Generalized additive models and their extensions: the penalized regression spline approach.

Simon N Wood, University of Bath, UK

1 Summary

This course provides an overview of the theory of generalized additive models represented by reduced rank penalized splines, and their practical use with the mgcv package in R. Here generalized additive models include generalized additive mixed models, varying coefficient/geographic regression models, structured additive regression models, generalized linear additive smooth structure models, signal regression models etc, since all of these fit into the same inferential and computational framework (quadratically penalized GLMs). The course will give a compact overview of the essential theory of penalized regression splines and GAMs, focussing on the key theoretical concepts that underpin the more detailed literature: bases, penalties, the Bayesian model of smoothing, and smoothing parameter selection. It will then cover the various types of smooth (one dimensional, isotropic and tensor product interactions) that form the basic toolkit for model construction. Model checking, building and selection will be discussed, including practical exercises with the mgcv package in R. The course will finish with a look at some more advanced GAM topics: spatial and temporal autocorrelation, functional data analysis, and inference via posterior simulation. Participants should preferably bring a laptop, with the latest version of R installed.

Reading: Wood SN, (2006) Generalized Additive Models: An introduction with R

2 Before the course

1. The course assumes that you are familiar with the theory and use of Generalized Linear Models, up to about the level of Chapter 2 of Wood (2006).
2. General familiarity with R and its help system (including browsing html help) is assumed.
3. It is also assumed that you are familiar with the use of `glm` in R for the fitting of GLMs, and have used R's `predict`, `summary`, `anova`, `residuals`, `plot` and `AIC` commands to examine fitted GLMs.
4. To complete the R practicals in the course you will need to bring a laptop with the latest version of R and `mgcv` installed. Installing package `gamair` and `gamm4` might also be a good idea. There will be about 2 hours of lab exercises, so a reasonable capacity laptop battery, fully charged, would be a good idea.

3 Course exercises

Data for the later exercises will be distributed on usb sticks. A selection of further, more advanced exercises will also be available if required.

The `mgcv` library in R provides several functions for estimating generalized additive (mixed) models

1. `gam` is the default version. It's use is much like `glm`, except that you can include smooth functions of covariates in the rhs of the model formula, specifying the linear predictor.
2. `bam` is a version of `gam` designed for large datasets. Its use is just like `gam`. `bam` uses an algorithm that can be faster, but is less stable, than `gam`. It's memory footprint is much less than `gam` by not forming the model matrix whole. It can use multiple cpu cores.
3. `gamm` fits generalized additive mixed models using PQL with `nlme:lme` as the underlying fitting engine. This allows the whole rich random effects and correlation structure available with `lme` to be used with GAMs, but has much the least stable fitting algorithm. Use of `gamm` is somewhat like `gam` and somewhat like `lme`. (Package `gamm4` provides the equivalent for `lme4`).

mgcv also provides numerous functions for plotting, checking, summarizing and predicting with GAMs.

1. Getting started.
   (a) Start R and type `library(mgcv)`, to load the package.
   (b) Type `help.start()` to launch HTML help. Navigate to the `mgcv` help pages in the browser that launches.
   (c) Look at the `gam` help file, and scroll down to the first example model fit. One line at a time, try running the example code up to `gam.check(b)`.
   (d) Try re-fitting the model with `gam` option `method="REML"`, and see if the effect estimates change much.
   (e) Try replacing the additive structure `s(x0)+s(x)` with a smooth interaction term. There are various options: `s(x0,x1)`, `te(x0,x1)` and `t2(x0,x1)`, try a couple and examine the fit using `plot`.
   (f) Now navigate to the help file `mgcv-package` and read it, to get an overview of what the package does.
2. Simple 1D smoothing.
   (a) Type `library(MASS)`, to load the motor-cycle crash data, `mcycle`.
   (b) Plot acceleration against time, to remind yourself what the data look like.
   (c) Use GAM to fit a simple smooth model to the data (acceleration is the response).
   (d) Plot the fitted gam object, using options `residuals=TRUE,pch=19,cex=.3`.
   (e) You can control the smooth in several ways. The `k` argument to `s` controls the basis dimension. Try refitting the model with increased k using something like `s(...,k=20)`.
   (f) You can also change the type of smoothing basis. Try `s(...,bs="ad")` for an adaptive smooth, the penalization of which changes along the length of the smooth. `s(...,bs="ps",m=c(0,2),k=40)` is the piecewise linear smoother used earlier.
3. Data frame `ozone` contains daily(ish) ozone measurements over Los Angeles (`O3`, ppm), along with

   `vh` The height at which the atmospheric pressure is 500mb in metres.

   `wind` the wind speed (reported as miles per hour, but this seems improbable).

   `humidity` (usual % scale).

   `temp` air temperature (Fahrenheit).

   `ibh` the inversion layer base height in feet.

   `ibt` the inversion base temperature (Fahrenheit).

   `dpg` 'Dagget air pressure gradient' (mmhg).

   `vis` visibility in miles.

   `doy` Julian day, where 1 is Jan 1 1976.

   The aim is to build a model to explore the relationship between ozone and the other variables.

   (a) Use something like `ozone <- read.table("ozone.txt")` to read the data into R. Use `pairs(ozone)` to look at it.
   (b) Try a model in which $\texttt{O3}_i \sim$ Gaussian, and $\log(E(\texttt{O3}_i))$ is given by a sum of smooth functions of each of the predictors.
   (c) Using the functions `residuals` and `fitted` to extract from the fitted model, plot residuals against fitted values. Modify the model accordingly.

(d) Check whether an additive (i.e. identity link) structure or a multiplicative (log link) structure might be better. AIC, GCV etc. can be useful here.

(e) Try simplifying the model (the `summary` function may help).

(f) Interpret the smooth plots for your final model: do they make sense?

Extra part (only if time): The smooth function of `doy` should arguably be cyclic (i.e. values and derivatives should match at 'year ends'). Using the `bs="cc"` option to `s` can achieve this, but to get wrapping at the year end (rather than the data ends) you need to supply knot locations for `doy`, which span a full year. To do this you can use the `knots` argument to `gam`: something like `knots=list(doy=c(25,390))` should do it (the strange range is because the data span less than a year, but are spread over 2 calender years).

4. (a) Use `setwd` to set the R working directory to wherever 'ragweed.rda' is located.

   (b) `load("ragweed.rda")` will load a dataset on ragweed pollen count into R. The aim is to predict pollen count using the predictor variables in the dataframe.

   (c) Fit an additive smooth model to the pollen counts (ragweed), which depends smoothly on day of the season, temperature and wind speed. Also allow a dependence on rain. Use REML for smoothness selection. Probably Tweedie or quasipoisson families are best.

   (d) Check your model using gam.check, and any other appropriate residual plots, adjusting if necessary. Examine the smooth effects using e.g. plot(model,pages=1)

   (e) Investigate whether a smooth interaction of temperature and windspeed gives a better model.

   (f) Use your best model to produce a plot of expected pollen count against day of season, for 1994, under conditions of no rain, windspeed 5 and temperature 75(F). Include 95% CIs on the plot. `predict.gam` is the function to use to get the plot data.

4 Some very sketchy bibliographic notes

This is really just some useful pointers, rather than anything complete (it's also clearly massively unbalanced towards my papers). The earliest use of something like the piecewise linear smoother seems to be Whittaker (1923). Hastie and Tibshirani (1990) are responsible for inventing the GAM framework and the software interface. Wahba 's work on splines and smoothing parameter estimation heavily influenced the development of the framework presented here (see e.g. Wahba, 1990 and references therein). Gu and Wahba (1991) produced the first multiple smoothing parameter estimation method, with Wood (2000) producing an equivalent for penalized regression splines, eventually refining the methods until the rather stable method of Wood (2011). Reiss and Ogden (2009) provide a particularly interesting comparison of likelihood and prediction error approaches to smoothness selection. The idea of penalized regression splines goes back to Wahba (1980), but was given renewed impetus by Eilers and Marx (1996, 1998) invention of P-splines, the sparseness of which greatly facilitated the development of Bayesian simulation approaches to GAMs exemplified by Fahrmeir et al. (2004). The link between smoothing and mixed models goes back to Kimeldorf and Wahba (1970), with the Bayesian presentation used here being essentially that of Silverman (1985). Ruppert, Wand and Carrol's (2003) book brought the idea of estimating smooth models as mixed models to wide attention. Duchon (1977) is the rather intense origin of thin plate splines and more, while Wood (2003) discusses their low rank eigen-approximation. Tensor product smoothing started in the smoothing spline literature. See Wood (2006) and Wood, Scheipl and Faraway (2012) for general treatments of the penalized regression spline equivalents (and references to other work on these). Finite area smoothing is discussed in Wood, Bravington and Hedley (2008). Bayesian Confidence intervals for model terms were first proposed in Wahba (1983). Nychka (1988) and Marra and Wood (2011) explain why they have good frequentist properties. Tests useful for smooth terms are compared in Scheipl (2008), while Wood (2013) derives usable p-values for smooth terms based on inversion of the Bayesian confidence intervals. For functional data analysis see in particular Marx and Eilers (1999) and Ramsay and Silverman (2005).

Duchon, J. (1977) Splines minimizing rotation-invariant semi-norms in Solobev spaces *in Construction Theory of Functions of Several Variables* Springer, Berlin.

Eilers, P.H.C. and B.D. Marx (1996) Flexible Smoothing with B-splines and Penalties. *Statistical Science*, 11(2), 89-121.

Fahrmeir, L., T. Kneib & S. Lang (2004) Penalized structured additive regression for space time data: A Bayesian perspective. *Statistica Sinica* 14, 731-761.

Gu, C and Wahba, G. (1991) Minimizing GCV/GML scores with multiple smoothing parameters via the Newton method. *SIAM J. Sci. Stat. Comp.*, 12(2): 383-398

Hastie, T. & R. Tibshirani (1990) *Generalized additive models*. Chapman & Hall, London.

Kimeldorf, G and G. Wahba (1970) A correspondence between Bayesian estimation of stochastic processes and smoothing by splines. *Annals of Mathematical Statistics* 41: 495-502.

Marra G, Wood SN (2012) Coverage Properties of Confidence Intervals for Generalized Additive Model Components Scandinavian Journal of Statistics 39, 53-74

Marx B. D. & P.H. Eilers (1999) Generalized Linear Regression on Sampled Signals and Curves: A P-Spline Approach *Technometrics* 41(1), 1-13.

Marx B. D. & P.H. Eilers (1998) Direct generalized additive modeling with penalized likelihood. *Computational Statistics and Data Analysis* 28, 193-209.

Nychka, D. (1988), Bayesian Confidence Intervals for Smoothing Splines, Journal of the American Statistical Association, 83, 1134–1143.

Ramsay, J.O. & B.W. Silverman (2005) *Functional Data Analysis*. Springer.

Reiss, P.T. & R.T. Ogden (2009) Smoothing parameter selection for a class of semiparametric linear models. *Journal of the Royal Statistical Society, Series B* 71, 505-524.

Ruppert, D., M.P. Wand & R.J. Carroll (2003) *Semiparametric Regression*. Cambridge.

Scheipl, F. and Greven, S. and Küchenhoff, H. (2008), Size and power of tests for a zero random effect variance or polynomial regression in additive and linear mixed models, Comp. Statist. Data Anal., 52,3283-3299.

Silverman, B.W. (1985) Some aspects of the spline smoothing approach to non-parametric regression curve fitting (with discussion). *Journal of the Royal Statistical Society, Series B* 47, 1-53.

Wahba, G (1990) *Spline models for observational data*. SIAM, Philadelphia.

Wahba, G. (1983) Bayesian confidence intervals for the cross validated smoothing spline. *Journal of the Royal Statistical Society, Series B* 45, 133-150.

Wahba, G. (1980) Spline bases, regularization and generalized cross validation for solving approximation problems with large quantities of noisy data. in E. Cheney (ed) *Approximation Theory III* Academic Press, London.

Whittaker, E.T. (1923)On a new method of graduation Proc. Edin. Math Soc. 41, 63-75.

Wood, SN, 2013, On p-values for smooth components of an extended generalized additive model. Biometrika.

— 2011, Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models. JRSSB, 73(1), 3-36

—, F. Scheipl, and JJ Faraway (2012) Straightforward intermediate rank tensor product smoothing in mixed models. Statistical Computing in press.

—, 2008, Fast stable direct fitting and smoothness selection for Generalized Additive Models. JRSSB 70(3), 495-518.

—, MV Bravington and SL Hedley, 2008, Soap film smoothing. JRSSB, 70(5), 931-955.

—, 2006, Low-Rank Scale-Invariant Tensor Product Smooths for Generalized Additive Mixed Models. Biometrics 62(4), 1025-1036.

—, 2004, Stable and Efficient Multiple Smoothing Parameter Estimation for Generalized Additive Models. JASA 99, 673-686

—, 2003, Thin plate regression splines. JRSSB, 65(1), 95-114

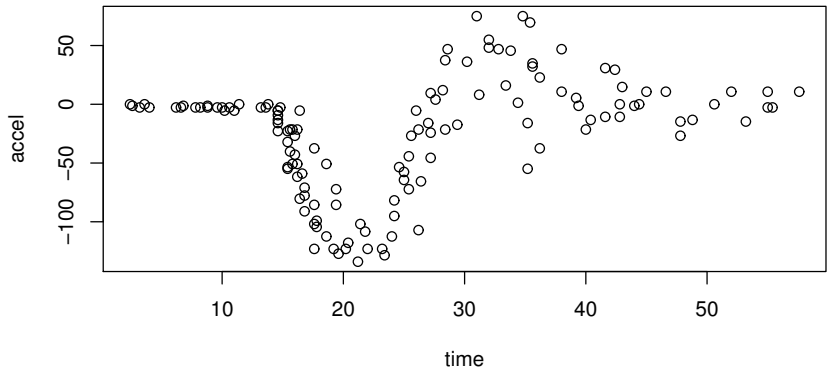
—, 2000, Modelling and smoothing parameter estimation with multiple quadratic penalties. JRSSB 62(2), 413-428

Basis Penalty Smoothers

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

Estimating functions

- Here are some ancient data...



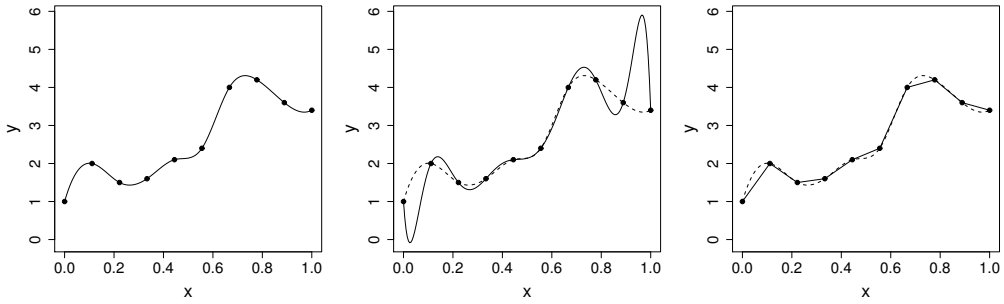
- If f is 'a smooth function', a suitable model might be

$$\texttt{accel}_i = f(\texttt{time}_i) + \epsilon_i.$$

- How to represent f? What function space should we search?
- A space that is good for approximating known functions would be a sensible starting point.

A space for f

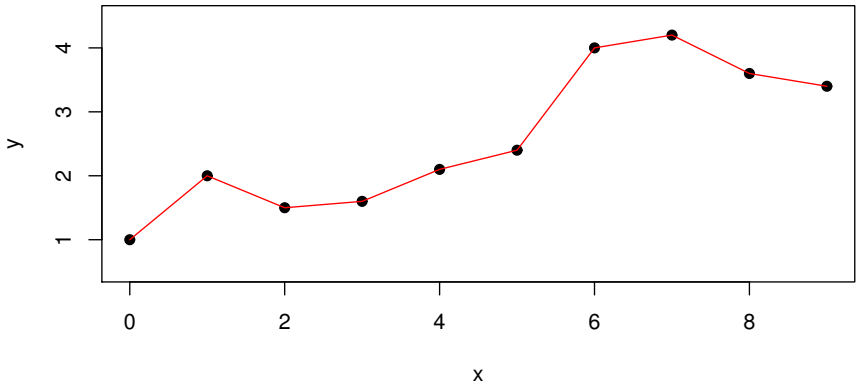
- Taylor's theorem might suggest using the space of polynomials, but look at the middle panel's attempt to approximate the function on the left with a polynomial.



- Trying to pass through the black dots and maintain continuity of all derivatives requires wild oscillation.
- Reducing the continuity requirements gives the better behaved piecewise linear interpolant on the right.

A simple basis for f

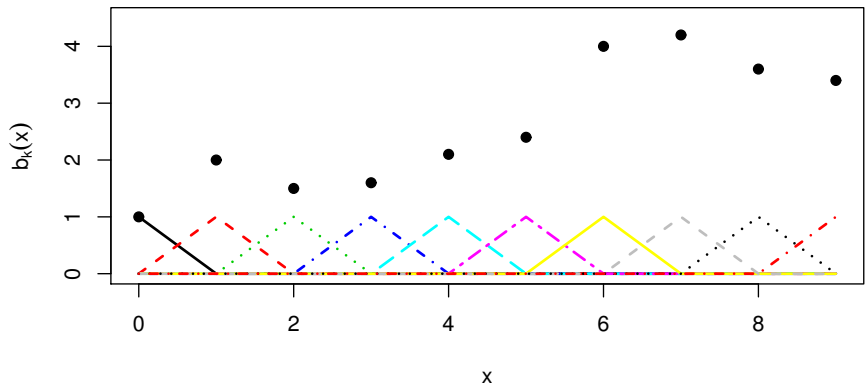
- So, for now, let's represent f as a piecewise linear function, with derivative discontinuities at x_k^*.



- ...this can be written $f(x) = \sum_k \beta_k b_k(x)$, where the b_k are *tent functions*: there is one per •. The coefficients β_k give $f(x_k^*)$ directly.

The tent basis

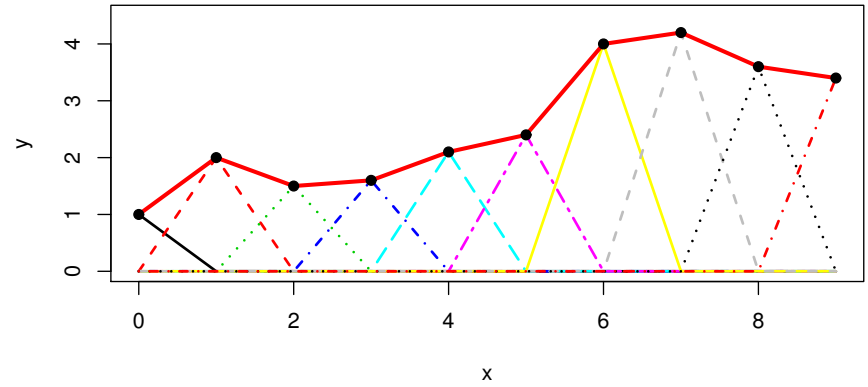
- The k^{th} tent function is 1 at x_k^* and descends linearly to zero at $x_{k\pm1}^*$. Elsewhere it is zero.
- The full set look like this...



