Extended smooth Modelling and R

Simon Wood
Mathematical Sciences, University of Bath, U.K.

Generalized Additive Models

- A *generalized additive models* relates a response variable y_i to some predictors x_{ji} via something like:

$$g\{\mathbb{E}(y_i)\} = \eta_i = \mathbf{X}_i^* \boldsymbol{\beta}^* + f_1(x_{1i}) + f_2(x_{2i}, x_{3i}) + f_3(x_{4i}) + \cdots$$

  - g is a known *link function*.
  - y_i independent with some exponential family distribution.
  - f_j are smooth unknown functions (subject to centering conditions).
  - $\mathbf{X}^* \boldsymbol{\beta}^*$ is parametric bit.
- In practice the f_j can be represented using basis expansions, with over-fit avoided by quadratic penalties on the basis coefficients.
- So a GAM is simply a quadratically penalized GLM.

In more detail

- We write $f_j(x) = \sum_k \beta_{jk} b_{jk}(x)$ where the basis functions $b_{jk}(x)$ are known, but the β_{jk} must be estimated.
- Function wiggliness is measured by $\beta_j^{\mathrm{T}} \mathbf{S}_j \beta_j$, where the elements of $\mathbf{S}_j$ are known and derive from the $b_{jk}(x)$.
- So the GAM is $g(\mathbb{E}(y_i)) = \mathbf{X}_i \boldsymbol{\beta}$ where $\mathbf{X}$ contains $\mathbf{X}^*$ + the basis functions evaluated at the predictor variable values.
- $\boldsymbol{\beta}$ is estimated as the minimizer of the penalized deviance

$$D(\boldsymbol{\beta}) + \sum_j e^{\rho_j} \boldsymbol{\beta}^{\mathrm{T}} \mathbf{S}_j \boldsymbol{\beta}$$

where the ρ_j are log smoothing parameters and the $\mathbf{S}_j$ have been padded with zeroes.

Actual GAM estimation

- Given $\boldsymbol{\rho}$, the penalized deviance

$$D(\boldsymbol{\beta}) + \sum_j e^{\rho_j} \boldsymbol{\beta}^{\mathrm{T}} \mathbf{S}_j \boldsymbol{\beta}$$

  is minimized by Penalized IRLS (Fisher scoring or Newton).
- In general $\boldsymbol{\rho} = \mathbf{L}\boldsymbol{\rho}_u$ where $\mathbf{L}$ known, and $\boldsymbol{\rho}_u$ is estimated by GCV or AIC type criteria.
- Alternatively the smooth terms can be viewed as random effects, the ρ_j as inverse variance components, and $\boldsymbol{\rho}_u$ estimated by (Laplace approximate) REML.
- Notice that the GAM estimation problem applies to a rather wider class of models than GAMs...

What is a smoother?

- In this GAM estimation framework, a smoother is simply a basis and associated quadratic penalty.
- To estimate the GAM all that we need from a smoother is a model matrix of evaluated basis functions, $\mathbf{X}_j$, say, and one or more penalty coefficient matrix, $\mathbf{S}_j$, say.
- All the details of how the basis is constructed are irrelevant for estimation.
- For prediction from the GAM, we will need to be able to produce further matrices of basis functions evaluated at new predictor values, but nothing else.
- So GAM modelling routines need exactly the same, rather limited, evaluated quantities from any smoother, and no smoother specific details.

Smooth classes in R

- The R/S language makes it very easy to exploit the inherent modularity of the GAM representation.
- ...`mgcv` is an R package for generalized additive modelling that tries to use this fact.
- Smooths can be set up as *objects* with particular *classes*, which have associated *methods*.
- `mgcv`'s GAM modelling routines then need know nothing about the internal details of how a smoother is constructed, but merely that it is a smooth.
- Equally, adding a new smooth class is easy: nothing in the modelling functions need change in order to do so.

