5,
# we just use the update function to refit the model with the new formula
zoo_comm_mod5 = update(zoo_comm_mod4,
                       formula = density_scaled~s(day, by=taxon, k=10,bs="cc"))

summary(zoo_comm_mod5)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## density_scaled ~ s(day, by = taxon, k = 10, bs = "cc")
##
## Parametric coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.001563   0.014736  -0.106    0.916
##
## Approximate significance of smooth terms:
##                                         edf Ref.df      F  p-value
## s(day):taxonCalanoida copepodites     6.968      8 44.248  < 2e-16 ***
## s(day):taxonChydorus sphaericus       5.595      8  8.933 2.47e-15 ***
## s(day):taxonCyclopoida copepodites    5.806      8 21.013  < 2e-16 ***
## s(day):taxonDaphnia mendotae          6.941      8 15.977  < 2e-16 ***
## s(day):taxonDiacyclops thomasi        6.663      8 38.303  < 2e-16 ***
## s(day):taxonKeratella cochlearis      3.904      8  3.527 1.35e-06 ***
## s(day):taxonLeptodiaptomus siciloides 6.223      8  5.891 3.36e-09 ***
## s(day):taxonMesocyclops edax          5.137      8 27.439  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =   0.31   Deviance explained = 32.1%
## -ML = 3601.5  Scale est. = 0.63978    n = 2947
```

We can see that both models have very similar fits, with an adusted R^2 of 0.308 for model 4 and 0.31 for model 5. 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 show very similar fit to the data:

```
library(ggplot2)
library(dplyr)

#Create synthetic data to use to compare predictions
zoo_plot_data = expand.grid(day = 1:365, taxon = factor(levels(zoo_train$taxon)))


zoo_mod4_fit = predict(zoo_comm_mod4, zoo_plot_data, se.fit = T)
zoo_mod5_fit = predict(zoo_comm_mod5, zoo_plot_data, se.fit = T)

zoo_plot_data$mod4_fit = as.numeric(zoo_mod4_fit$fit)
zoo_plot_data$mod5_fit = as.numeric(zoo_mod5_fit$fit)
```

```r
zoo_plot_data$mod4_se = as.numeric(zoo_mod4_fit$se.fit)
zoo_plot_data$mod5_se = as.numeric(zoo_mod5_fit$se.fit)

zoo_plot = ggplot(zoo_plot_data, aes(x=day))+
  facet_wrap(~taxon, nrow = 2)+
  geom_point(data= zoo_train, aes(y=density_scaled),size=0.1)+
  geom_line(aes(y=mod4_fit))+
  geom_line(aes(y=mod5_fit),color="red")+
  geom_ribbon(aes(ymin = mod4_fit - 2*mod4_se, ymax = mod4_fit + 2*mod4_se), alpha=0.25)+
  geom_ribbon(aes(ymin = mod5_fit - 2*mod5_se, ymax = mod5_fit + 2*mod5_se), alpha=0.25, fill="red")+
  theme_bw()

print(zoo_plot)
```



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*:

```r
#Getting the out of sample predictions for both models:
zoo_test$mod4 = as.numeric(predict(zoo_comm_mod4,zoo_test))
zoo_test$mod5 = as.numeric(predict(zoo_comm_mod5,zoo_test))

# We'll look at the correlation between fitted and observed values for all species:
zoo_test_summary = zoo_test %>%
  group_by(taxon)%>%
  summarise(mod4_cor = round(cor(density_scaled, mod4),2),
            mod5_cor = round(cor(density_scaled, mod5),2))

print(zoo_test_summary)
```

```
## # A tibble: 8 x 3
##   taxon                     mod4_cor mod5_cor
##   <fct>                        <dbl>    <dbl>
## 1 Calanoida copepodites        0.690    0.700
## 2 Chydorus sphaericus          0.360    0.350
## 3 Cyclopoida copepodites       0.540    0.530
## 4 Daphnia mendotae             0.340    0.350
## 5 Diacyclops thomasi           0.810    0.810
## 6 Keratella cochlearis         0.230    0.200
## 7 Leptodiaptomus siciloides    0.330    0.330
## 8 Mesocyclops edax             0.570    0.570
```

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.