- Under this definition of $b_k(x)$, we would interpolate x_k^*, y_k^* data by just setting $\beta_k = y_k^*$.

How the tent basis works

- So the function is represented by multiplying each tent function by its coefficient, β_k, and summing the results...



- Given the basis functions and coefficients, we can *predict* the value of f anywhere in the range of the x^* values.

Prediction matrix

- f is defined by the x_k^* values defining the tent basis, and coefficients β_k.
- Now suppose that we want to evaluate the interpolant at a series of values x_i.
- If $\mathbf{f} = [f(x_1), f(x_2), \ldots]^T$, then

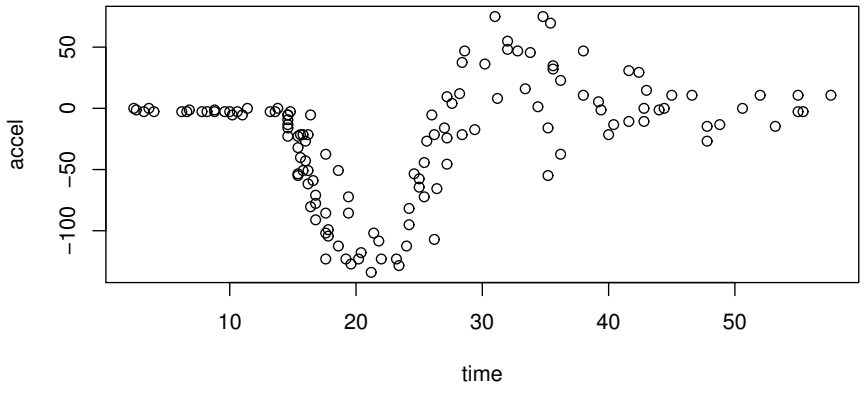
$$\mathbf{f} = \mathbf{X}\boldsymbol{\beta}$$

where the *prediction matrix* is given by

$$\mathbf{X} = \begin{bmatrix} b_1(x_1) & b_2(x_1) & b_3(x_1) & . \\ b_1(x_2) & b_2(x_2) & . & . \\ . & . & . & . \\ . & . & . & . \end{bmatrix}$$

Regression with a basis

- Returning to these data...



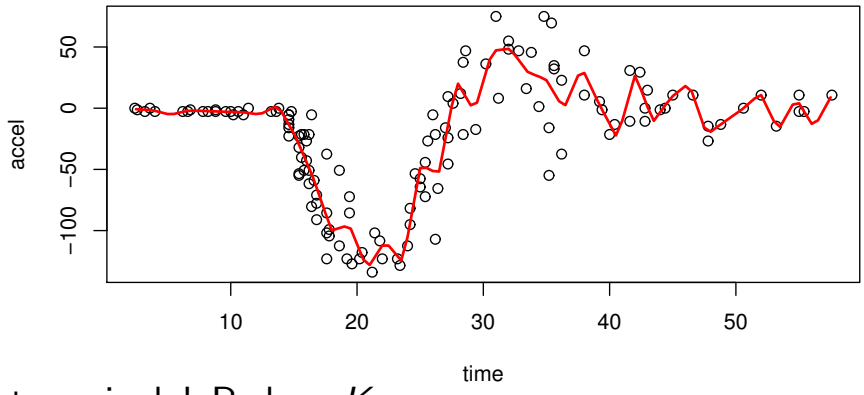
- We can define a tent basis by choosing some t_k^* values spread evenly through the range of observed times.
- Then the model, $a_i = f(t_i) + \epsilon_i$ becomes

$$\mathbf{a} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

...a straightforward linear model.

Estimation in R

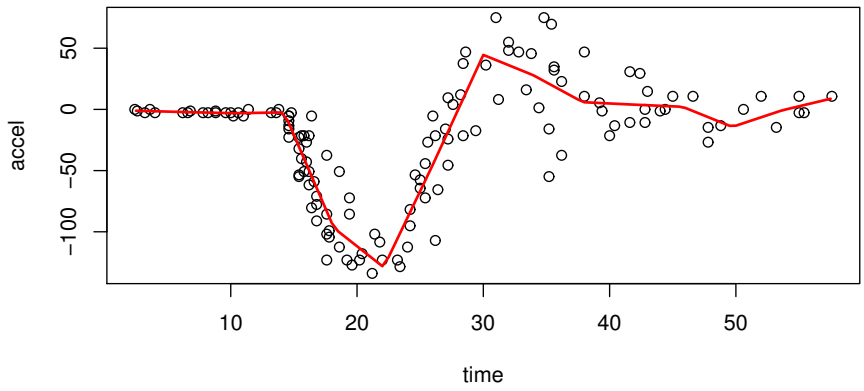
- A few lines of R code are enough to produce $\mathbf{X}$. Then `lm` can be used to fit the model.
- Here is the result using K=40 evenly spaced t_k^* (knots).



- Far too wiggly! Reduce K

Reducing K

- After some experimentation, $K = 15$ seems reasonable...



- ...but K selection is a bit fiddly and ad hoc.
  1. Models with different K are not nested, so we can't use hypothesis testing.
  2. We have little choice but to fit with every possible K value if AIC is to be used.
  3. Very difficult to generalize this model selection approach to models with more than one function.

Smoothing

- Using the basis for *regression* was ok, but there are some problems choosing K and deciding where to put the *knots*, x_k^*.
- To overcome these consider using the basis for *smoothing*.
  1. Make K 'large enough' that bias is negligible.
  2. Use even x_k^* spacing.
  3. To avoid overfit, penalize the wiggliness of f using, e.g.

$$\mathcal{P}(f) = \sum_{k+1}^{K-1} (\beta_{k-1} - 2\beta_k + \beta_{k+1})^2$$

Evaluating the penalty

- To get the penalty in convenient form, note that

$$\begin{bmatrix} \beta_1 - 2\beta_2 + \beta_3 \\ \beta_2 - 2\beta_3 + \beta_4 \\ . \\ . \end{bmatrix} = \begin{bmatrix} 1 & -2 & 1 & 0 & . & . \\ 0 & 1 & -2 & 1 & .. & \\ . & . & . & . & . & . \\ . & . & . & . & . & . \end{bmatrix} \boldsymbol{\beta} = \mathbf{D}\boldsymbol{\beta}$$

  by definition of $\mathbf{D}$
- Hence

$$\mathcal{P}(f) = \boldsymbol{\beta}^{\mathsf{T}}\mathbf{D}^{\mathsf{T}}\mathbf{D}\boldsymbol{\beta} = \boldsymbol{\beta}^{\mathsf{T}}\mathbf{S}\boldsymbol{\beta}$$

  by definition of $\mathbf{S}$.

Penalized fitting

- Now the penalized least squares estimates are

$$\hat{\boldsymbol{\beta}} = \arg\min_{\beta} \sum_i \{a_i - f(t_i)\}^2 + \lambda \mathcal{P}(f)$$

  smoothing parameter λ controls the fit-wiggliness tradeoff.
- For computational purposes this is re-written

$$\hat{\boldsymbol{\beta}} = \arg\min_{\beta} \|\mathbf{a} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \boldsymbol{\beta}^{\mathsf{T}} \mathbf{S} \boldsymbol{\beta}.$$

- Formally,

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^{\mathsf{T}}\mathbf{X} + \lambda \mathbf{S})^{-1} \mathbf{X}^{\mathsf{T}} \mathbf{a}$$

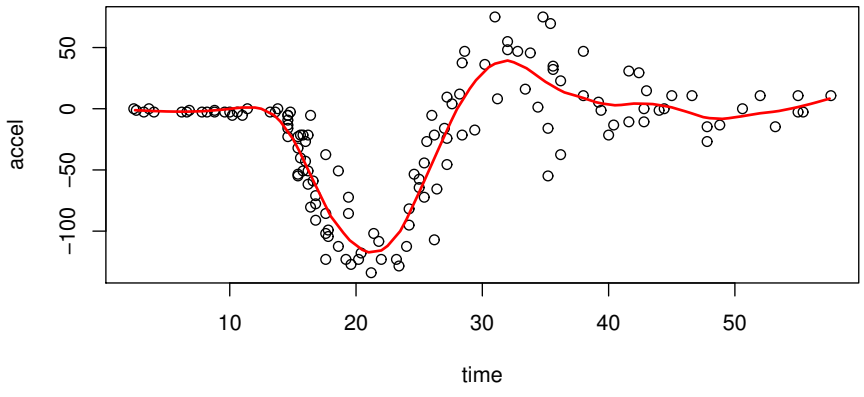
  but direct use of this expression has sub-optimal computational stability.

Computing the smooth fit

- In fact

$$\|\mathbf{a} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \boldsymbol{\beta}^{\mathsf{T}} \mathbf{S} \boldsymbol{\beta} = \left\| \begin{bmatrix} \mathbf{a} \\ \mathbf{0} \end{bmatrix} - \begin{bmatrix} \mathbf{X} \\ \sqrt{\lambda}\mathbf{D} \end{bmatrix} \boldsymbol{\beta} \right\|^2$$

- The rhs is the RSS for an augmented linear model, which can be stably fit using `lm`. Here's an example using $K = 40$, but now penalizing...



Issues raised by smoothing

- Notice the dominant role of the penalty in the smoothed f — the discontinuity of the basis is barely visible, the penalty has so smoothed the results.
- But the dramatic effect of penalization raises questions
  1. How do we measure complexity of the model now that penalization has clearly yielded a result much smoother than K=40 would suggest?
  2. What distributional properties will $\hat{f}$ have under penalized estimation?
  3. How do we go about choosing/estimating the degree of penalization (λ)?

The natural basis

- To get started on these questions note that any basis-penalty smoother can be reparameterized so that its basis matrix is orthognal and its penalty is diagonal.
- Let a smoother have model matrix $\mathbf{X}$ and penalty matrix $\mathbf{S}$.
- Form QR decomposition $\mathbf{X} = \mathbf{QR}$, followed by symmetric eigen-decompostion

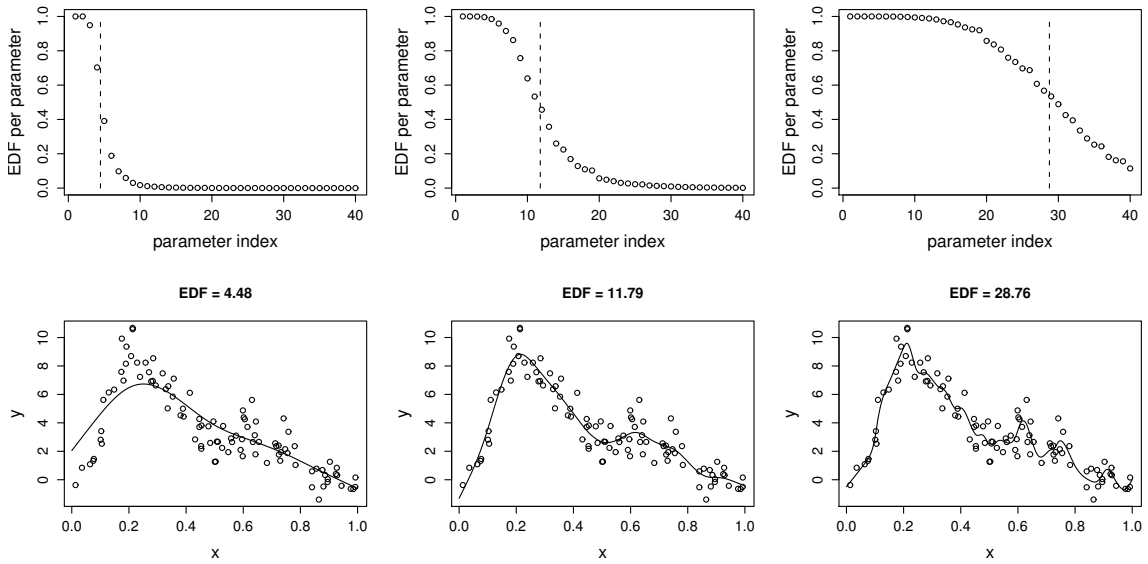
$$\mathbf{R}^{-\mathsf{T}}\mathbf{S}\mathbf{R}^{-1} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathsf{T}}$$

- Define $\mathbf{P} = \mathbf{U}^{\mathsf{T}}\mathbf{R}$. And reparameterize $\boldsymbol{\beta}' = \mathbf{P}\boldsymbol{\beta}$.
- In the new parameterization the model matrix is $\mathbf{X}' = \mathbf{QU}$, which has orthogonal columns. ($\mathbf{X} = \mathbf{X}'\mathbf{P}$.)
- The penalty matrix is now the diagonal matrix $\mathbf{\Lambda}$ (eigenvalues in decreasing order down leading diagonal).

Effective Degrees of Freedom

- Penalization restricts the freedom of the coefficients to vary. So with 40 coefficients we have < 40 *effective degrees of freedom* (EDF).
- How the penalty restricts the coefficients is best seen in the natural parameterization. (Let $\mathbf{y}$ be the response.)
- Without penalization the coefficients would be $\tilde{\boldsymbol{\beta}}' = \mathbf{X}'^T\mathbf{y}$.
- With penalization the coefficients are $\hat{\boldsymbol{\beta}}' = (\mathbf{I} + \lambda\boldsymbol{\Lambda})^{-1}\mathbf{X}'^T\mathbf{y}$.
- i.e. $\hat{\beta}_j = \tilde{\beta}_j(1 + \lambda\Lambda_{jj})^{-1}$.
- So $(1 + \lambda\Lambda_{jj})^{-1}$ is the *shrinkage factor* for the i^{th} coefficient, and is bounded between 0 and 1. It gives the EDF for $\hat{\beta}_j$.
- So total EDF is $\text{tr}\{(1 + \lambda\Lambda_{jj})^{-1}\} = \text{tr}(\mathbf{F})$, where $\mathbf{F} = (\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda\mathbf{S})^{-1}\mathbf{X}^\mathsf{T}\mathbf{X}\}$, the 'EDF matrix'.

EDF Illustrated



Smoothing bias

- The formal expression for the penalized least squares estimates is $\hat{\boldsymbol{\beta}} = (\mathbf{X}^{\mathsf{T}}\mathbf{X} + \lambda\mathbf{S})^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{y}$
- Hence

$$\begin{aligned} E(\hat{\boldsymbol{\beta}}) &= (\mathbf{X}^{\mathsf{T}}\mathbf{X} + \lambda\mathbf{S})^{-1}\mathbf{X}^{\mathsf{T}}E(\mathbf{y}) \\ &= (\mathbf{X}^{\mathsf{T}}\mathbf{X} + \lambda\mathbf{S})^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{X}\boldsymbol{\beta} \\ &= \mathbf{F}\boldsymbol{\beta} \neq \boldsymbol{\beta} \end{aligned}$$

- Smooths are baised!
- i.e. we control model mis-specification bias by using a large K ... but to control the resulting variance we have to penalize ... which leads to smoothing bias.
- The bias makes frequentist inference difficult (including bootstrapping!).

A Bayesian smoothing model

- We penalize because we think that the truth is more likely to be smooth than wiggly.
- Things can be formalized by putting a prior on wiggliness

$$\text{wiggliness prior} \propto \exp(-\lambda\boldsymbol{\beta}^{\mathsf{T}}\mathbf{S}\boldsymbol{\beta}/(2\sigma^2))$$

- ... equivalent to a prior $\boldsymbol{\beta} \sim N(\mathbf{0}, \mathbf{S}^{-}\sigma^2/\lambda)$ where $\mathbf{S}^{-}$ is a generalized inverse of $\mathbf{S}$.
- From the model $\mathbf{y}|\boldsymbol{\beta} \sim N(\mathbf{X}\boldsymbol{\beta}, \mathbf{I}\sigma^2)$, so from Bayes' Rule

$$\boldsymbol{\beta}|\mathbf{y} \sim N(\hat{\boldsymbol{\beta}}, (\mathbf{X}^{\mathsf{T}}\mathbf{X} + \lambda\mathbf{S})^{-1}\sigma^2)$$

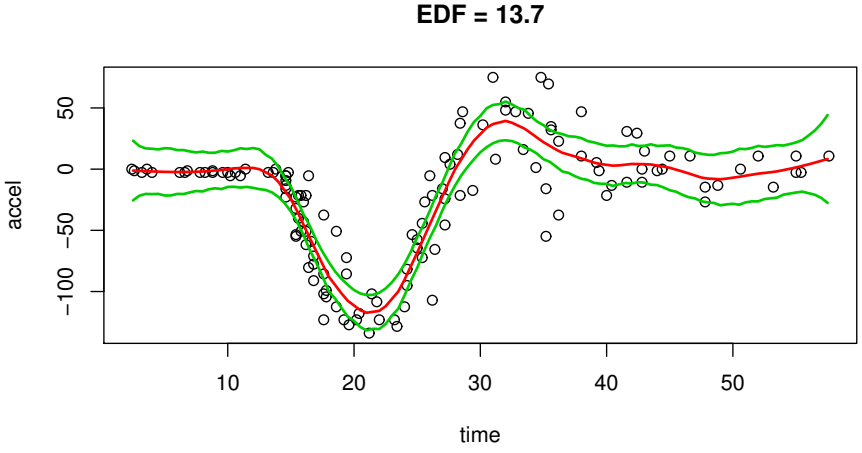
- Finally $\hat{\sigma}^2 = \|\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}\|^2/\{n - \mathrm{tr}(\mathbf{F})\}$ is useful.

Consequences of the Bayesian model

- The Bayesian model has the same structure as a linear mixed model, and can be computed as such.
- $\boldsymbol{\beta} \sim N(\mathbf{0}, \mathbf{S}^{-}\sigma^2/\lambda) \Rightarrow \mathbf{f} \sim N(\mathbf{0}, (\mathbf{X}\mathbf{S}\mathbf{X}^{\mathsf{T}})^{-}\sigma^2/\lambda)$, i.e. f is equivalent to a Gaussian random field with covariance matrix $(\mathbf{X}\mathbf{S}\mathbf{X}^{\mathsf{T}})^{-}\sigma^2/\lambda$.
- But even if we compute f using mixed model technology, we are really being Bayesian in most cases...
- ...usually we do not expect f to be re-drawn from the prior on each replication of the response data, as a true random effect would be.