e.g. penalized cubic regression spline

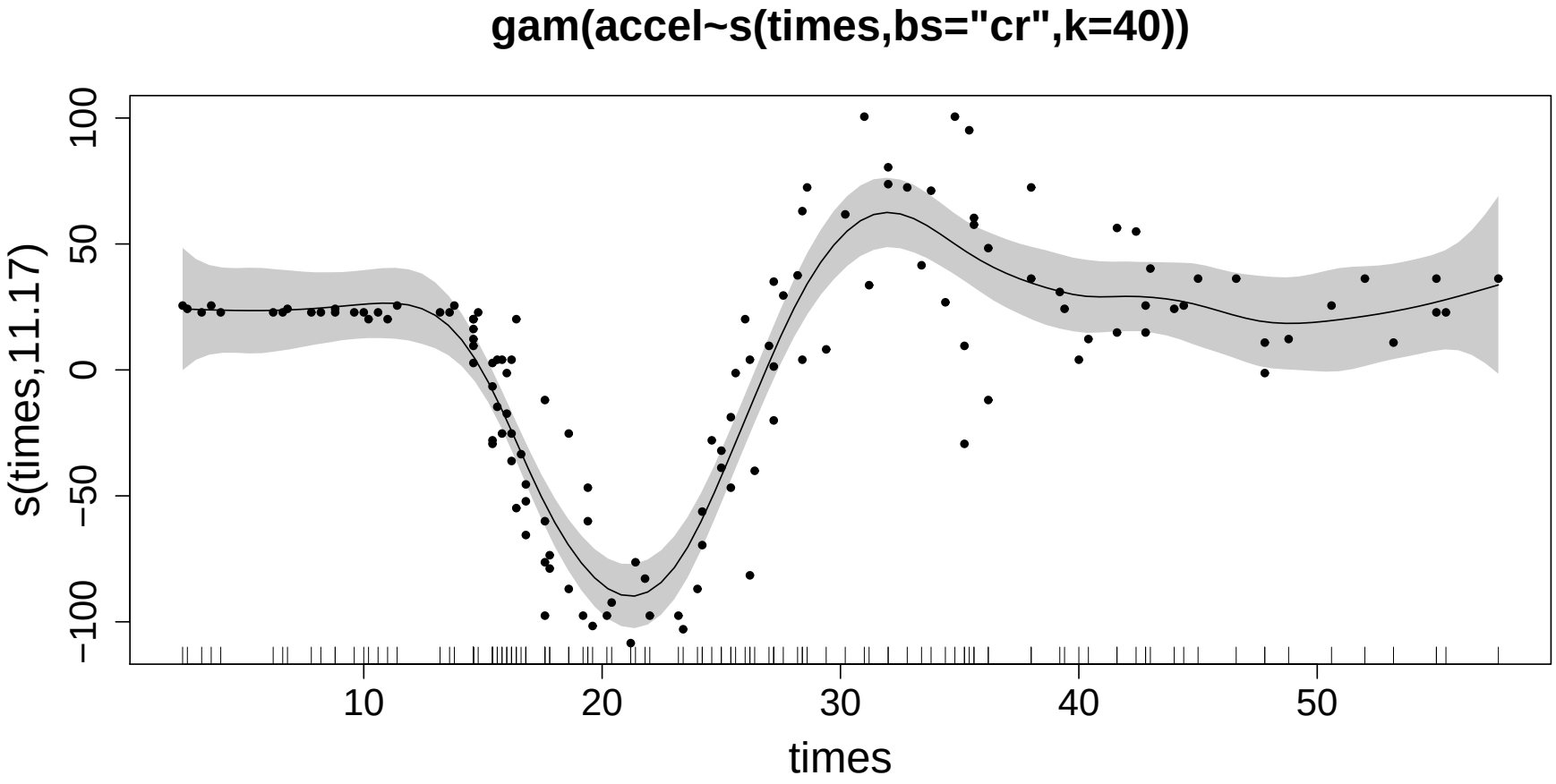
- Consider `gam(y ~ s(z,bs="cr"))`.
- `s(z,bs="cr")` in the model formula generates a *smooth specification object*, `ss`, say, of class `cr.smooth.spec`.
- `gam` needs no details about `ss`, it simply passes it to function `smooth.construct`.
- `smooth.construct` looks at the class of `ss` and passes it to `smooth.construct.cr.smooth.spec`.
- `smooth.construct.cr.smooth.spec` does basis set up, returning an object, `sm` say, of class `cr.smooth`.
- `sm` contains the matrices `X` and `S` that `gam` needs for fitting + basis specific details needed for prediction.

Prediction example

- Prediction is equally modular.
- `predict.gam` simply calls the `Predict.matrix` method function with each smooth in turn as argument.
- E.g. `predict.matrix(sm,NewData)` would cause `Predict.matrix.cr.smooth` to be called based on the class of `sm`.
- `Predict.matrix.cr.smooth` returns a *prediction matrix* of basis functions evaluated at the prediction data values. It takes care of all the basis specific detail, so that `predict.gam` is completely generic.
- So, adding a new type of smoother requires no more than the writing of new `smooth.construct` and `Predict.matrix` method functions.