```
daphnia_train = subset(zoo_train,taxon=="Daphnia mendotae")
daphnia_test = subset(zoo_test,taxon=="Daphnia mendotae")

zoo_daph_mod1 = gam(density_scaled~s(day, bs="cc",k=10),
           data=daphnia_train,
           knots = list(day =c(1,365)), #we need to specify the start and end knots for day
           method = "ML" #We'll use ML as we are comparing models that differ in fixed effects
           )

summary(zoo_daph_mod1)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## density_scaled ~ s(day, bs = "cc", k = 10)
##
## Parametric coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.309e-16  4.245e-02       0        1
##
## Approximate significance of smooth terms:
##          edf Ref.df     F p-value
## s(day) 6.824      8 13.96  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.217   Deviance explained = 23.1%
## -ML = 519.41  Scale est. = 0.7262    n = 403
```

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

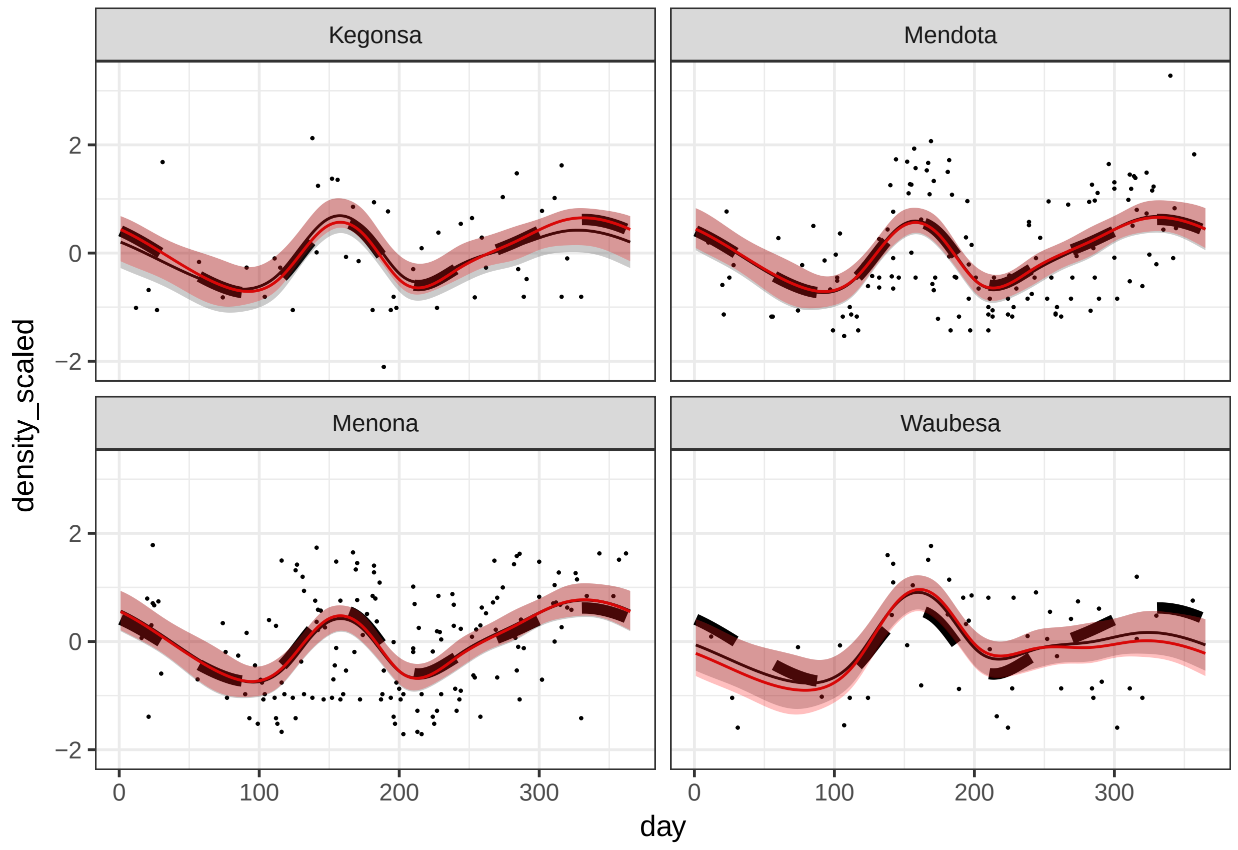
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## density_scaled ~ s(day, bs = "cc", k = 10) + s(day, lake, k = 10,
##     bs = "fs", xt = list(bs = "cc"))
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.007772   0.043919   0.177     0.86
##
## Approximate significance of smooth terms:
##                edf Ref.df      F p-value
## s(day)       6.797      8 10.881  <2e-16 ***
## s(day,lake) 5.348     35  0.432  0.0039 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.246   Deviance explained = 26.9%
## -ML = 516.48  Scale est. = 0.69947   n = 403
```