The Bayesian model in action

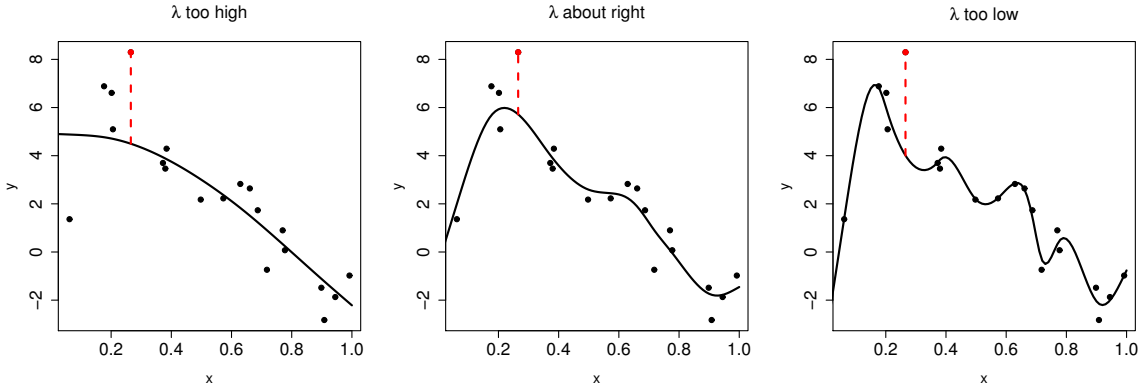
- An argument due to Nychka (1988) shows that the intervals for f based on the Bayesian posterior have good across the function frequentist coverage, because the Bayesian covariance matrix can be viewed as including a squared bias component.
- Here is an example of such an interval



Smoothness selection approaches

- The smoothing model $y_i = f(x_i) + \epsilon_i$, $\epsilon_i \sim N(0, \sigma^2)$, is represented via a basis expansion of f, with coefficients $\boldsymbol{\beta}$.
- The $\boldsymbol{\beta}$ estimates are $\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \boldsymbol{\beta}^{\mathsf{T}} \mathbf{S} \boldsymbol{\beta}$ where $\mathbf{X}$ is the model matrix derived from the basis, and $\mathbf{S}$ is the wiggliness penalty matrix.
- λ controls smoothness — how should it be chosen?
- There are 3 main statistical approaches
  1. Choose λ to minimize error in predicting new data.
  2. Treat smooths as random effects, following the Bayesian smoothing model, and estimate λ as a variance parameter using a marginal likelihood approach.
  3. Go fully Bayesian by completing the Bayesian model with a prior on λ (requires simulation and not pursued here).

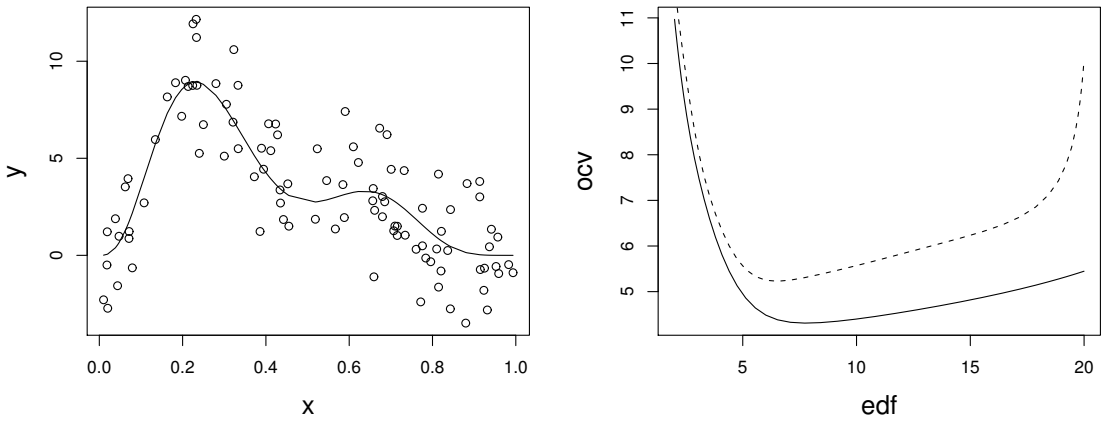
Prediction error: cross validation



1. Choose λ to try to minimize the error predicting new data.
2. Minimize the average error in predicting single datapoints *omitted* from the fit. Each datum left out once in average.
3. If $\mathbf{A} = \mathbf{X}(\mathbf{X}^{\mathsf{T}}\mathbf{X} + \lambda \mathbf{S})^{-1}\mathbf{X}^{\mathsf{T}}$, it turns out that

$$\mathcal{V}_o(\lambda) = \frac{1}{n}\sum_i (y_i - \hat{\mu}_i^{[-i]})^2 = \frac{1}{n}\sum_i \frac{(y_i - \hat{\mu}_i)^2}{(1 - A_{ii})^2}$$

OCV not invariant



- OCV is not invariant in an odd way. If $\mathbf{Q}$ is orthogonal then fitting objective

$$\|\mathbf{Qy} - \mathbf{QX}\boldsymbol{\beta}\|^2 + \lambda\boldsymbol{\beta}^{\mathsf{T}}\mathbf{S}\boldsymbol{\beta}$$

yields identical inferences about $\boldsymbol{\beta}$ as the original objective, but it gives a different $\mathcal{V}_o$.

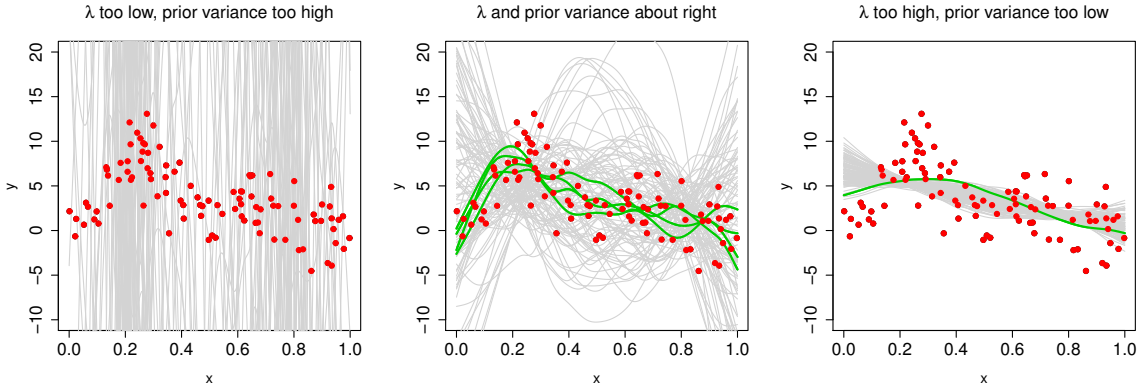
GCV: generalized cross validation

- If we find the $\mathbf{Q}$ that causes the leading diagonal elements of $\mathbf{A}$ to be constant, and then perform OCV, the result is the invariant alternative GCV:

$$\mathcal{V}_g = \frac{n\|\mathbf{y} - \hat{\boldsymbol{\mu}}\|^2}{\{n - \mathrm{tr}(\mathbf{A})\}^2}$$

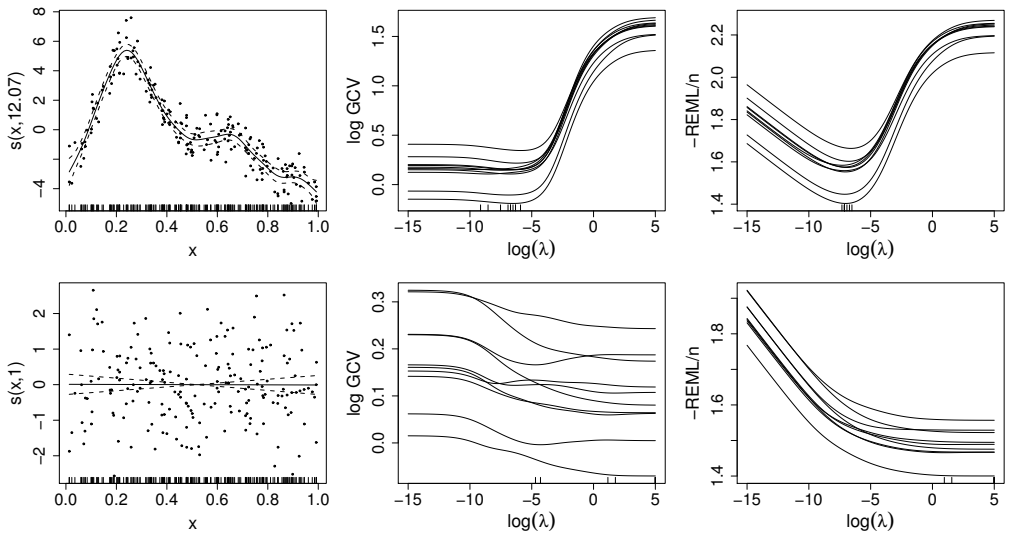
- It is easy to show that $\mathrm{tr}(\mathbf{A}) = \mathrm{tr}(\mathbf{F})$, where $\mathbf{F}$ is the degrees of freedom matrix.
- In addition to invariance, GCV is much easier to optimize efficiently in the multiple smoothing parameter case.

Marginal Likelihood smoothness selection



1. Choose λ to maximize the average likelihood of random draws from the prior implied by λ.
2. If λ too low, then almost all draws are too variable to have high likelihood. If λ too high, then draws all underfit and have low likelihood. The right λ maximizes the proportion of draws close enough to data to give high likelihood.
3. Formally, maximize e.g. $\mathcal{V}_r(\lambda) = \log \int f(\mathbf{y}|\boldsymbol{\beta}) f_\lambda(\boldsymbol{\beta}) d\boldsymbol{\beta}$. - Marginal Likelihood.

Prediction error vs. likelihood $\boldsymbol{\lambda}$ estimation



1. Pictures show GCV and REML scores for different replicates from same truth.
2. Compared to REML, GCV penalizes overfit only weakly, and so is more likely to occasionally undersmooth.

Summary

- We can construct smoothers from sets of basis functions, with associated quadratic penalties.
- Estimation is then by quadratically penalized least squares.
- Penalization reduces freedom to vary: we need a notion of effective degrees of freedom.
- A Bayesian view of smoothing is useful for further inference.
- The appropriate amount of penalization can be estimated by marginal likelihood or prediction error methods.

Generalized Additive Models

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

Introduction

- We have seen how to
  1. turn model $y_i = f(x_i) + \epsilon_i$ into $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$ and a wiggliness penalty $\boldsymbol{\beta}^{\mathsf{T}}\mathbf{S}\boldsymbol{\beta}$.
  2. estimate $\boldsymbol{\beta}$ given $\boldsymbol{\lambda}$ as $\hat{\boldsymbol{\beta}} = \arg\ \min_{\beta} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda\boldsymbol{\beta}^{\mathsf{T}}\mathbf{S}\boldsymbol{\beta}$.
  3. estimate $\boldsymbol{\lambda}$ by GCV, AIC, REML etc.
  4. use $\boldsymbol{\beta}|\mathbf{y} \sim N(\hat{\boldsymbol{\beta}}, (\mathbf{X}^{\mathsf{T}}\mathbf{X} + \lambda\mathbf{S})^{-1}\sigma^2)$ for inference.
- . . . all this can be extended to models with multiple smooth terms, for exponential family response data . . .

Additive Models

- Consider the model

$$y_i = \mathbf{A}_i\boldsymbol{\theta} + \sum_j f_j(x_{ji}) + \epsilon_i, \qquad \epsilon_i \sim N(0, \sigma^2)$$

  - $\mathbf{A}_i$ is the i^{th} row of the model matrix for any parametric terms, with parameter vector $\boldsymbol{\theta}$. Assume it includes an intercept.
  - f_j is a smooth function of covariate x_j, which may be vector valued.
- The f_j are confounded via the intercept, so that the model is only estimable under identifiability constraints on the f_j.
- The best constraints are $\sum_i f_j(x_{ji}) = 0 \;\; \forall\, j$.
- If $\mathbf{f} = [f(x_1), f(x_2), \ldots]$ then the constraint is $\mathbf{1}^{\mathsf{T}}\mathbf{f} = 0$, i.e. $\mathbf{f}$ is orthogonal to the intercept. Other constraints give wider CIs for the constrained f_j.

Representing the model

- Choose a basis and penalty for each f_j.
- Let the model matrix for f_j be $\mathbf{X}$ and let $\lambda\boldsymbol{\beta}^{\mathsf{T}}\mathbf{S}\boldsymbol{\beta}$ be the penalty (more generally $\sum_j \lambda_j\boldsymbol{\beta}^{\mathsf{T}}\mathbf{S}_j\boldsymbol{\beta}$).
- Reparameterize to absorb the constraint $\mathbf{1}^{\mathsf{T}}\mathbf{X} = 0$. The simplest recipe is as follows
  1. Subtract the column mean from each column of $\mathbf{X}$ to give $\mathbf{X}'$.
  2. Drop the column of $\mathbf{X}'$ with lowest variance to give constrained model matrix $\mathbf{X}^{[j]}$, and drop the corresponding row and column of $\mathbf{S}$ to give constrained penalty matrix $\mathbf{S}_j$.
  3. After fitting, when creating a new version of $\mathbf{X}^{[j]}$ for predicting at new covariate values, it's important to subtract the original column means $\mathbf{x}$ from the new matrix's columns, and to drop the same column as before (simply repeating steps 1 and 2 on the new model matrix will lead to an interesting mess).

The estimable AM

- Now $y_i = \mathbf{A}_i\boldsymbol{\theta} + \sum_j f_j(x_{ji}) + \epsilon_i$ becomes $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$ where

$$\mathbf{X} = [\mathbf{A} : \mathbf{X}^{[1]} : \mathbf{X}^{[2]} : \cdots]$$

  and $\boldsymbol{\beta}$ contains $\boldsymbol{\theta}$ followed by the basis coefficients for the f_j.
- After suitable padding of the $\mathbf{S}_j$ with zeroes the penalty becomes $\sum_j \lambda_j \boldsymbol{\beta}^{\mathsf{T}} \mathbf{S}_j \boldsymbol{\beta}$.
- Now $\hat{\boldsymbol{\beta}} = \arg\ \min_\beta \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \sum_j \lambda_j \boldsymbol{\beta}^{\mathsf{T}} \mathbf{S}_j \boldsymbol{\beta}$.
- Again $\boldsymbol{\lambda}$ can be estimated by GCV, REML etc.

Linear functional generalization

- Occasionally we may want a model that depends on an f_j in some way other than simple evaluation. So let L_{ij} be a linear operator and consider an extended model

$$y_i = \mathbf{A}_i\boldsymbol{\theta} + \sum_j L_{ij} f_j(x_j) + \epsilon_i$$

  e.g. $L_{ij} f_j = \int k_i(x) f_j(x) dx$ (k_i known), or just $L_{ij} f_j = f(x_{ji})$.
- Dropping j for now, we can discretize $L_i f(x) \simeq \sum_k \tilde{L}_{ik} f(x_k)$.
- So $L_i f(x) \simeq \sum_k \tilde{L}_{ik} \tilde{\mathbf{X}}_k \boldsymbol{\beta}$, where $\tilde{\mathbf{X}}_k$ is k^{th} row of model matrix evaluating $f(x)$ at the points x_k.
- Then the model matrix for $L_i f(x)$ is $\tilde{\mathbf{L}}\tilde{\mathbf{X}}$. The penalties are just those for f.
- Hence the extended model can be written in the same general form as the simple AM.

Generalized Additive Models

- Generalizing again, we have

$$g(\mu_i) = \mathbf{A}_i\boldsymbol{\theta} + \sum_j L_{ij} f_j(x_j), \quad y_i \sim \text{EF}(\mu_i, \phi)$$

  g is a known smooth monotonic link function, EF an exponential family distribution so that $\text{var}(y_i) = V(\mu_i)\phi$.
- Set up model matrix and penalties as before.
- Estimate $\boldsymbol{\beta}$ by penalized MLE. Defining the *Deviance*. $D(\boldsymbol{\beta}) = 2\{l_{\max} - l(\boldsymbol{\beta})\}$ ($l_{\max}$ is saturated log likelihood)...

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} D(\boldsymbol{\beta}) + \sum_j \lambda_j \boldsymbol{\beta}^{\mathsf{T}} \mathbf{S}_j \boldsymbol{\beta}$$

- $\boldsymbol{\lambda}$ estimation is by generalizations of GCV, REML etc.

GAM computation: $\hat{\boldsymbol{\beta}}|\mathbf{y}$

- Penalized likelihood maximization is by Penalized IRLS.
- Initialize $\hat{\boldsymbol{\eta}} = g(\mathbf{y})$ and iterate the following to convergence.
  1. Compute pseudodata $z_i = g'(\hat{\mu}_i)(y_i - \hat{\mu}_i)/\alpha_i + \hat{\eta}_i$ and iterative weights, $w_i = \alpha_i / \left\{V(\hat{\mu}_i) g'(\hat{\mu}_i)^2\right\}$ as for any GLM.
  2. Compute a revised $\boldsymbol{\beta}$ estimate

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \sum_i w_i (z_i - \mathbf{X}_i\boldsymbol{\beta})^2 + \sum \lambda_j \boldsymbol{\beta}^{\mathsf{T}} \mathbf{S}_j \boldsymbol{\beta}$$

     and hence revised estimates $\hat{\boldsymbol{\eta}}$ and $\hat{\boldsymbol{\mu}}$.
- $\alpha_i = 1 + (y_i - \hat{\mu}_i)(V_i'/V_i + g_i''/g_i')$ gives Newton's method.
- $\alpha_i = 1$ gives *Fisher scoring*, where the expected Hessian of the likelihood replaces the actual Hessian in Newton's method.
- Newton based versions of w_i and z_i are best here, as it makes $\boldsymbol{\lambda}$ estimation easier.