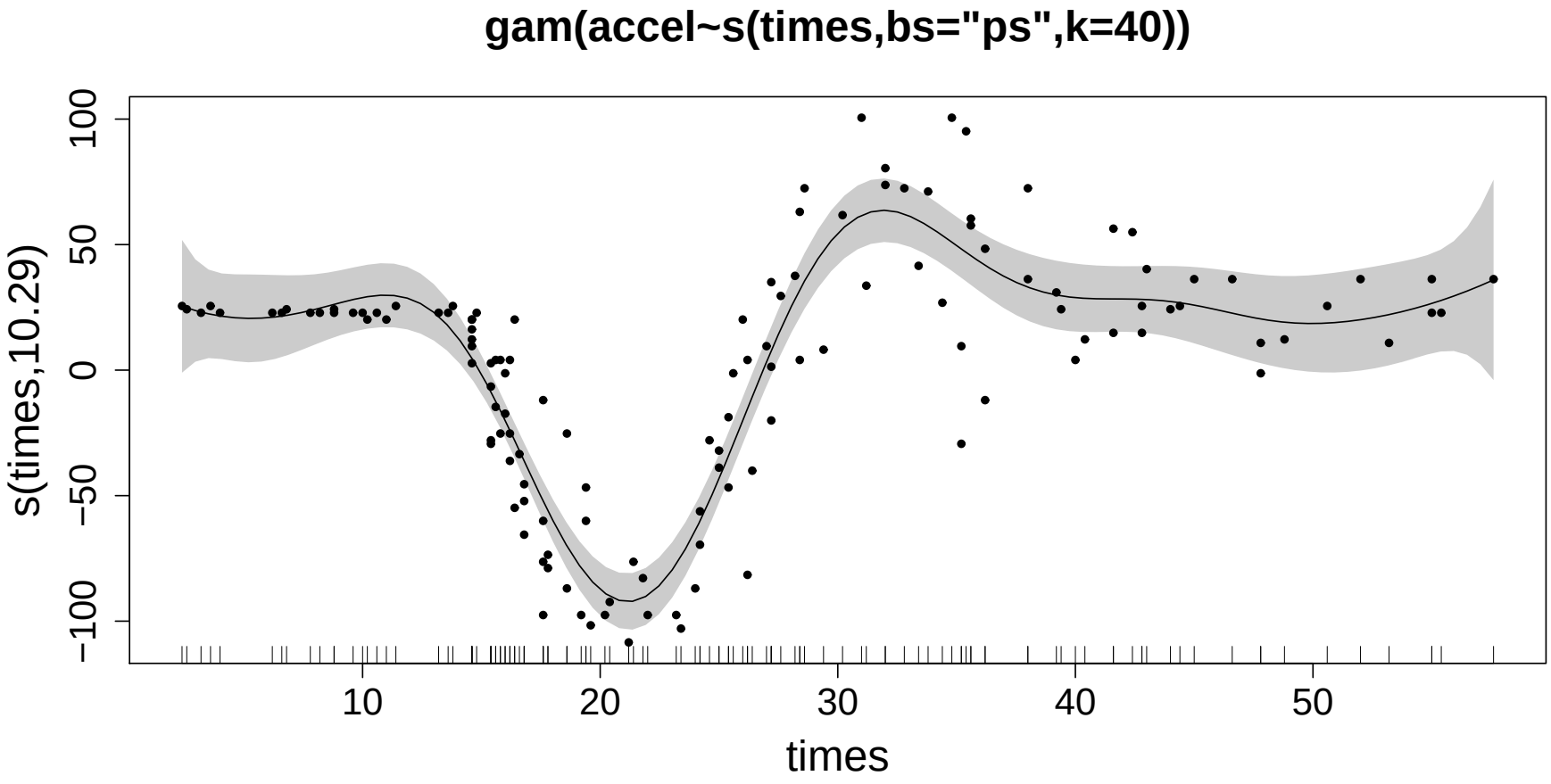
Adding smoother example

- Perhaps I don't like this cubic spline smooth of the `mcycle` data ...