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

summary(zoo_daph_mod3)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## density_scaled ~ s(day, bs = "cc", k = 10) + s(day, by = lake,
##     k = 10, bs = "cc")
##
## Parametric coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.0007017  0.0416980  -0.017    0.987
##
## Approximate significance of smooth terms:
##                           edf Ref.df      F p-value
## s(day)             6.8552361      8 13.985 < 2e-16 ***
## s(day):lakeKegonsa 0.0488015      8  0.006 0.34599
## s(day):lakeMendota 0.0005693      8  0.000 0.73813
## s(day):lakeMenona  0.9270931      8  0.198 0.16129
## s(day):lakeWaubesa 2.2353408      8  1.454 0.00116 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.246   Deviance explained = 26.5%
## -ML = 514.72  Scale est. = 0.69938   n = 403
```

The AIC values indicate that both model 2 and 3 are better fits than model 1, but models 2 and 3 have similar fits to one another. There does not seem to be a large amount of inter-lake variability (all three models have similar adjusted R^2), and model 3 indicates that only Lake Waubesa deviates substantially from the overall dynamics. The plots for all three models (model 1 as dashed line, model 2 in black and model 3 in red) show that Medota 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 (in black) shows that it is still detectable in Lake Kegonsa if we do not need to fit a separate penalty for each lake.

```r
library(ggplot2)
library(dplyr)

#Create synthetic data to use to compare predictions
daph_plot_data = expand.grid(day = 1:365, lake = factor(levels(zoo_train$lake)))


daph_mod1_fit = predict(zoo_daph_mod1, daph_plot_data, se.fit = T)
daph_mod2_fit = predict(zoo_daph_mod2, daph_plot_data, se.fit = T)
daph_mod3_fit = predict(zoo_daph_mod3, daph_plot_data, se.fit = T)


daph_plot_data$mod1_fit = as.numeric(daph_mod1_fit$fit)
daph_plot_data$mod2_fit = as.numeric(daph_mod2_fit$fit)
daph_plot_data$mod3_fit = as.numeric(daph_mod3_fit$fit)

daph_plot_data$mod1_se = as.numeric(daph_mod1_fit$se.fit)
daph_plot_data$mod2_se = as.numeric(daph_mod2_fit$se.fit)
daph_plot_data$mod3_se = as.numeric(daph_mod3_fit$se.fit)

daph_plot = ggplot(daph_plot_data, aes(x=day))+
  facet_wrap(~lake, nrow = 2)+
  geom_point(data= daphnia_train, aes(y=density_scaled),size=0.1)+
  geom_line(aes(y=mod1_fit),linetype=2, size=2)+
  geom_line(aes(y=mod2_fit),color="black")+
  geom_line(aes(y=mod3_fit),color="red")+
  geom_ribbon(aes(ymin = mod2_fit - 2*mod2_se, ymax = mod2_fit + 2*mod2_se), alpha=0.25)+
  geom_ribbon(aes(ymin = mod2_fit - 2*mod3_se, ymax = mod2_fit + 2*mod2_se), alpha=0.25, fill="red")+
  theme_bw()

print(daph_plot)
```



In this case, model 2 is able to predict as good or better out of sample as model 1 or 3, 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 correlation of only 0.11 between predicted and observed densities), indiciating that this model may be be missing substantial year-to-year variability in *D. mendotae* dynamics:

```
#Getting the out of sample predictions for both models:
daphnia_test$mod1 = as.numeric(predict(zoo_daph_mod1,daphnia_test))
daphnia_test$mod2 = as.numeric(predict(zoo_daph_mod2,daphnia_test))
daphnia_test$mod3 = as.numeric(predict(zoo_daph_mod3,daphnia_test))

# We'll look at the correlation between fitted and observed values for all species:
daph_test_summary = daphnia_test %>%
  group_by(lake)%>%
  summarise(mod1_cor = round(cor(density_scaled, mod1),2),
            mod2_cor = round(cor(density_scaled, mod2),2),
            mod3_cor = round(cor(density_scaled, mod3),2))

print(daph_test_summary)
```

```
## # A tibble: 4 x 4
##   lake    mod1_cor mod2_cor mod3_cor
##   <fct>      <dbl>    <dbl>    <dbl>
## 1 Kegonsa   0.0800    0.110   0.0900
## 2 Mendota   0.420     0.430   0.430
## 3 Menona    0.350     0.400   0.390
## 4 Waubesa   0.290     0.290   0.270
```

Bibliography

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.

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.

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.

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

Wood, Simon N. 2006. *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.

Wood, Simon N., Fabian Scheipl, and Julian J. Faraway. 2012. "Straightforward Intermediate Rank Tensor Product Smoothing in Mixed Models." *Statistics and Computing* 23 (3): 341–60. doi:10.1007/s11222-012-9314-z.