EDF, $\boldsymbol{\beta}|\mathbf{y}$ and $\hat{\phi}$

- Let $\mathbf{S} = \sum_j \lambda_j \mathbf{S}_j$ and $\mathbf{W} = \text{diag}\{E(w_i)\}$ (Fisher version).
- The Effective Degrees of Freedom matrix becomes

$$\mathbf{F} = (\mathbf{X}^{\mathsf{T}}\mathbf{W}\mathbf{X} + \mathbf{S})^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{W}\mathbf{X}$$

- Then the EDF is $\text{tr}(\mathbf{F})$. EDFs for individual smooths are found by summing the F_{ii} values for their coefficients.
- In the $n \to \infty$ limit

$$\boldsymbol{\beta}|\mathbf{y} \sim N(\hat{\boldsymbol{\beta}}, (\mathbf{X}^{\mathsf{T}}\mathbf{W}\mathbf{X} + \mathbf{S})^{-1}\phi)$$

- The scale parameter can be estimated by

$$\hat{\phi} = \sum_i w_i(z_i - \mathbf{X}_i\hat{\boldsymbol{\beta}})^2/\{n - \text{tr}(\mathbf{F})\}.$$

$\boldsymbol{\lambda}$ estimation

- There are 2 basic computational strategies for $\boldsymbol{\lambda}$ selection.
  1. Single iteration schemes estimate $\boldsymbol{\lambda}$ at each PIRLS iteration step, by applying GCV, REML or whatever to the working penalized linear model. This approach need not converge.
  2. Nested iteration, defines a $\boldsymbol{\lambda}$ selection criterion in terms of the model deviance and optimizes it directly. Each evaluation of the criterion requires an 'inner' PIRLS to obtain $\hat{\boldsymbol{\beta}}_\lambda$. This converges, since a properly defined function of $\boldsymbol{\lambda}$ is optimized.
- The second option is usually preferable on grounds of reliability, but the first option can be made very memory efficient with very large datasets.
- The first option simply uses the smoothness selection criteria for the linear model case, but the second requires that these be extended...

Deviance based $\boldsymbol{\lambda}$ selection criteria

- Mallows' C_p/ UBRE generalizes to

$$\mathcal{V}_a = D(\hat{\boldsymbol{\beta}}_\lambda) + 2\phi \mathrm{tr}(\mathbf{F})$$

- GCV generalizes to

$$\mathcal{V}_g = nD(\hat{\boldsymbol{\beta}}_\lambda)/\{n - \mathrm{tr}(\mathbf{F})\}^2$$

- Laplace approximate (negative twice) REML is

$$\mathcal{V}_r = \frac{D(\hat{\boldsymbol{\beta}}) + \hat{\boldsymbol{\beta}}^{\mathsf{T}}\mathbf{S}\hat{\boldsymbol{\beta}}}{\phi} - 2l_s(\phi) + (\log|\mathbf{X}^{\mathsf{T}}\mathbf{W}\mathbf{X} + \mathbf{S}| - \log|\mathbf{S}|_+) - M_p \log(2\pi\phi).$$

Nested iteration computational strategy

- Optimization wrt $\boldsymbol{\rho} = \log \boldsymbol{\lambda}$ is by Newton's method, using analytic derivatives.
- For each trial $\boldsymbol{\lambda}$ used by Newton's method...
  1. Re-parameterize for maximum numerical stability in computing $\hat{\boldsymbol{\beta}}$ and terms like $\log|\mathbf{S}|_+$.
  2. Compute $\hat{\boldsymbol{\beta}}$ by PIRLS (full Newton version).
  3. Calculate derivatives of $\hat{\boldsymbol{\beta}}$ wrt $\boldsymbol{\rho}$ by implicit differentiation.
  4. Evaluate the $\boldsymbol{\lambda}$ selection criterion and its derivatives wrt $\boldsymbol{\rho}$
- ...after which all the ingredients are in place for Newton's method to propose a new $\boldsymbol{\lambda}$ value.
- As usual with Newton's method, some step halving may be needed, and the Hessian will have to be peturbed if it is not positive definite.

One last generalization: GAMM

- A generalized additive mixed model has the form

$$g(\mu_i) = \mathbf{A}_i\boldsymbol{\theta} + \sum_j L_{ij} f_j(x_j) + \mathbf{Z}_i\mathbf{b}, \quad \mathbf{b} \sim N(\mathbf{0}, \boldsymbol{\psi}), \quad y_i \sim \text{EF}(\mu_i, \phi)$$

- . . . actually this is not much different to a GAM. The random effects term $\mathbf{Zb}$ is just like a smooth with penalty $\mathbf{b}^{\mathsf{T}}\boldsymbol{\psi}^{-1}\mathbf{b}$.
- If $\boldsymbol{\psi}^{-1}$ can be written in the form $\sum_k \lambda_k \mathbf{S}_k$ then the GAMM can be treated *exactly* like a GAM. (`gam`).
- Alternatively, using the mixed model representation of the smooths, the GAMM can be written in standard GLMM form and estimated as a GLMM. (`gamm`/`gamm4`).
- The latter option is often preferable when there are many random effects, and the former when there are fewer.

Summary

- A GAM is simply a GLM in which the linear predictor partly depends linearly on some unknown smooth functions.
- GAMs are estimated by a penalized version of the method used to fit GLMs.
- An extra criterion has to be optimized to find the smoothing parameters.
- A GAMM is simply a GLMM in which the linear predictor partly depends linearly on some unknown smooth functions.
- From the mixed model representation of smooths, GAMMs can be estimated as GAMs or GLMMs.
- Bayesian results are useful for inference.

A toolbox of smooths

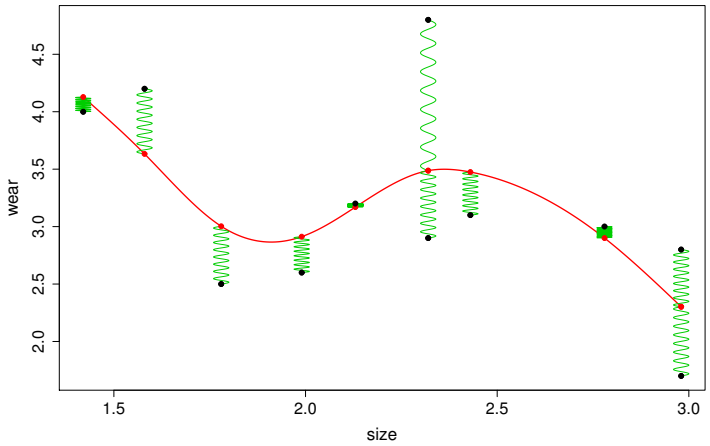
Simon Wood
Mathematical Sciences, University of Bath, U.K.

Smooths for semi-parametric GLMs

- The piecewise linear smoother is not bad, but we can find better and more general basis-penalty smoothers for a variety of modelling purposes.
- In one dimension there are several alternatives, and not alot to choose between them.
- In 2 or more dimensions there is a major choice to make.
  - If the arguments of the smooth function are variables which all have the same units (e.g. spatial location variables) then an *isotropic* smooth may be appropriate. This will tend to exhibit the same degree of flexibility in all directions.
  - If the relative scaling of the covariates of the smooth is essentially arbitrary (e.g. they are measured in different units), then *scale invariant* smooths should be used, which do not depend on this relative scaling.

Splines

- All the smooths covered here are based on *splines*. Here's the basic idea ...

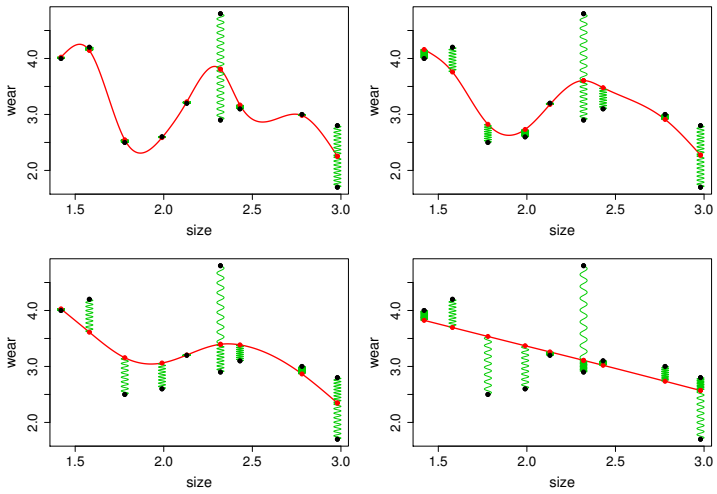


- Mathematically the red curve is the *function* minimizing

$$\sum_i (y_i - f(x_i))^2 + \lambda \int f''(x)^2 dx.$$

Splines have variable stiffness

- Varying the flexibility of the strip (i.e. varying λ) changes the *spline function* curve.



- But irrespective of λ the spline functions always have the same basis.

Why splines are special

- We can produce splines for a variety of penalties, including for functions of several variables. e.g.

$$\int f'''(x)^2 dx \quad \text{or} \quad \int\int f_{xx}(x,z)^2 + 2f_{xz}(x,z)^2 + f_{zz}(x,z)^2 dxdz$$

- Splines always have an n dimensions basis - quadratic penalty representation.
- If $y_i = g(x_i)$ and f is the cubic spline interpolating x_i, y_i then

$$\max |f - g| \leq \frac{5}{384} \max(x_{i+1} - x_i)^4 \max(g'''')$$

  (best possible — end conditions are a bit unusual for this).
- Bases that are optimal for approximating known functions are a good starting point for approximating unknown functions.

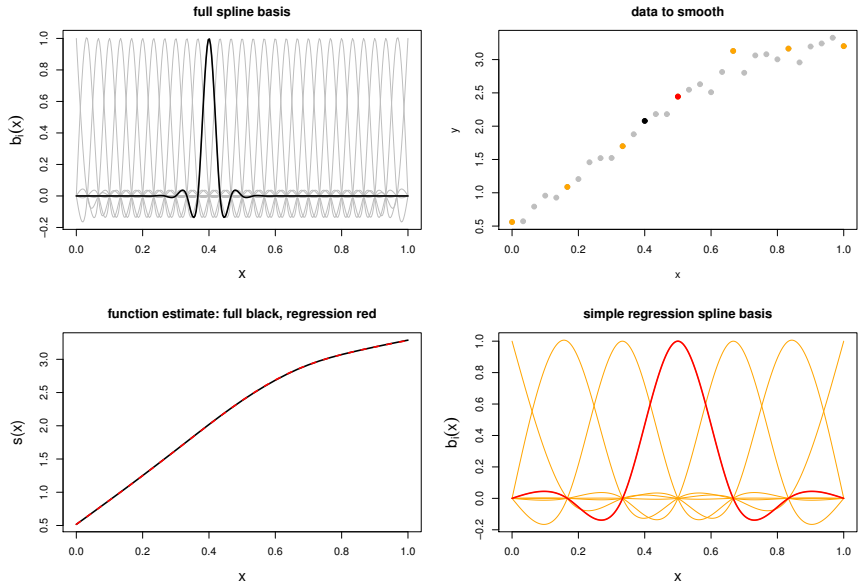
Penalized regression splines

- Full splines have one basis function per data point.
- This is computationally wasteful, when penalization ensures that the *effective* degrees of freedom will be much smaller than this.
- Penalized regression splines simply use fewer spline basis functions. There are two alternatives:
  1. Choose a representative subset of your data (the 'knots'), and create the spline basis as if smoothing only those data. Once you have the basis, use it to smooth all the data.
  2. Choose how many basis functions are to be used and then solve the problem of finding the set of this many basis functions that will optimally approximate a full spline.

  I'll refer to 1 as *knot based* and 2 as *eigen based*.

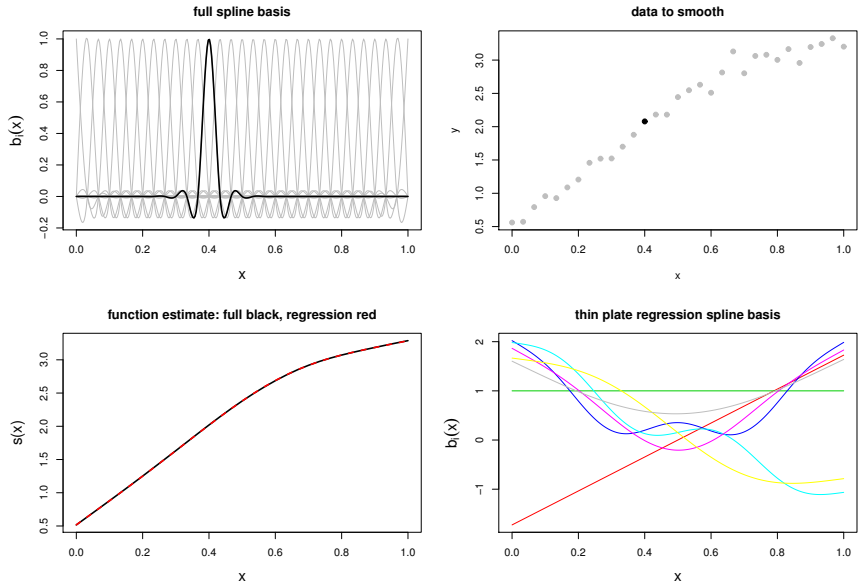
Knot based example: "cr"

- In mgcv the "cr" basis is a knot based approximation to the minimizer of $\sum_i (y_i - f(x_i))^2 + \lambda \int f''(x)^2 dx$ — a cubic spline. "cc" is a cyclic version.



Eigen based example: "tp"

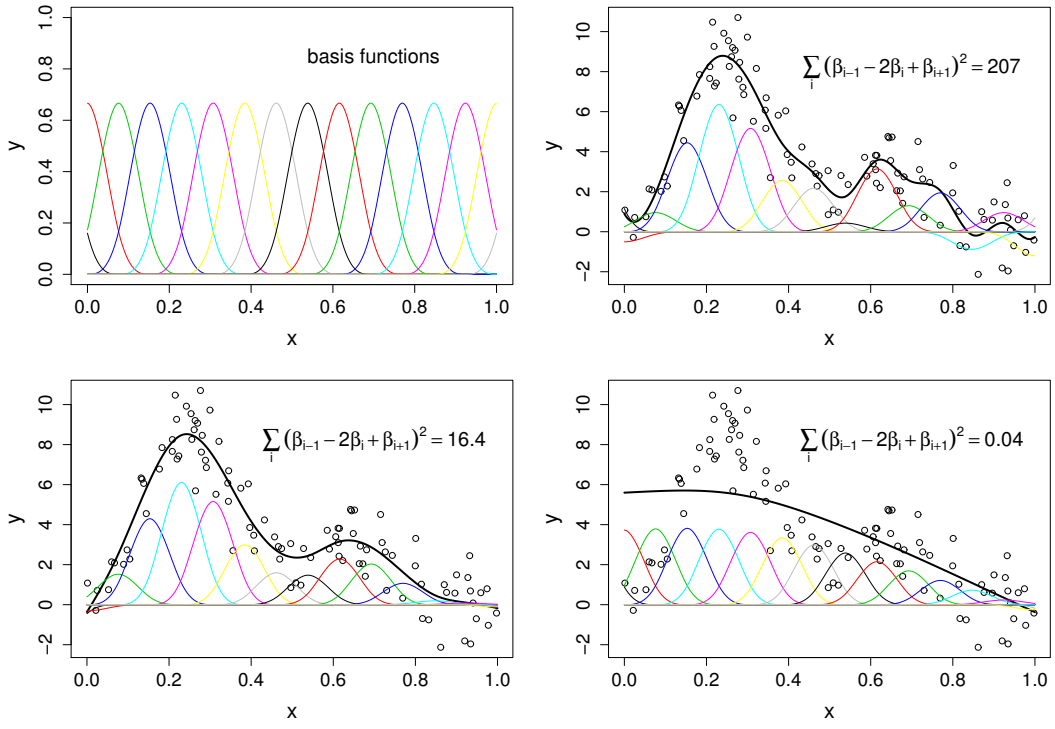
- The "tp", *thin plate regression spline* basis is an eigen approximation to a thin plate spline (including cubic spline in 1 dimension).



P-splines: "ps" & "cp"

- There are many equivalent spline bases.
- With bases for which all the basis functions are translations of each other, it is sometimes possible to penalize the coefficients of the spline directly, rather than penalizing something like $\int f''(x)^2 dx$.
- Eilers and Marx coined the term 'P-splines' for this combination of spline bases with direct discrete penalties on the basis coefficients.
- P-splines allow a good deal of flexibility in the way that bases and penalties are combined.
- However splines with derivative based penalties have good approximation theoretic properties bound up with the use of derivative based penalties, and as a result tend to slightly out perform P-splines for routine use.

P-spline illustration



An adaptive smoother

- Can let the p-spline penalty vary with the predictor. e.g.

$$\mathcal{P}_a = \sum_{k=2}^{K-1} \omega_k(\beta_{k-1} - 2\beta_k + \beta_{k+1})^2 = \boldsymbol{\beta}^{\mathrm{T}}\mathbf{D}^{\mathrm{T}}\mathrm{diag}(\boldsymbol{\omega})\mathbf{D}\boldsymbol{\beta}$$

  where $\mathbf{D} = \begin{bmatrix} 1 & -2 & 1 & 0 & \cdot \\ 0 & 1 & -2 & 1 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix}$.

- Now let ω_k vary smoothly with k, using a B-spline basis, so that $\boldsymbol{\omega} = \mathbf{B}\boldsymbol{\lambda}$, where $\boldsymbol{\lambda}$ is the vector of basis coefficients.
- So, writing $\mathbf{B}_{\cdot k}$ for the k^{th} column of $\mathbf{B}$ we have

$$\boldsymbol{\beta}^{\mathrm{T}}\mathbf{D}^{\mathrm{T}}\mathrm{diag}(\boldsymbol{\omega})\mathbf{D}\boldsymbol{\beta} = \sum_k \lambda_k \boldsymbol{\beta}^{\mathrm{T}}\mathbf{D}^{\mathrm{T}}\mathrm{diag}(\mathbf{B}_{\cdot k})\mathbf{D}\boldsymbol{\beta} = \sum_k \lambda_k \boldsymbol{\beta}^{\mathrm{T}}\mathbf{S}_k\boldsymbol{\beta}.$$

1 dimensional smoothing in `mgcv`

- Smooth functions are specified by terms like `s(x,bs="ps")`, on the rhs of the model formula.
- The `bs` argument of `s` specifies the class of basis...

  `"cr"` knot based cubic regression spline.

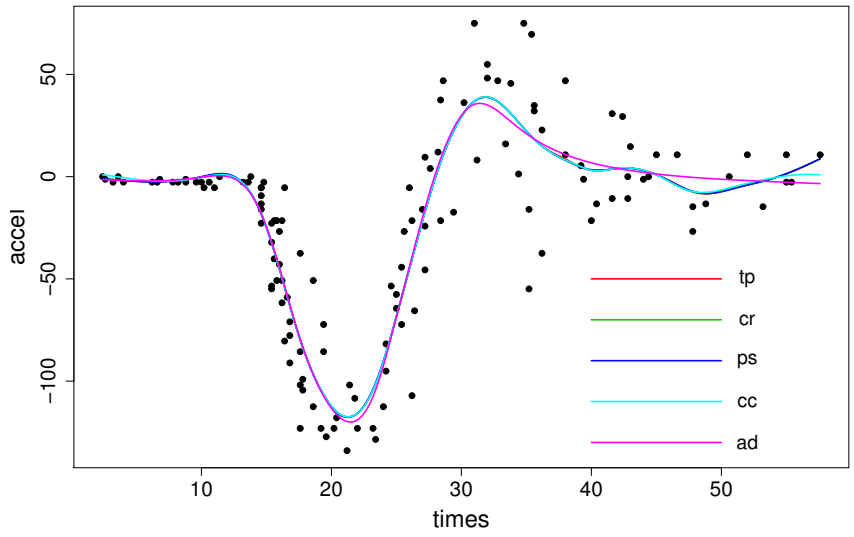
  `"cc"` cyclic version of above.