Adding smoothers example

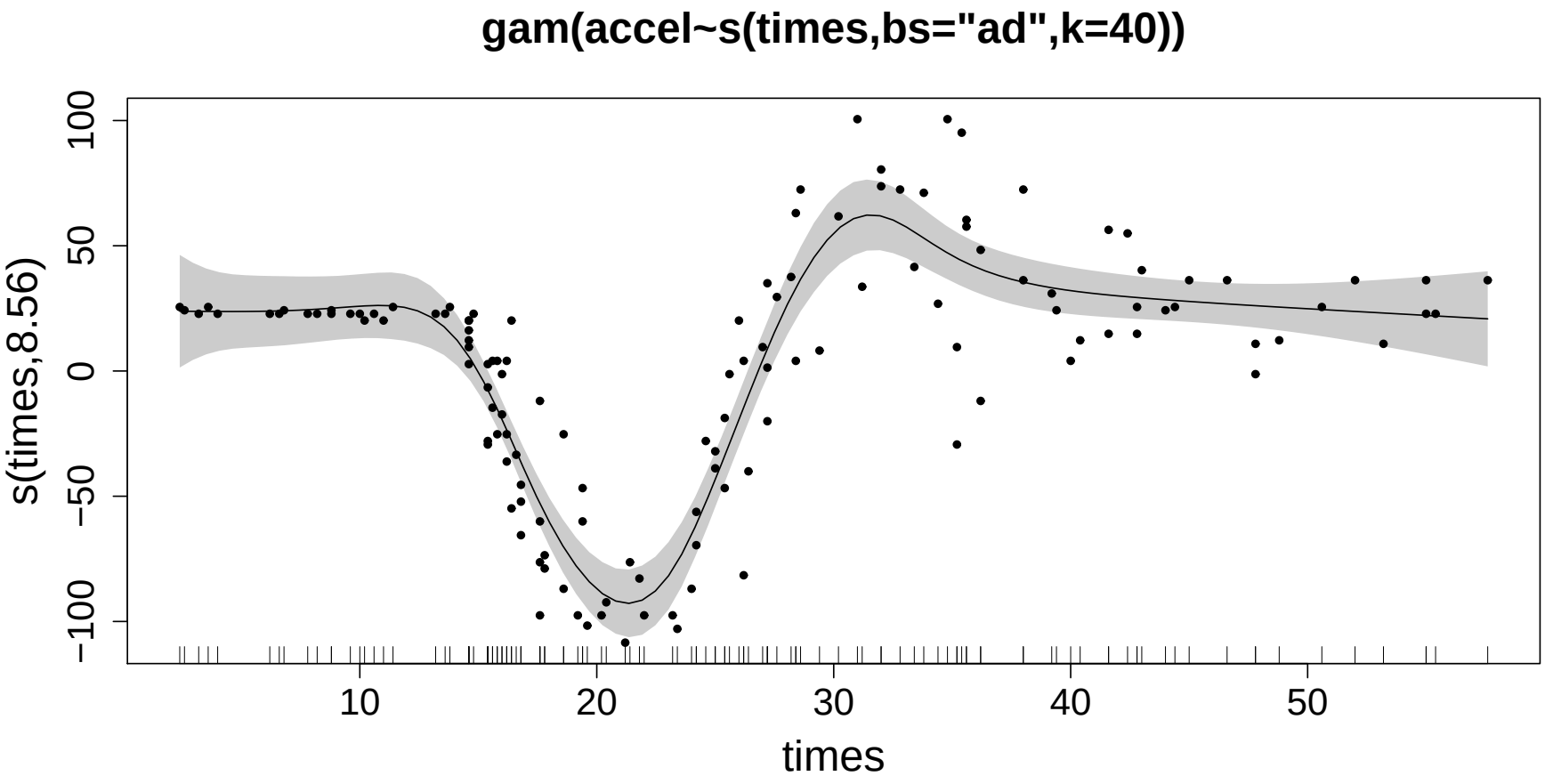
- P-splines are a popular alternative, maybe they will improve matters. . . it's easy to add a `"ps"` class.



- It's worse, but should I really use adaptive smoothing?

Adding smoothers

- It's straightforward to allow the strength of the P-spline penalty to vary smoothly and freely with the smoothing covariate, controlled by multiple smoothing parameters.
- So an adaptive smoothing class can be written, and does give some improvement . . .