  `"ps"` Eilers and Marx style p-splines, with flexibility as to order of penalties and basis functions.

  `"ad"` adaptive smoother in which strength of penalty varies with covariate.

  `"tp"` thin plate regression spline. Optimal low rank eigen approx. to a full spline: flexible order penalty derivative.
- Smooth classes can be added (`?smooth.construct`).

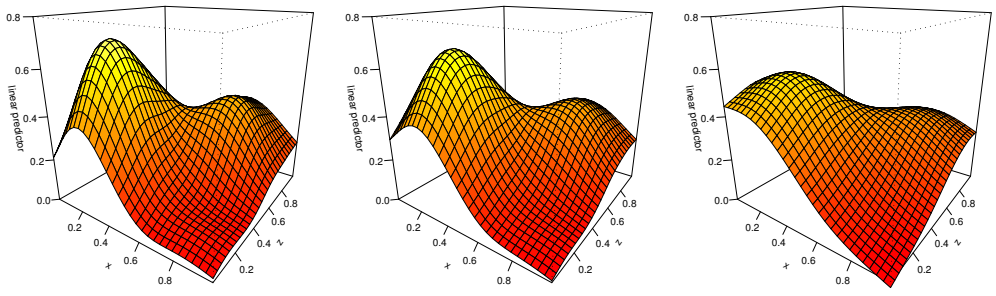
1D smooths compared



- So cubic regression splines, P-splines and thin plate regression splines give very similar results.
- A cyclic smoother is a little different, of course.
- An adaptive smoother can look very different.

Isotropic smooths

- One way of generalizing splines from 1D to several D is to turn the flexible strip into a flexible sheet (hyper sheet).
- This results in a *thin plate spline*. It is an *isotropic* smooth.
- Isotropy may be appropriate when different covariates are naturally on the same scale.
- In `mgcv` terms like `s(x,z)` generate such smooths.



Thin plate spline details

- In 2 dimensions a thin plate spline is the function minimizing

$$\sum_i \{y_i - f(x_i, z_i)\}^2 + \lambda \int f_{xx}^2 + 2f_{xz}^2 + f_{zz}^2 dxdz$$

- This generalizes to any number of dimensions, d, and any order of differential, m, such that $2m > d + 1$.
- *Any* thin plate spline is computed as

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{n} \delta_i \eta_i(\mathbf{x}) + \sum_{i=1}^{M} \alpha_i \phi_i(\mathbf{x})$$

where η_i and ϕ_i are basis functions of known form and $\boldsymbol{\alpha}, \boldsymbol{\delta}$ minimize $\|\mathbf{y} - \mathbf{E}\boldsymbol{\delta} - \mathbf{T}\boldsymbol{\alpha}\|^2 + \boldsymbol{\delta}^{\mathrm{T}}\mathbf{E}\boldsymbol{\delta}$ s.t. $\mathbf{T}^{\mathrm{T}}\boldsymbol{\delta} = \mathbf{0}$, where $\mathbf{E}$ and $\mathbf{T}$ are computed using the η_i and ϕ_i.

Thin plate regression splines

- Full thin plate splines have n parameters and $O(n^3)$ computational cost.
- This drops to $O(k^3)$ if we replace $\mathbf{E}$ by its rank k eigen approximation, $\mathbf{E}_k$, at cost $O(n^2 k)$. Big saving if $k \ll n$
- Out of all rank k approximations this one minimizes

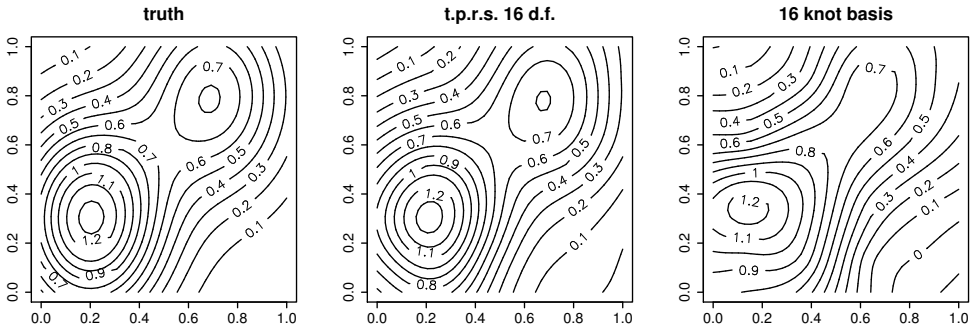
$$\max_{\delta \neq \mathbf{0}} \frac{\|(\mathbf{E} - \mathbf{E}_k)\delta\|}{\|\delta\|} \quad \text{and} \quad \max_{\delta \neq \mathbf{0}} \frac{\delta^{\mathrm{T}}(\mathbf{E} - \mathbf{E}_k)\delta}{\|\delta\|^2}$$

i.e. the approximation is somewhat optimal, and avoids choosing 'knot locations'.

- For very large datasets, randomly subsample the data the data and work out the truncated basis from the subsample, to avoid $O(n^2 k)$ eigen-decomposition costs being too high.

TPRS illustration

- As the theory suggests, the eigen approximation is quite effective. The following figure compares reconstructions of of the true function on the left, using and eigen based thin plate regression spline (middle), and one based on choosing knots. Both are rank 16 approximations.



Duchon Splines `s(x,z,bs="ds")`

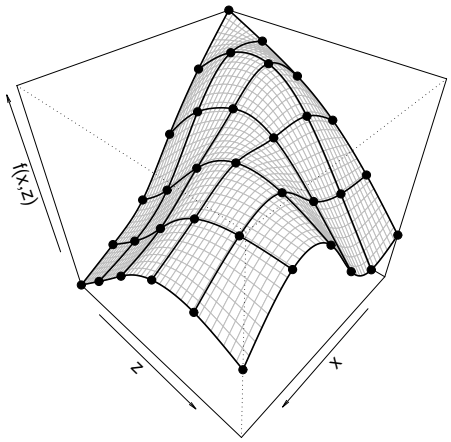
- The $m > d/2$ requirement causes thin plate splines in more than a few dimensions to be impractical, as the null space of the penalty rapidly becomes too high dimensional.
- But thin plate splines are only one special case of the splines introduced by Duchon (1977). He also considered penalties

$$\int_{\mathbb{R}^d} \|\tau\|^{2s} \sum_{\nu_1+\cdots+\nu_d=m} \frac{m!}{\nu_1!\ldots\nu_d!} \left(\mathfrak{F}\frac{\partial^m f}{\partial x_1^{\nu_1}\ldots\partial x_d^{\nu_d}}(\tau)\right)^2 \mathrm{d}\tau$$

  where $\mathfrak{F}$ denotes Fourier transform and τ is frequency.
- With $s = 0$ this is a thin plate spline penalty, but with $s > 0$ higher frequencies of the derivative field are penalized more heavily.
- Smoothers using this penalty exist if $m + s > d/2$, and have the form of a TPS, with a reduced dimensional null space. e.g. $m = 2$, $s = d/2 - 1$ gives null space dimension $d + 1$.
- Eigen-approximation is as for TPS.

Scale invariant smoothing: tensor product smooths

- Isotropic smooths assume that a unit change in one variable is equivalent to a unit change in another variable, in terms of function variability.
- When this is not the case, isotropic smooths can be poor.
- *Tensor product smooths* generalize from 1D to several D using a lattice of bendy strips, *with different flexibility in different directions*.



Tensor product smooths

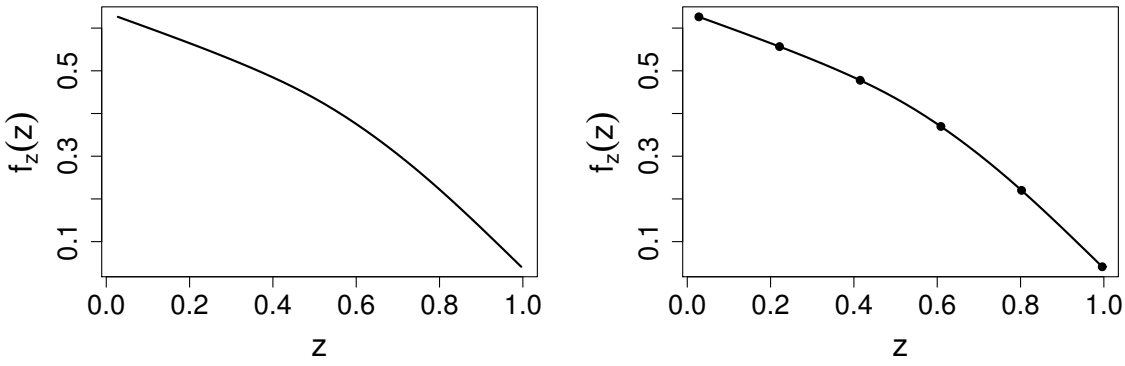
- Carefully constructed tensor product smooths are scale invariant.
- Consider constructing a smooth of x, z.
- Start by choosing *marginal* bases and penalties, as if constructing 1-D smooths of x and z. e.g.

$$f_x(x) = \sum \alpha_i a_i(x), \quad f_z(z) = \sum \beta_j b_j(z),$$

$$J_x(f_x) = \int f_x''(x)^2 dx = \boldsymbol{\alpha}^{\mathrm{T}} \mathbf{S}_x \boldsymbol{\alpha} \;\; \& \;\; J_z(f_z) = \mathcal{B}^{\mathrm{T}} \mathbf{S}_z \mathcal{B}$$

Marginal reparameterization

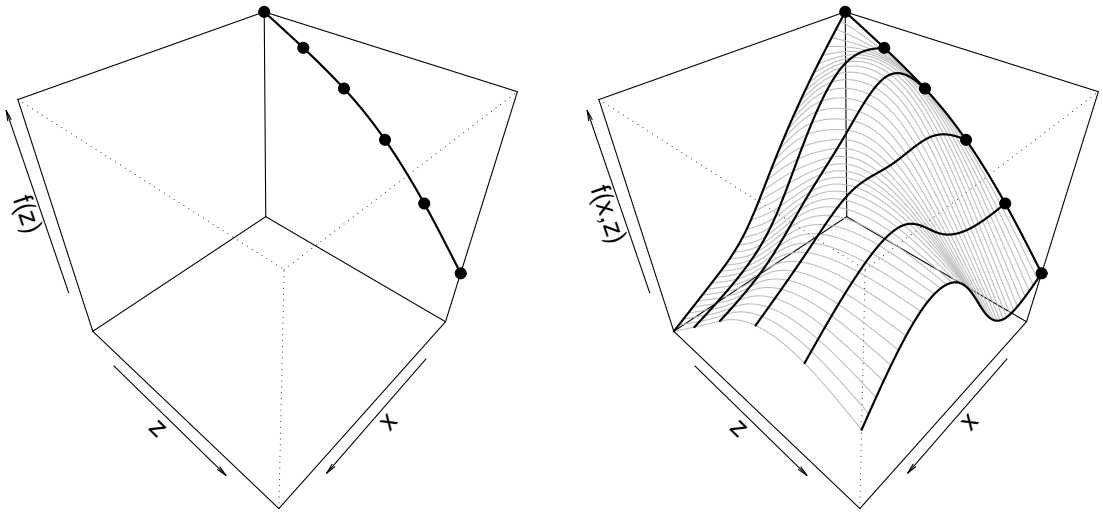
- Suppose we start with $f_z(z) = \sum_{i=1}^{6} \beta_j b_j(z)$, on the left.



- We can always re-parameterize so that its coefficients are functions heights, at knots (right). Do same for f_x.

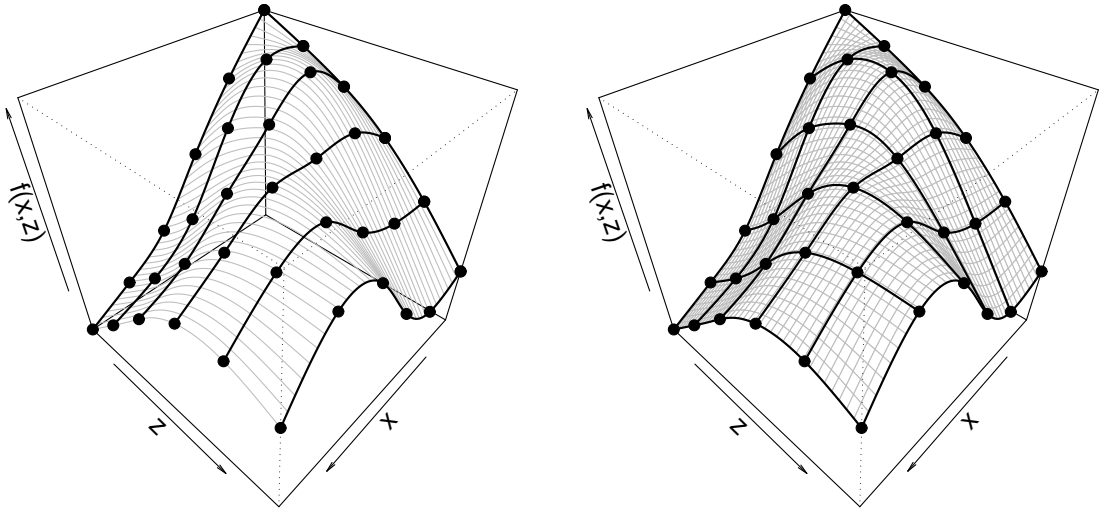
Making f_z depend on x

- Can make f_z a function of x by letting its coefficients vary smoothly with x



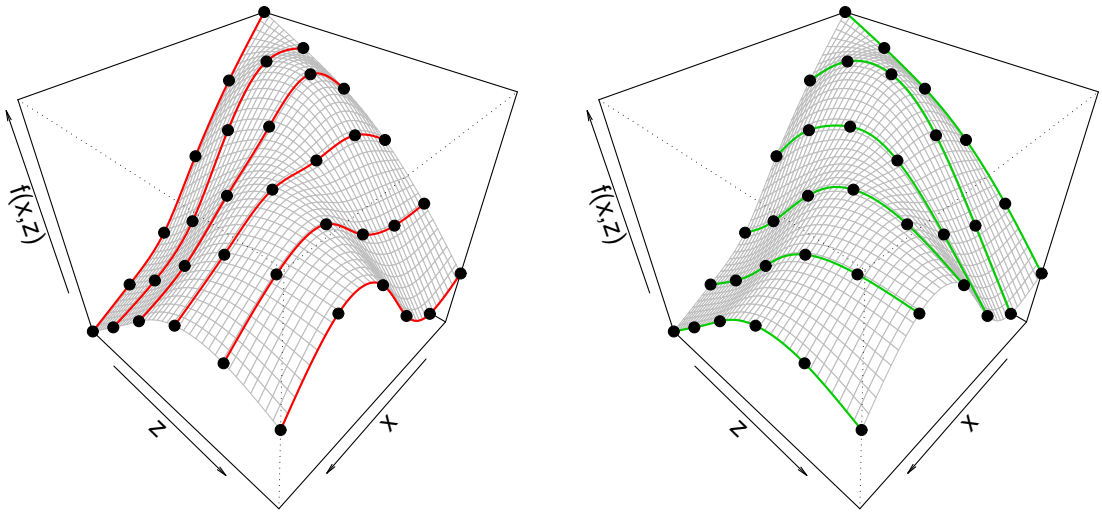
The complete tensor product smooth

- Use f_x basis to let f_z coefficients vary smoothly (left).
- Construct in symmetric (see right).



Tensor product penalties - one per dimension

- x-wiggliness: sum marginal x penalties over red curves.
- z-wiggliness: sum marginal z penalties over green curves.



Tensor product expressions

- So the tensor product basis construction gives:

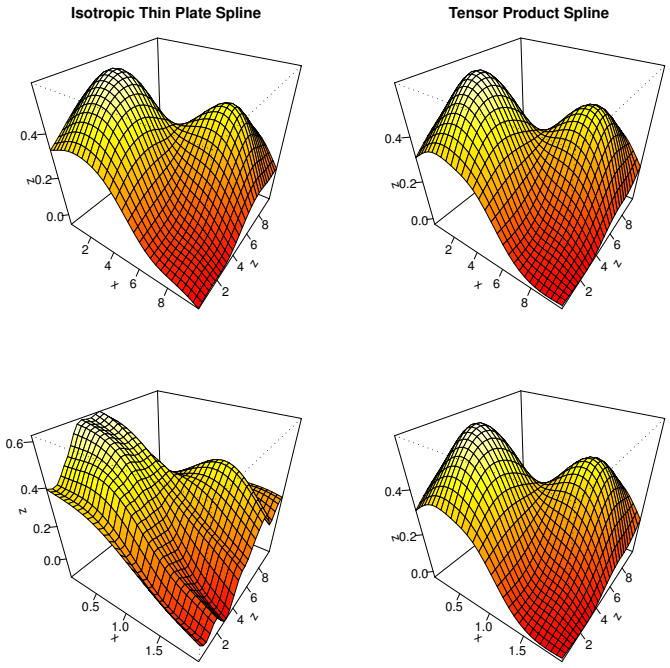
$$f(x,z) = \sum\sum \beta_{ij} b_j(z) a_i(x)$$

- With double penalties

$$J_z^*(f) = \beta^{\mathrm{T}} \mathbf{I}_I \otimes \mathbf{S}_z \beta \text{ and } J_x^*(f) = \beta^{\mathrm{T}} \mathbf{S}_x \otimes \mathbf{I}_J \beta$$

- The construction generalizes to any number of marginals and multi-dimensional marginals.
- Can start from any marginal bases & penalties (including mixtures of types).
- Note that the penalties maintain the basic meaning inherited from the marginals.

Isotropic vs. tensor product comparison



. . . each figure smooths the same data. The only modification is that x has been divided by 5 in the bottom row.

Tensor product smoothing in `mgcv`

- Tensor product smooths are constructed automatically from *marginal* smooths of lower dimension. The resulting smooth has a penalty for each marginal basis.
- `mgcv` can construct tensor product smooths from any *single penalty* smooths useable with `s` terms.
- `te` terms within the model formula invoke this construction. For example:
  - `te(x,z,v,bs="ps",k=5)` creates a tensor product smooth of x, z and v using rank 5 P-spline marginals: the resulting smooth has 3 penalties and basis dimension 125.
  - `te(x,z,t,bs=c("tp","cr"),d=c(2,1),k=c(20,5))` creates a tensor product of an isotropic 2-D TPS with a 1-D smooth in time. The result is isotropic in x,z, has 2 penalties and a basis dimension of 100. This sort of smooth would be appropriate for a location-time interaction.
- `te` terms are invariant to linear rescaling of covariates.

`t2` alternative tensor products

An alternative construction, due to Fabian Scheipl, and closely related to smoothing spline ANOVA, starts from a different marginal reparameterization

- Reparameterize each marginal smooth into unpenalized components and a component with an identity penalty.
- Form tensor product bases from each combination of unpenalized and penalized components, picking one from each margin.
- Each of the resulting bases is subject to a separate identity penalty, except for the basis made up only from unpenalized marginal components.
- The basis for the whole smooth is the sum of all these bases.
- `t2` in `mgcv` implements this using same syntax as `te`.

Other interactions with smooths `s(...,by=z)`

- Suppose we want a term of the form $f(x)z$, where z is metric.
- `s(x,by=z)` achieves this.
- An interaction with a factor variable, a, is also possible.
- `s(x,by=a)` produces a smooth of x for each level of a.
- `s(x,by=a,id="foo")` forces all these smooths to have the same smoothing parameter.
- `s(x,a,bs="fs")` is similar, but efficient with `gamm` and `gamm4` when a has many levels.
- `te/2` terms also accept `by` variables.

The basis dimension

- You have to choose the number of basis functions to use for each smooth, using the `k` argument of `s` or `te`.
- The default is essentially arbitrary.
- Provided `k` is not too small its exact value is not critical, as the smoothing parameters control the actual model complexity. However
  1. if `k` is too small then you will oversmooth.
  2. if `k` is much too large then computation will be very slow.
- Checking that k is not too small will be covered in a later segment.

Miscellanea

- Most smooths will require an identifiability condition to avoid confounding with the model intercept: `gam` handles this by automatic reparameterization.
- `gam` will also handle the side conditions required for nested smooths. e.g. `gam(y~s(x)+s(z)+s(x,z))` will work.
- However, such nested models are not always easy to interpret.
- `te`, `t2`, `s(...,bs="tp")` and `s(...,bs="ds")` can, in principle, handle any number of covariates.
- The `"ad"` basis can handle 1 or 2 covariates, but no more.

Discrete spatial smoothing: Markov random fields

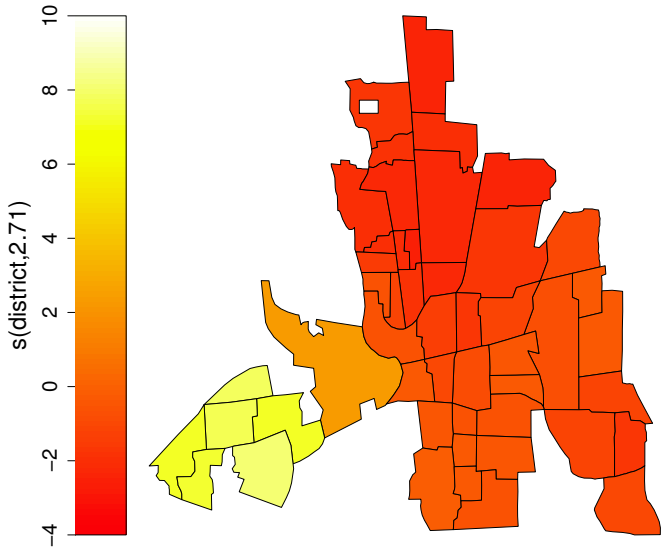
- Sometimes data come allocated to irregular partitions of space (e.g. administrative regions).
- Markov random fields area a popular way of smoothing such data.
- The smooth has a coefficient, γ_i, for each region.
- The neighbouring regions of each region are found, and a quadratic penalty constructed. If N_i is the set of indices of the neighbours of region i, then the simplest penalty is

$$\sum_i (\sum_{j \in N_i} (\gamma_i - \gamma_j))^2$$

- Eigen based rank reduction is effective here.

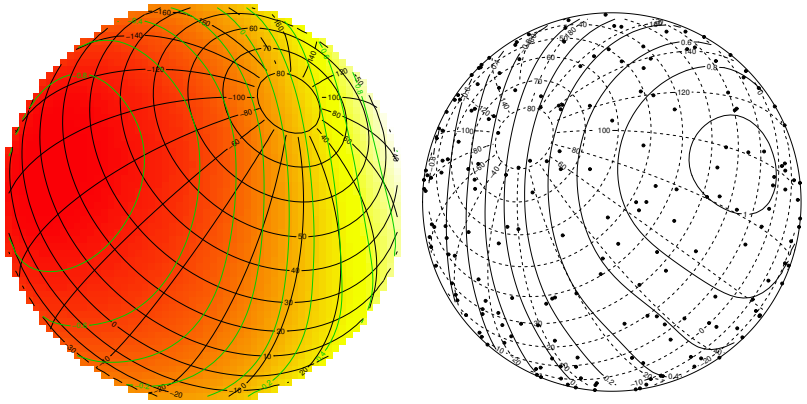
Markov random field illustration

```
data(columb.polys) ## district shapes list
xt <- list(polys=columb.polys)
gam(crime ~ s(district,bs="mrf",xt=xt),data=columb)
```



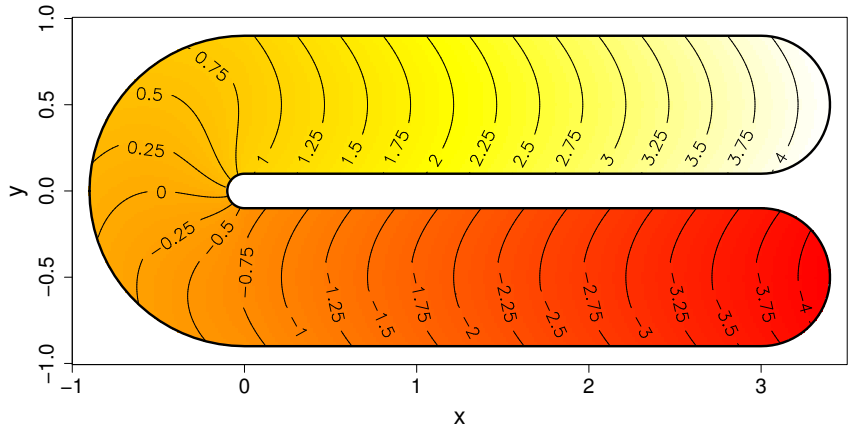
Smoothing on the globe

- Thin plate spline like smoothers can be constructed for the sphere [`s(la,lo,bs="sos")`]...



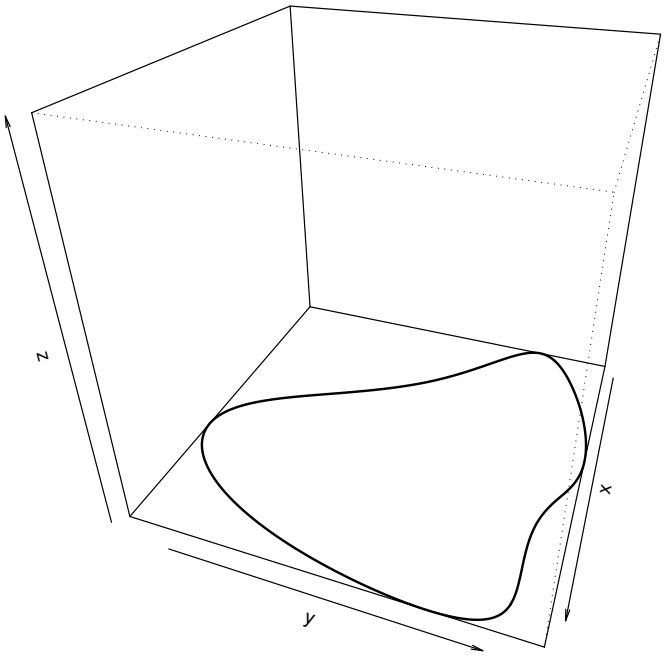
Finite area smoothing

- Suppose now want to smooth samples from this function

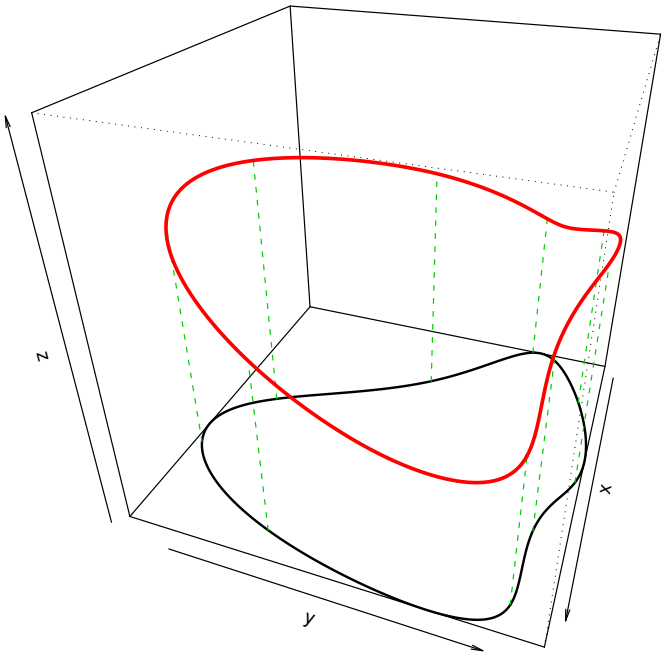


- . . . without ‘smoothing across’ the gap in the middle?
- Let’s use a soap film . . .

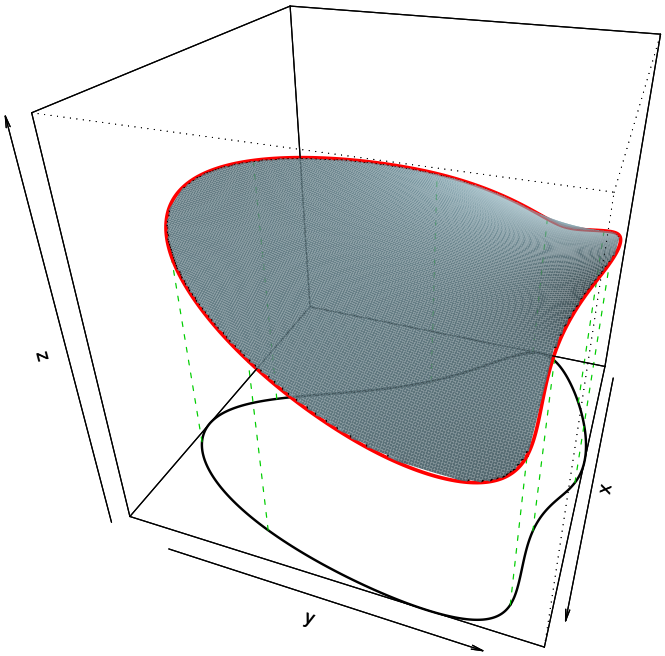
The domain



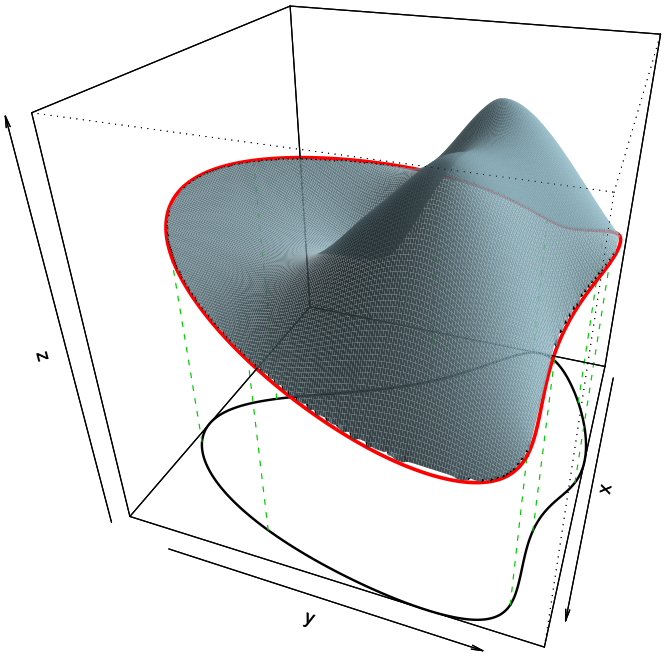
The boundary condition



The boundary interpolating film

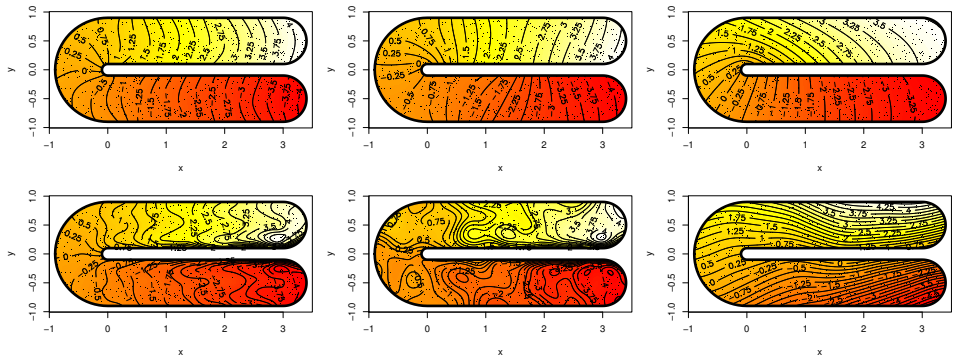


Distorted to approximate data



Soap film smoothers

- Mathematically this smoother turns out to have a basis-penalty representation.
- See package `soap` from my web page.
- It also turns out to work. . .



Random effect `s(...,bs="re")`

- Statistically, smooths consist of a basis and a quadratic penalty, where the penalty matrix can be treated as the generalized inverse of a covariance matrix.
- They can therefore be estimated as random effects.
- Reversing this, we can treat simple random effects as (zero dimensional) smooths.
- `s(a,b,bs="re")` creates a terms with model matrix `model.matrix(~a:b-1)` and a scaled identity penalty/covariance matrix.
- Any number of covariates are possible.
- Function `gam.vcomp` helps later interpretation by converting smoothing parameters to variance components.

Summary

- We can treat simple random effects as 0 dimensional smooths.
- In 1 dimension, the choice of basis is not critical. The main decisions are whether it should by cyclic or not and whether or not it should be adaptive.
- In 2 dimensions and above the key decision is whether an isotropic smooth, `s`, or a scale invariant smooth, `te`/`t2`, is appropriate. (`te`/`2` terms may be isotropic in some marginals.)
- Smooths and factors can be made to interact.
- Spatial smoothing may sometimes require more specialized smoothers (Markov random fields, spherical splines, finite area smooths).
- The basis dimension is a modelling decision that should be checked.

Checking & Selecting GAMs

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

Model checking overview

- Since a GAM is just a penalized GLM, residual plots should be checked exactly as for a GLM.
- It should be checked that smoothing basis dimension is not restrictively low. Defaults are essentially arbitrary.
- The GAM analogue of co-linearity is often termed 'concurvity'. It occurs when one predictor variable could be reasonably well modelled as a smooth function of another predictor variable. Like co-linearity it is statistically destabilising and complicates interpretation, so is worth checking for.

Residual checking

- Deviance, Pearson, working and raw residuals are defined for a GAM in the same way as for any GLM.
- In `mgcv` the `residuals` function will extract them, defaulting to deviance residuals.
- Residuals should be plotted against
  1. fitted values.
  2. predictor variables (those included and those dropped).
  3. time, if the data are temporal.
- Residual plotting aims to show that there is something wrong with the model assumptions. It's good to fail.
- The key assumptions are
  1. The assumed mean variance relationship is correct, so that scaled residuals have constant variance.
  2. The response data are independent, so that the residuals appear approximately so.

Distribution checking

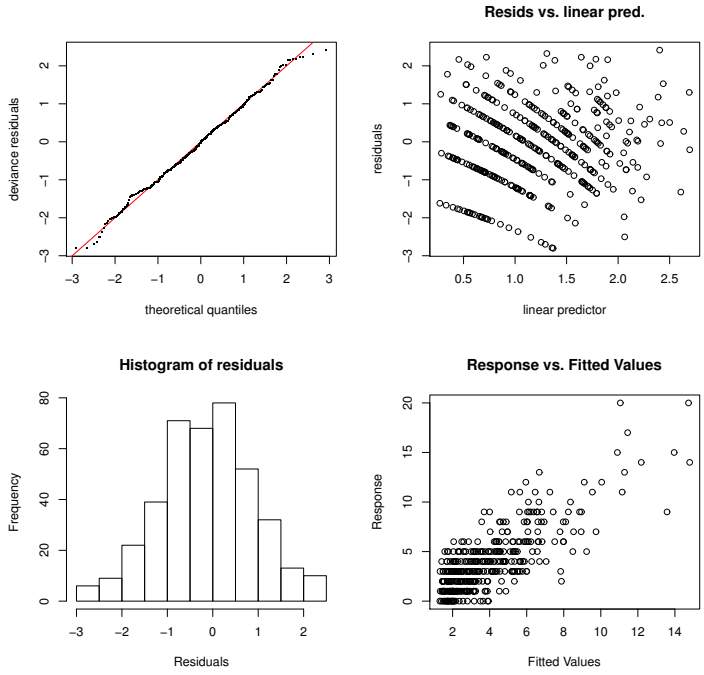
- If the independence and mean-variance assumptions are met then it is worth checking the distributional assumption more fully.
- The implication of quasi-likelihood theory is that provided the mean variance relationship is right, the other details of the distribution are not important for many inferential tasks.
- QQ-plots of residuals against standard normal quantiles can be misleading in some circumstances: for example low mean Poisson data, with many zeroes.
- It is better to obtain the reference quantiles for the deviance residuals by repeated simulation of response data, and hence residuals, from the fitted model. `mgcv` function `qq.gam` will do this for you.
- `gam.check` produces some default residual plots for you.