Building smooths from smooths

- Tensor product smoothing builds smooths of several variables from *marginal* smooths of single variables.
- Given any marginal bases and penalties, the construction is completely automatic. All that is needed from the marginal smooths is the ability to evaluate model matrices, penalty coefficient matrices and prediction matrices.
- So it is easy to write a smooth class which produces a new tensor product smooth class, by combining existing smooth classes.
- `te` terms in an `mgcv::gam` formulae does just that. As far as `gam` is concerned the `te.smooth.spec` object produced is just like any other smooth specification object, as is the tensor smooth object itself.

Extending the model

- Since smooths can be treated as random effects, some random effects can be treated as smooths...
- Suppose the GAM linear predictor should also include a random effect term $\mathbf{Zb}$ where $\mathbf{b} \sim N(\mathbf{0}, \mathbf{I}\sigma_b^2)$.
- This random effect is equivalent to a smooth with penalty matrix $\mathbf{I}$, and can be treated as such.
- It is straightforward to write constructor and prediction matrix method functions to implement such a random effect as a smoother.
- Nothing about model estimation need change in order to do this.

Random effect class example

- This code implements a simple i.i.d. random effect class...

```
smooth.construct.re.smooth.spec ← function(object,data,knots) {
   object$form ← as.formula(paste(" ~ ",object$term,"-1"))
   object$X ← model.matrix(object$form,data)
   object$rank ← ncol(object$X)
   object$S[[1]] ← diag(object$rank)
   object$C ← matrix(0,0,object$rank)
   object$levels ← levels(data[[object$term]])
   class(object) ← "ranef"
   object
}

Predict.matrix.ranef ← function(object,data) {
   model.matrix(object$form,data,xlev=object$levels)
}
```

- Something like `gam(y~s(g,bs="re"))` invokes it.

Simple random effect continued

- Consider `Rail` data from `nlme` package, on `travel` time of ≤ 3 replicate sound pulses along 6 `Rail`s. (Dropped some reps to unbalance).
- Obvious model has an independent random effect for each rail, with variance σ_b^2.
- `lme(travel~1,data=Rail[-ind,],~1|Rail)` fits the model (`ind` is `(4,5,17)`).
  - $\hat{\sigma}_b = 24.5$ and $\hat{\sigma} = 3.73$ (REML).
- The same model can be fit by `gam(travel~s(Rail,bs="re"),data=Rail[-ind,])`
  - $\hat{\sigma}_b = 27.7$ and $\hat{\sigma} = 3.73$ (GCV).
- Interesting models can be estimated in the same way!

Other model extensions

- The modularity allowed by the R/S class mechanism allows generic extension of the GAM class of models, without any need to modify our toolbox of smooths.
- For example, *varying coefficient* models are parametric GLMs where some coefficients vary smoothly with covariates such as space or time. e.g. something like

$$g\{\mathbb{E}(y_i)\} = \cdots f(x_i)z_i \cdots$$

  where f is smooth and z_i and x_i are covariates.
- So, the i^{th} row of f's model matrix get's multiplied by z_i but nothing else changes from a 'normal' smooth term.
- In `mgcv`, `s(x,by=z)` implements this extension in a smooth class independent way.

More general extensions

- Consider the functional GLM

$$g\{\mathbb{E}(y_i)\} = \int f(t)x_i(t)dt$$

  where predictor x_i is a *known function* and $f(t)$ is an unknown smooth *regression coefficient function*.
- Typically f and x_i are discretized so that $g\{\mathbb{E}(y_i)\} = \mathbf{f}^{\mathrm{T}}\mathbf{x}_i$ where $\mathbf{f}^{\mathrm{T}} = [f(t_1), f(t_2)\ldots]$ and $\mathbf{x}_i^{\mathrm{T}} = [x_i(t_1), x_i(t_2)\ldots]$.
- Generically this is an example of dependence on a linear functional of a smooth. i.e

$$g\{\mathbb{E}(y_i)\} = \cdots L_i f \cdots$$

- Again this would be discretized as $g\{\mathbb{E}(y_i)\} = \cdots \mathbf{L_i f} \cdots$