Residual checking example

```
> b <- gam(y~s(x0)+s(x1,x2,k=40)+s(x3)+s(x4),
+ family=poisson,data=dat,method="REML")
>
> gam.check(b)

Method: REML   Optimizer: outer newton
full convergence after 8 iterations.
Gradient range [-0.0001167555,3.321004e-05]
(score 849.8484 & scale 1).
Hessian positive definite, eigenvalue range [9.66288e-05,10.52249].

[edited]
```

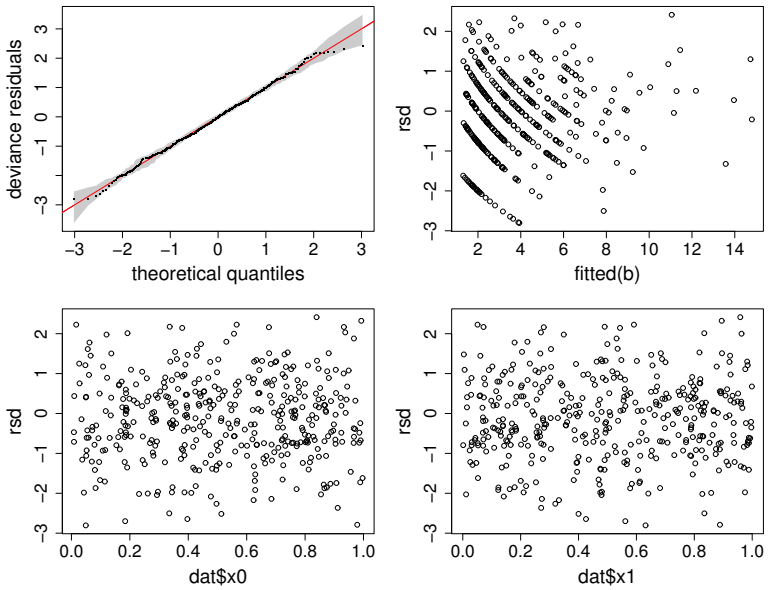
- The printed output is rather detailed information about smoothing parameter estimation convergence.
- 4 residual plots are produced, the first is from `qq.gam`, unless quasi-likelihood is used, in which case we have to fall back on a normal QQ-plot (but anyway don't care about this plot). The rest are self explanatory.

`gam.check` plots



More residual plots

```
rsd <- residuals(b)
qq.gam(b,rep=100); plot(fitted(b),rsd)
plot(dat$x0,rsd); plot(dat$x1,rsd)
```



Checking *k* the basis dimension

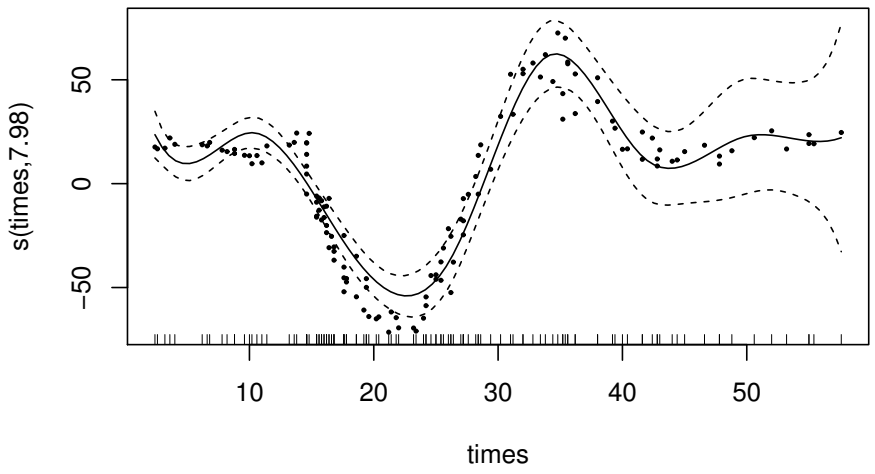
- Provided it is not restrictively low the choice of basis dimension, *k*, is not critical, because the effective degrees of freedom of a term are primarily controlled by the smoothing penalty.
- But it *must* be checked that *k* is *not* restrictively low — default values are arbitrary.
- Four checking methods are useful.
  1. Plot *partial residuals* over term estimates, looking for systematic departures.
  2. Test the residuals for residual pattern.
  3. Try re-smoothing the model deviance residuals with respect to the covariate(s) of interest using a higher *k*, to see if any pattern is found.
  4. Try re-fitting the model with increased *k* and see if the smoothness selection criterion increases substantially.
- 1 and 2 should be routine. 3 and 4 are useful if you are suspicious, but are also more time consuming.

Partial residuals

- Partial residuals are specific to each smooth term.
- Recall that the *working residuals* for a GLM are the weighted residuals from the working linear model using in the IRLS fitting scheme, at convergence.
- The partial residuals for f_j are the working residuals that you obtain using a linear predictor with $\hat{f}_j$ set to zero. These are the same as the working residual added to $\hat{f}_j$.
- The partial residuals should look like a random scatter around the smooth.
- Systematic deviation of the mean partial residual from $\hat{f}_j$ can indicate that k is too low.

Partial residual example

```
library(MASS)
m <- gam(accel~s(times,bs="ps"),data=mcycle,weights=w)
plot(m,residuals=TRUE,pch=19,cex=.3)
```



. . . note the systematic pattern in the departure of the partial residuals from the smooth. Should increase k.

A simple residual test

- An estimate of scale parameter ϕ can be obtained by differencing scaled residuals.
- Differencing residuals that are neighbours according to some covariate(s) should give an estimate of ϕ that is statistically indistinguishable from a differencing estimate obtained with any random ordering of residuals, *if there is no residual pattern with respect to the covariates*.
- This is the basis for a simple, and rapid, randomisation test.
- If pattern is detected, then it may indicate that *k* is too low.
- . . . but care is needed: pattern may also be caused by mean-variance problems, missing covariates, structural infelicities, zero inflation etc. . .

Residual test example

```
> gam.check(m)
[edited]
Basis dimension (k) checking results. Low p-value (k-index<1) may
indicate that k is too low, especially if edf is close to k'.

            k'   edf k-index p-value
s(times) 9.000 7.981   0.529       0
```

- k-index is ratio of neighbour differencing scale estimate to fitted model scale estimate.
- k' is the maximum possible EDF for the term.
- Here a low p-value coupled with high EDF suggests k may be too low.

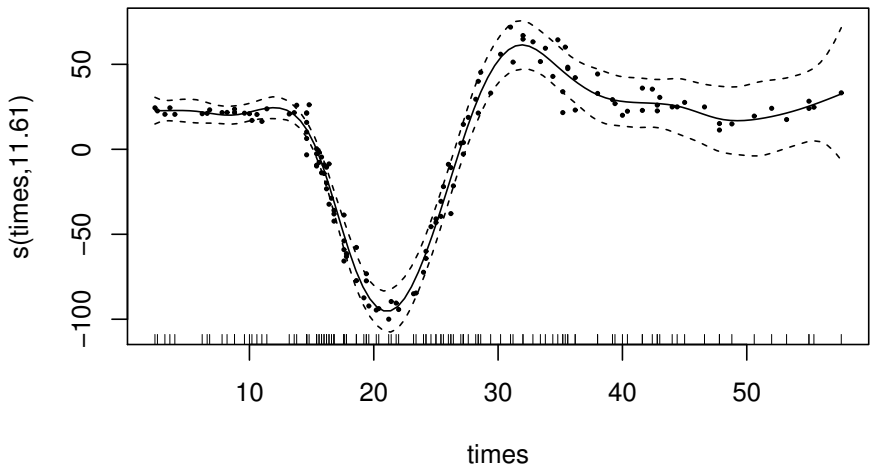
Alternative check example

```
> rsd <- residuals(m)
> ## smooth residuals with k doubled, to check pattern
> ## gamma>1 favours smoother models.
> gam(rsd~s(times,bs="ps",k=20),data=mcycle,gamma=1.4)

Family: gaussian
Link function: identity

Formula:
rsd ~ s(times, bs = "ps", k = 20)

Estimated degrees of freedom:
12.875  total = 13.87489

GCV score: 83.21058
```

This approach is not really needed for a single term model, but is usefully efficient, relative to refitting with larger k, when there are many terms present.

k fixed

```
m <- gam(accel~s(times,bs="ps",k=20),data=mcycle,weights=w)
```



- Further check now find no suggestion that k is too low.
- There are some differences in the k required with different bases. The default ”tp” basis gives acceptable results with k=10 (it is designed to be the optimal basis at a given k).

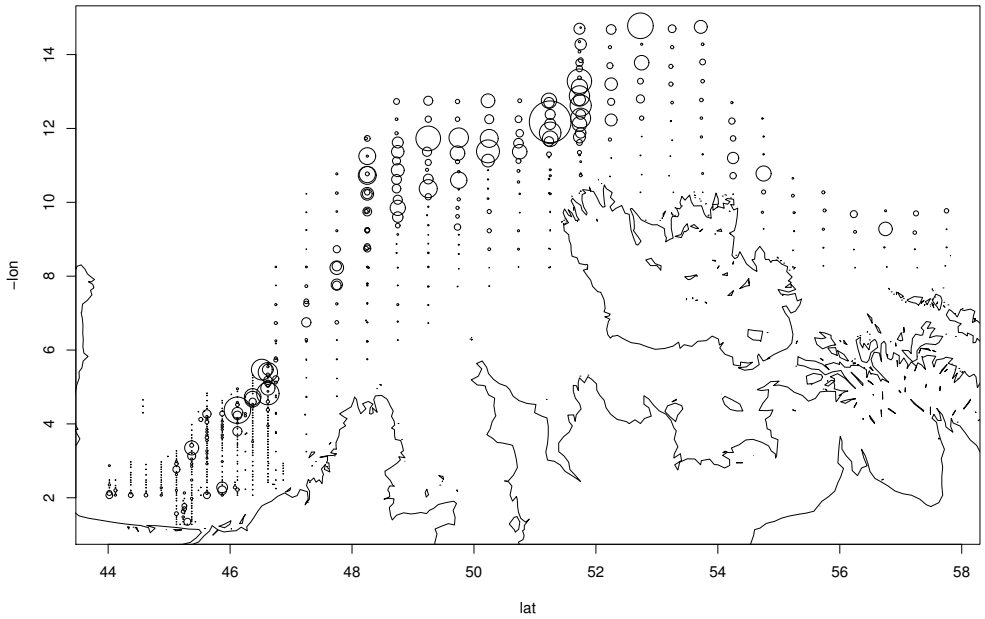
Concurvity

- Consider a model containing smooths f_1 and f_2.
- We can decompose $f_2 = f_{12} + f_{22}$ where f_{12} is the part of f_2 representable in the space of f_1, while f_{22} is the remaining component, which lies exclusively in the space of f_2.
- A measure of concurvity is $\alpha = \|f_{12}\|^2/\|f_2\|^2$, leading to 3 estimates
  1. $\hat{\alpha} = \|\hat{f}_{12}\|^2/\|\hat{f}_2\|^2$.
  2. The maximum value that α could take for any estimates, using the given bases for f_1 and f_2.
  3. The ratio of the 'size' of the basis for f_{12} relative to the basis for f_2, using some matrix norm.
- Function `concurvity` reports 1 as 'observed', 2 as 'worst' and 3 as 'estimated'. All are in $[0, 1]$.
- The measure generalizes to more components.

Concurvity consequences

- Concurvity can make interpretation difficult.
- Spatial confounding is a common example: you need a spatial effect in the model, but all the other covariates are somehow functions of space.
- A technical problem is that smoothing parameter estimates may become highly correlated and variable, which degrades the performance of inferential methods that are conditional on those estimates (confidence intervals and p-values).
- Under ML or REML smoothness selection `sp.vcov` and `gam.vcomp` can help diagnose this problem.
- Model averaging over the sampling distribution of the smoothing parameters can help in severe cases.

Concurvity/Spatial confounding example



`concurvity` example

```
library(gamair)
data(mack)
gm <- gam(egg.count~s(lon,lat,k=100)+s(I(b.depth^.5))+s(salinity)+s(temp.20m)
          +offset(log.net.area),data=mack,family=quasipoisson,method="REML")
concurvity(gm)
                  para s(lon,lat) s(I(b.depth^0.5)) s(salinity) s(temp.20m)
worst    1.063513e-17  0.9899778         0.9874163   0.9300386   0.9621984
observed 1.063513e-17  0.8308139         0.9518048   0.9232639   0.8736039
estimate 1.063513e-17  0.5500618         0.9360886   0.8952500   0.9294740
```

- This output shows the concurvity of each term with all the other terms in the model. Basically space is confounded with everything.
- With spatial confounding it sometimes helps to increase the smoothing parameter for space, e.g. until the REML score is just significantly different to its maximum.

```
gm1 <- gam(egg.count~s(lon,lat,k=100,sp=0.05)+s(I(b.depth^.5)) +s(salinity)+s(temp.20m)
           +offset(log.net.area),data=mack,family=quasipoisson,method="REML")
```

Model selection

- A large part of what would usually be thought of as model selection is performed by smoothing parameter estimation, but smoothing selection does not usually remove terms altogether.
- There are three common approaches to deciding what terms to include.
  1. Get smoothing parameter estimation to do all the work, by adding a penalty for the un-penalized space of each term.
  2. Compute approximate p-values for testing terms for equality to zero, and use conventional selection strategies (backwards, forwards, backwards-forwards, etc).
  3. Use similar strategies based on AIC, or on the GCV or ML scores for the model.

Penalizing the penalty null space

- The penalty for a term is of the form $\boldsymbol{\beta}^{\mathrm{T}}\mathbf{S}\boldsymbol{\beta}$.
- Usually $\mathbf{S}$ is not full rank so some finite (M) dimensional space of functions is un-penalized.
- In consequence penalization can not completely remove the term from the model.
- Consider eigen-decomposition $\mathbf{S} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{\mathrm{T}}$. The last M eigenvalues will be zero. Let $\tilde{\mathbf{U}}$ denote their corresponding eigenvectors.
- $\boldsymbol{\beta}^{\mathrm{T}}\tilde{\mathbf{U}}\tilde{\mathbf{U}}^{\mathrm{T}}\boldsymbol{\beta}$ can be used as an extra penalty on just the component of the term that is unpenalized by $\boldsymbol{\beta}^{\mathrm{T}}\mathbf{S}\boldsymbol{\beta}$.
- Adding such a penalty to all the smooth terms in the model allows smoothing parameter selection to remove terms from the model altogether.

Null space penalization in action

```
> gm <- gam(egg.count~s(lon,lat,k=100)+s(I(b.depth^.5))+
+           s(c.dist) + s(temp.surf)
+           +s(salinity)+s(temp.20m)+offset(log.net.area),
+           data=mack,family=quasipoisson,method="REML",select=TRUE)
> gm

Family: quasipoisson
Link function: log

Formula:
egg.count ~ s(lon, lat, k = 100) + s(I(b.depth^0.5)) + s(c.dist) +
    s(temp.surf) + s(salinity) + s(temp.20m) + offset(log.net.area)

Estimated degrees of freedom:
60.60  2.17  0.42  0.00  1.83  5.17  total = 71.19

REML score: 515.0758
```

- So `temp.surf` is penalized out, and `c.dist` nearly so!

p-values and all that

- A p-values for smooth term, f, with a finite dimensional un-penalized space can be computed by a rather involved inversion of the Bayesian intervals for a smooth, which give good frequentist performance.
- The test statistic is $\hat{\mathbf{f}}^{\mathrm{T}}\mathbf{V}_f^{\tau-}\hat{\mathbf{f}}$ where $\mathbf{V}_f^{\tau'-}$ is a generalized rank τ' pseudoinverse of the Bayesian covariance matrix for $\mathbf{f}$ the vector of f evaluated at the observed covariate values. τ' is a version of the effective degrees of freedom of $\hat{f}$, based on $2\mathbf{F} - \mathbf{FF}$ in place of $\mathbf{F}$.
- For random effects, and smooths with no un-penalized space, another approach is needed.
- In both cases the p-values are conditional on the smoothing parameter estimates (you have been warned!)
- Refitting the egg model without null space penalization and calling `summary(gm)` gives...

summary(gm)

```
Family: quasipoisson
Link function: log

...

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.9506     0.1237   23.85   <2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Approximate significance of smooth terms:
                     edf Ref.df     F  p-value
s(lon,lat)        61.280 73.602 3.094 5.28e-13 ***
s(I(b.depth^0.5))  2.593  3.164 3.154  0.02354 *
s(c.dist)          1.000  1.000 1.532  0.21688
s(temp.surf)       1.000  1.000 0.133  0.71597
s(salinity)        1.001  1.001 8.891  0.00313 **
s(temp.20m)        5.960  6.941 3.504  0.00136 **
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

R-sq.(adj) =  0.825   Deviance explained = 90.2%
REML score = 510.79  Scale est. = 4.4062    n = 330
```

generalized AIC etc

- An approximate AIC is often used for model selection:

$$-2l(\hat{\beta}) + 2\tau$$

  where $\hat{\beta}$ are the maximum *penalized* likelihood estimates and τ is the effective degrees of freedom of the whole model, and the UBRE (Mallows C_p) score used for smoothness selection for known scale parameter is directly proportional to this.
- AIC usually gives very similar results to selecting models on the basis of the GCV (or UBRE) score.
- The ML score can also be used in the same way, but not REML (because of the usual lack of comparability between models with different fixed effect structures).

GLRT via `anova`

- Approximate generalized likelihood ratio testing can also be performed, again based on the maximum penalized likelihood estimates and effective degrees of freedom, and again, conditional on the smoothing parameter estimates.
- The `anova` function in R can be used for this purpose.
- The approximation has limited justification. If the model terms can all be closely approximated by unpenalized terms, then the approximation is often reasonable, but note that random effects can not be approximated in this way, and the approximation breaks down in this case.
- Unless your smooths are really frequentist random effects, resampled from their prior/marginal with every replication of the data, then a GLRT (or AIC) based on the ML or REML score is a bad idea.

Additive versus Interaction

- $f_1(x) + f_2(z)$, $f_3(x, z)$ or $f_1(x) + f_2(z) + f_3(x, z)$?
- Conceptually $f_1(x) + f_2(z)$ appears nested in $f_3(x, z)$, but unless you choose the smoothing penalties *very* carefully it won't be.
- The `t2` tensor product construction in `mgcv` build smooths with penalties that do nest $f_1(x) + f_2(z)$ in $f_3(x, z)$ (basically following a reduced rank version of the SS-ANOVA approach of Gu and Wahba), but the price you pay is the need to use penalties that are somewhat un-intuitive. `pen.edf` and `gam.vcomp` are useful with such terms.
- A cruder approach simply identifies smooths that are nested in a model, and supplies just enough constraints to make them identifiable. `mgcv` does this.
- Nested terms estimates are often so highly correlated that single term p-values must be treated with suspicion.

Summary

- Model checking is just like for a GLM + *check that smoothing basis dimensions are not too small*.
- Concurvity is the generalization of co-linearity to worry about in interpretation.
- A variety of model selection tools are available, including full penalization, generalized AIC, term specific p-values and approximate GLRT tests.
- Tests/p-values are approximate and conditional on smoothing parameter estimates.
  1. When smoothing parameter estimators are highly correlated (see e.g.`sp.vcov`), single term p-values should be treated with caution.
  2. GLRT tests are a particularly crude approximation, and can fail completely when random effects are involved.
- GAMs are statistical models and there are reasonable statistical tools available to help in the process of model building, but if you want machine learning, GAMs are probably not the place to start.

Some more advanced topics

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

Posterior simulation

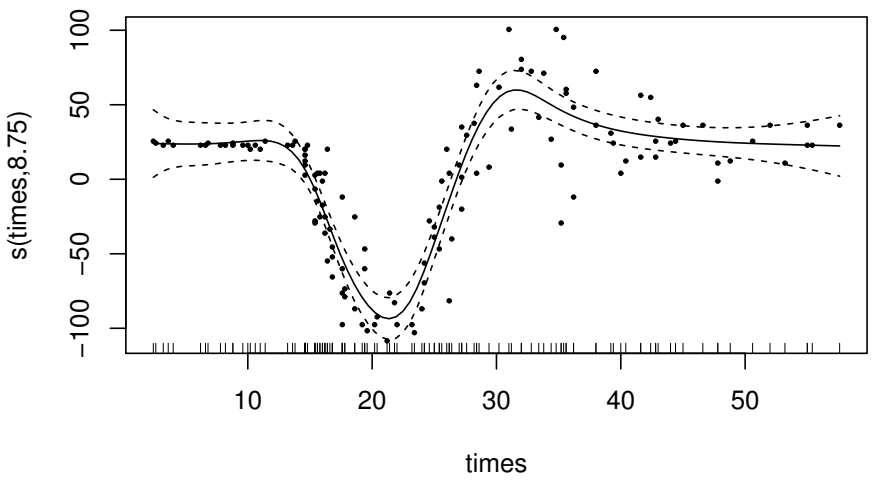
- Recall that for any fitted GAM we have the result

$$\boldsymbol{\beta}|\mathbf{y} \sim N(\hat{\boldsymbol{\beta}}, \mathbf{V}_\beta)$$

  (large sample approximation in the generalized case).
- This means that we can rapidly simulate from the posterior of any quantity derived from the fitted model.
- Such simulation is made much easier, if we can obtain the prediction matrix $\mathbf{X}_p$, mapping the model coefficents to the linear predictor, for any desired set of predictor variable values.
- `mgcv:predict.gam` computes such an $\mathbf{X}_p$ using `predict(...,type="lpmatrix")`

Posterior simulation example

- Here is an adaptive smooth fit to the motorcycle data.



- Suppose we would like a 95% CI for the trough to peak height.

Trough to peak CI

```
pd <- data.frame(times=seq(10,40,length=1000))
Xp <- predict(b,pd,type="lpmatrix") ## map coefs to fitted curves
beta <- coef(b);Vb <- vcov(b) ## posterior mean and cov of coefs
n <- 10000
br <- mvrnorm(n,beta,Vb) ## simulate n rep coef vectors from post.
a.range <- rep(NA,n)
for (i in 1:n) { ## loop to get trough to peak diff for each sim
  pred.a <- Xp%*%br[i,]  ## curve for this replicate
  a.range[i] <- max(pred.a)-min(pred.a) ## range for this curve
}
quantile(a.range,c(.025,.975))

    2.5%    97.5%
137.0796 174.5402
```

- This is very fast compared to boot-strapping, and less problematic.
- The for loop is only for clarity, it can be eliminated.

Correlated data

- Correlated data can be modelled using high rank Gaussian random fields (smoothers), or by GEE type assumption of a covariance structure for the response.
- For Gaussian data it is straightforward to incorporate a known correlation structure into the likelihood. If such a structure is sparse (it's Choleski factor, or inverse Choleski factor is sparse) then efficient computation is sometimes possible.
- An AR1 model is an example of such a sparse structure.
- Unknown correlation parameters can be optimized numerically, or by simple profile likelihood grid search.
- Software for correlated data is a bit limited at the moment (but if you don't have too many smooth terms, check out INLA).

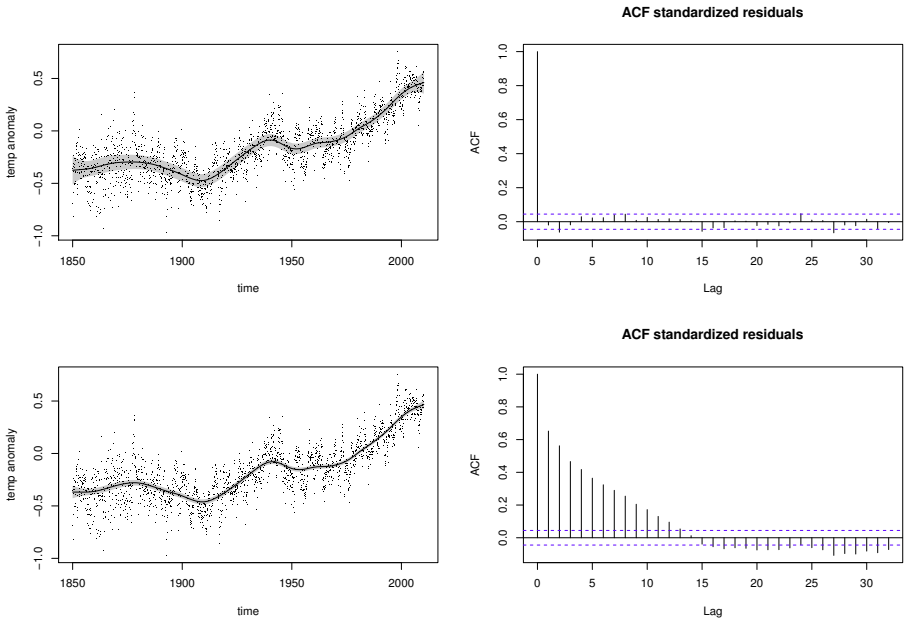
GAM + AR1 example

- The Hadley Centre (UK) assembles monthly global mean temperature datasets, going back to 1850 (e.g. hadcrut3 from their web site).
- The data appear quite noisy, so it is important to be able to say, objectively, what the underlying smooth trend in the data looks like.
- There is an annual cycle in the data, essentially because the Northern and Southern Hemispheres respond differently to incoming solar radiation.
- A reasonable model, of temperature anomaly, a_i, is

$$a_i = f(t_i) + g(m_i) + e_i$$

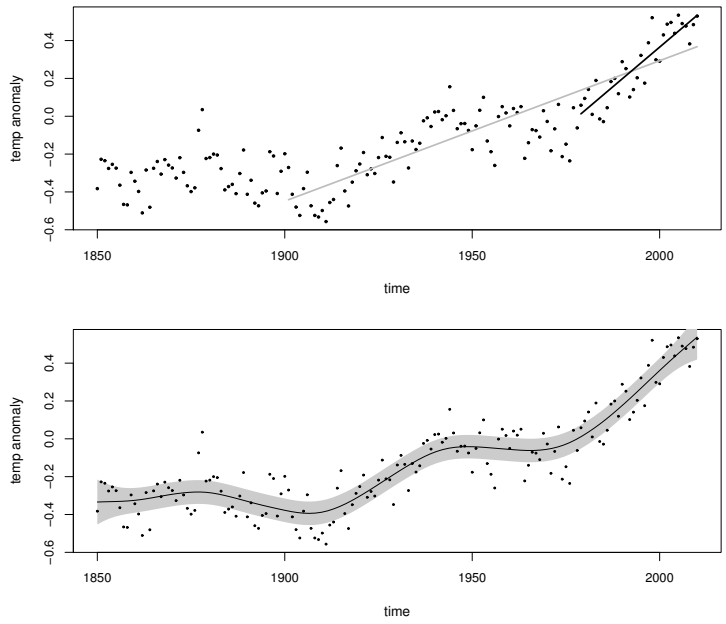
where the e_i are AR1 gaussian errors, with unknown correlation parameter. $g(m_i)$ is cyclic function of month.

`bam(...,rho=.98)` fit of AR1 GAM



- Upper is fit with REML optimal correlation parameter, lower is equivalent model without auto-correlation, but forced to have same smoothing parameters.

Annual data



- Top is what is currently done for IPCC presentation to policy makers.

Spatial correlation

- Sometimes a relatively high rank smooth suffices (e.g. a thin plate spline of space).
- Sometimes `bam` can be more efficient than `gam` for such high rank terms, but much over rank 1000 and the methods become impractically slow.
- A GEE type approach to correlation can be used with `gamm` via `nlme` type correlation structures, but convergence is not very reliable.
- Essentially the approach assumes a parameterized correlation structure for the working data used at each PQL iteration during fitting (of course this is just a likelihood method if the response is Gaussian).

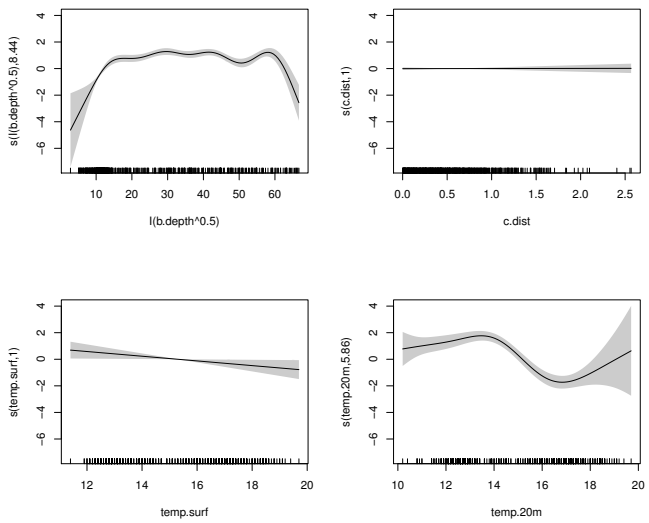
Spatial correlation example

- Revisiting the fish egg data, from earlier, we could try to force more of the explanatory power onto the covariates by replacing the spatial smooth with an assumption about residual spatial autocorrelation. Let's assume a simple model in which correlation decays as a half Gaussian...

```
mack$lon <- mack$lon + (runif(n)-.5)/20 ## jitter location

gmm <- gamm(egg.count ~ s(I(b.depth^.5)) + s(c.dist) +
            s(temp.surf) + s(temp.20m)+offset(log.net.area),
            data=mack,family=quasipoisson,
            correlation=corGaus(.1,form=~lon+lat))
```

- See `nlme` documentation for more on the correlation structure.
- Fitting takes 10s of minutes...

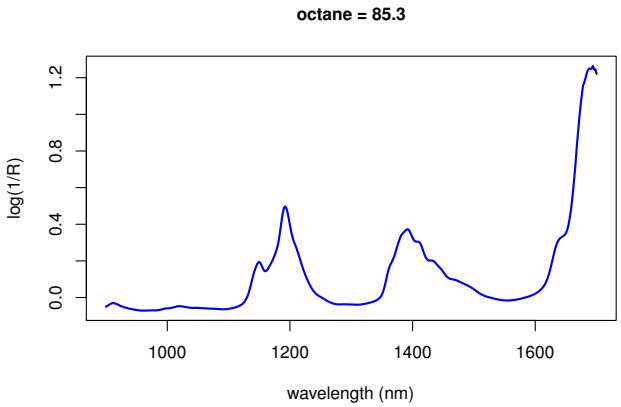
CorGaus GAM effects



- The sea bed depth effect is much stronger in this model.
- Spatial correlation in these models is an active area of research.

Functional data

- Function on scalar, and scalar on function regressions can readily be cast as GAMs/ penalized GLMs.
- Start with scalar on function and consider predicting octane rating from near infrared spectrum of gasoline.



scalar on function

- There are 60 such spectrum ($k_i(x)$) - octane (y_i) pairs (x is wavelength), and a model might be

$$y_i = \alpha + \int f(x)k_i(x)dx + \epsilon_i \simeq \alpha + \frac{1}{h}\sum_{k=1}^{p} k_i(x_k)f(x_k) + \epsilon_i$$

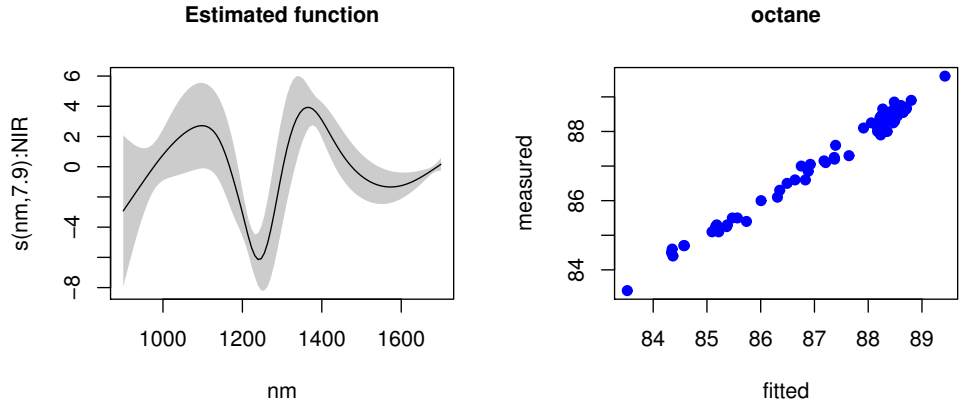
  where $f(x)$ is a smooth function of wavelength, and the x_k are evenly spaced h apart.
- Let $X_{ik} = x_k \; \forall \; i$ and $L_{ik} = k_i(x_k)/h$. In `mgcv:gam`

  ```
  s(X,by=L)
  ```

  evaluates $\sum_k f(X_{ik})L_{ik} = \frac{1}{h}\sum_{k=1}^{p} k_i(x_k)f(x_k)$, by invoking a summation convention for matrix arguments of smooths (including `te/2`).

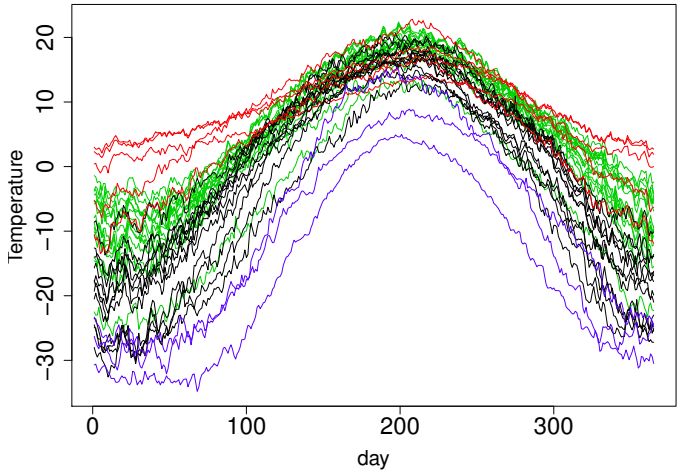
Octane fit

```
library(pls);data(gasoline);gas <- gasoline
nm <- seq(900,1700,by=2) ## create wavelength matrix...
gas$nm <- t(matrix(nm,length(nm),length(gas$octane)))
b <- gam(octane~s(nm,by=NIR,bs="ad"),data=gas)
plot(b,rug=FALSE,shade=TRUE,main="Estimated function")
plot(fitted(b),gas$octane,...)
```



function-on-scalar

- Annual temperature data from some Canadian locations.

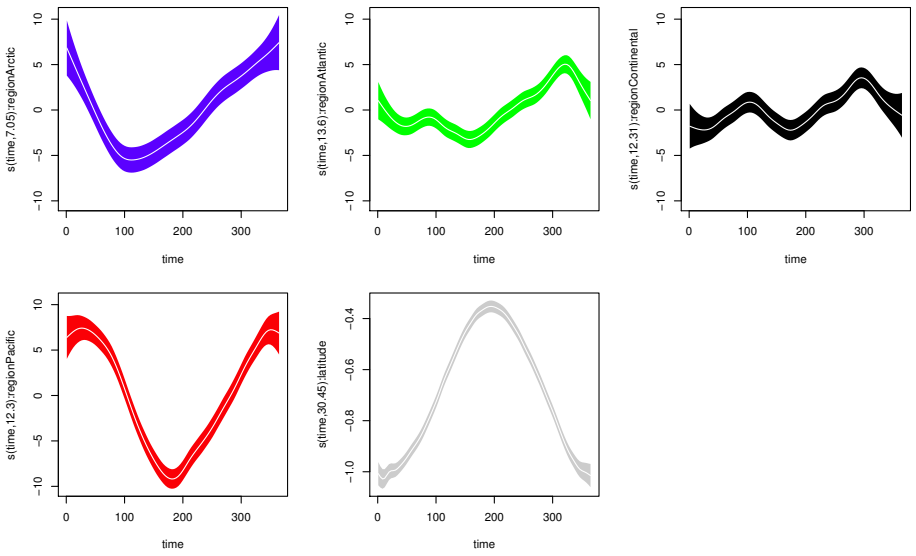


- Colour denotes region: blue is Arctic, black continental, red Pacific, green Atlantic.
- Model: if profile from region j:

$$\texttt{temp}_i = f_j(\texttt{t}_i) + f(\texttt{t}_i)\texttt{latitude}_i + \epsilon(\texttt{t}_i)$$

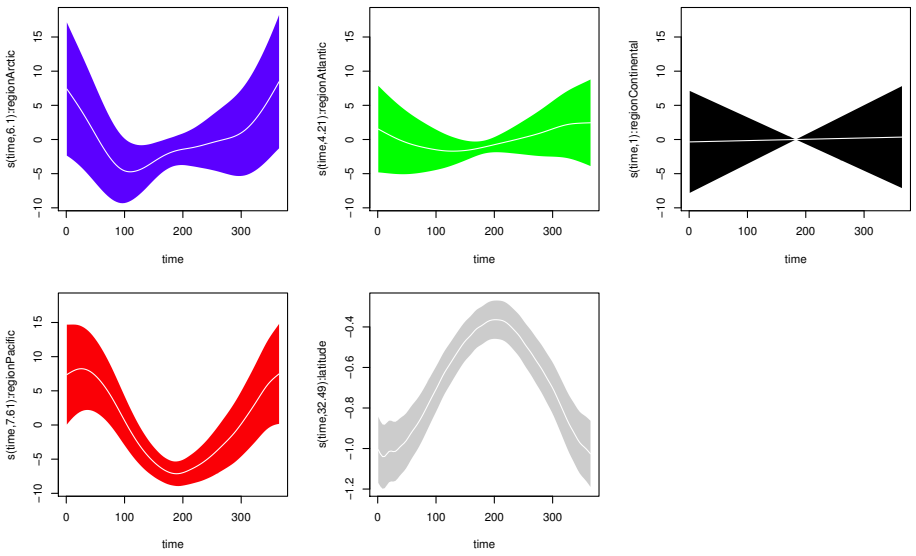
i.i.d error fit

```
b <- gam(T~region+s(time,k=20,bs="cr",by=region)+
         s(time,k=40,bs="cr",by=latitude),
         data=dat,method="REML")
```



AR1 error fit

```
b1 <- gamm(T~region+s(time,k=20,bs="cr",by=region)+
           s(time,k=40,bs="cr",by=latitude),
           data=dat,correlation=corAR1(form=~1|place))
```



The end.