Linear functional terms

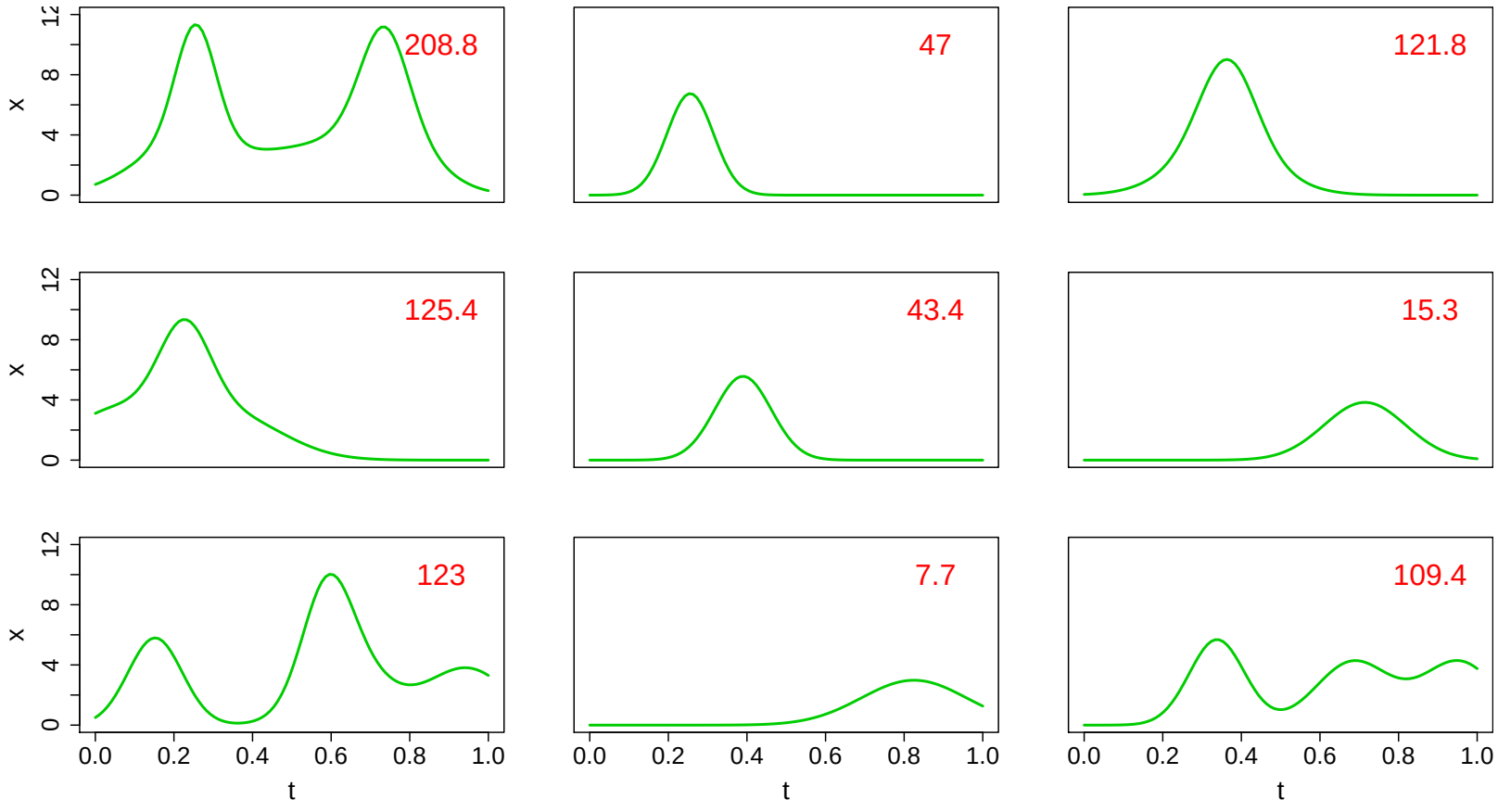
- Such linear functional terms require only that the smoothing basis can be evaluated at some particular values of the smooth argument, so no modification of smooth classes is needed for this.
- `mgcv` handles such terms by allowing `s` to accept matrix arguments and matrix `by` variables.
- So if `X` and `L` are matrices, then `s(X,by=L)` generates

$$g\{\mathbb{E}(y_i)\} = \cdots \sum_j f(X_{ij})L_{ij} \cdots$$

allowing dependence on any linear functional of any existing smooth.

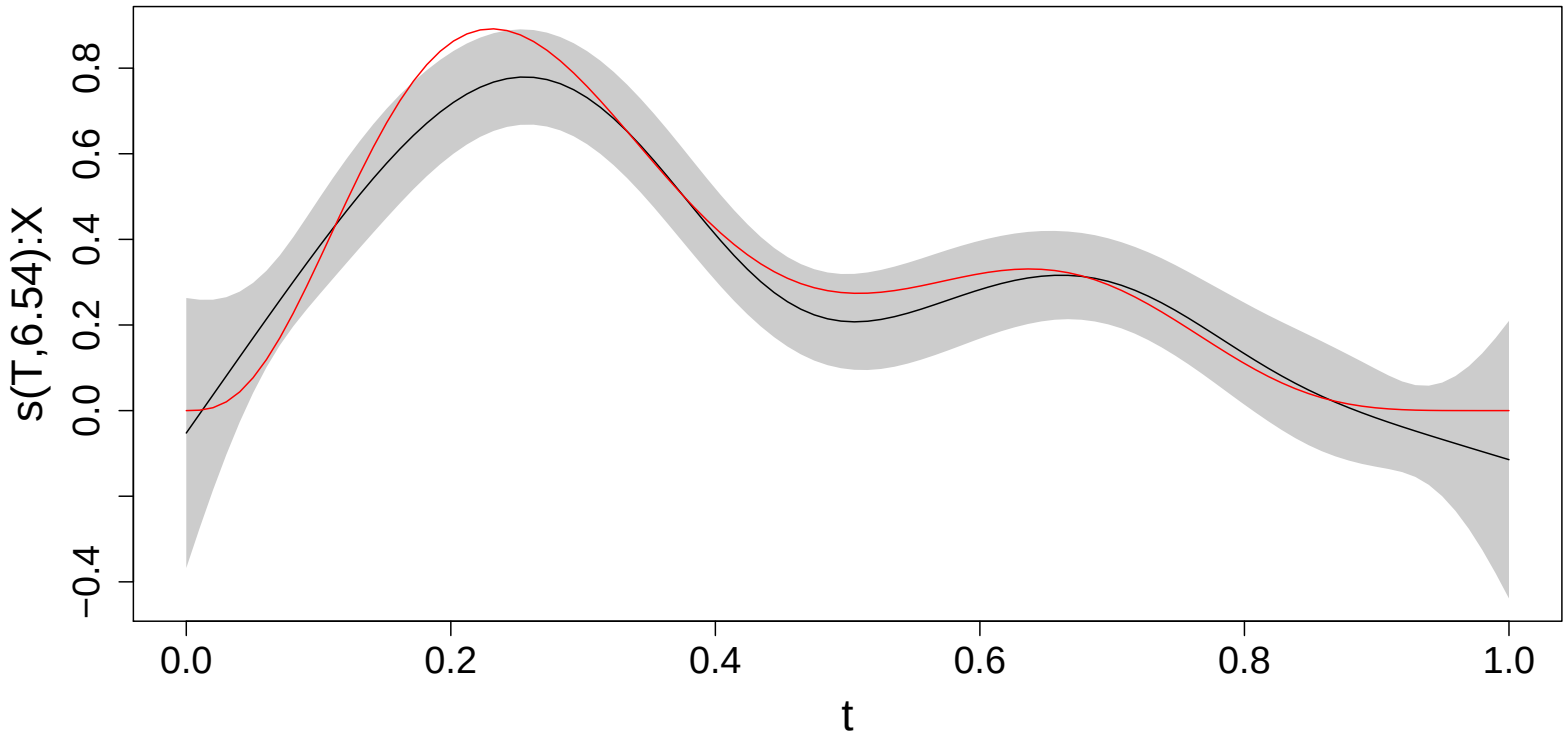
FGLM example

- 150 functions, $x_i(t)$, (each observed at $\mathbf{t}^{\mathrm{T}} = (t_1, \ldots, t_{200})$), with corresponding noisy univariate response, y_i.
- First 9 $(x_i(t), y_i)$ pairs are ...



FGLM fitting

- Want to estimate smooth function f in model $y_i = \int f(t)x_i(t)dt + \epsilon_i$.
- `gam(y~s(T,by=X))` will do this, if `T` and `X` are matrices.
- i^{th} row of `X` is the observed (discretized) function $x_i(t)$. Each row of `T` is a replicate of the observation time vector **t**.



Conclusions

- In the `mgcv` package, the object orientation built into R allows the range of smooths, *and* the way that they are employed as model components, to be extended almost entirely independently of each other.
- So new types of smooth are easy to put into practical use, while any smooth implemented can be used very flexibly in model components.
- The modular setup also enhances software reliability.
- In future it should be possible to re-use the smoother class methods as building blocks for models well outside the GLM class, where a more general penalized likelihood approach would be appropriate.