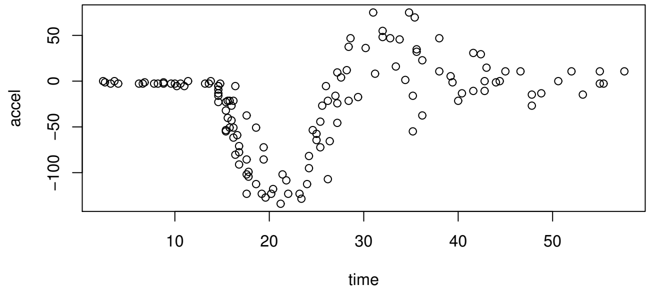
Basis Penalty Smoothers

Simon Wood
School of Mathematics, University of Bristol, U.K.

Estimating functions

- Here are some ancient data…



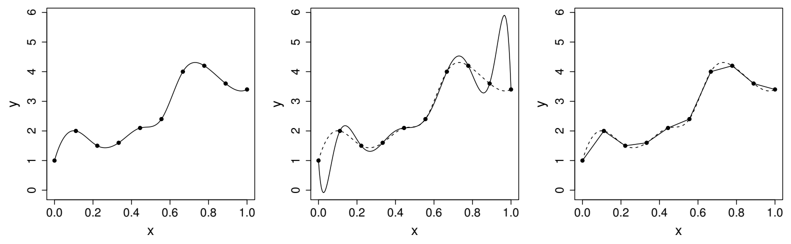
- If f is ‘a smooth function’, a suitable model might be

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

- 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

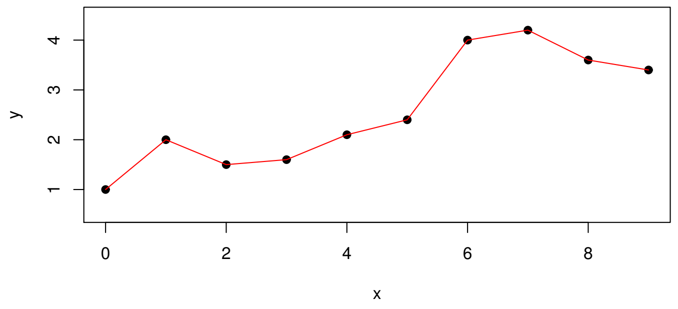
- Taylor’s theorem might suggest using the space of polynomials, but look at the middle panel’s attempt to approximate the function on the left with a polynomial.



- 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

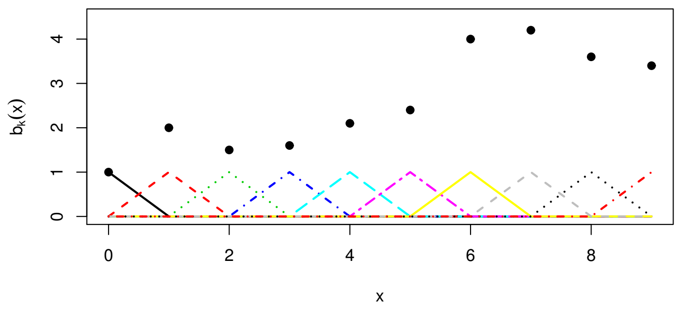
- So, for now, let’s represent f as a piecewise linear function, with derivative discontinuities at *knots*, x_k^*.



- …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

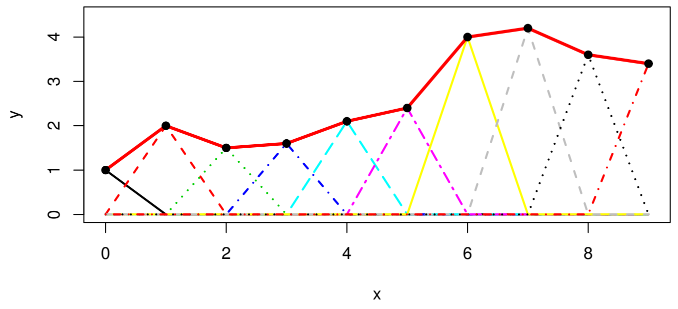
- The k^{th} tent function is 1 at x_k^* and descends linearly to zero at $x_{k\pm1}^*$. Elsewhere it is zero.
- The full set look like this...



- 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

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

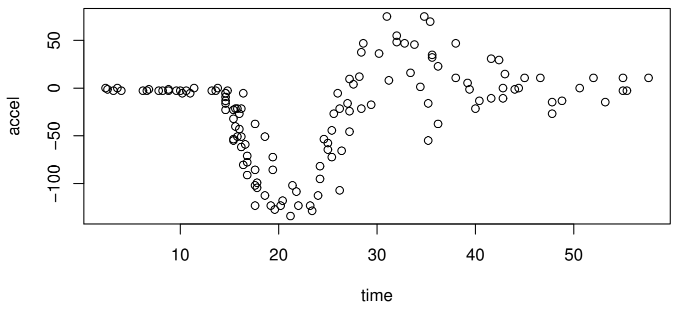
where the *prediction matrix* is given by

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

(1 row per x_i value, one column per x_k^* value.)

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

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

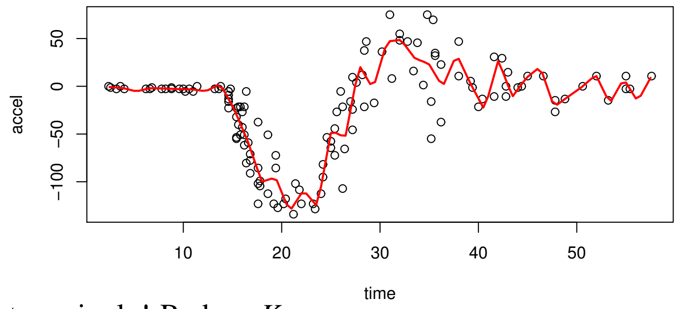
...a straightforward linear model, estimable via least squares.

Estimation in R

- Two lines of R code are enough to produce **X**. Then `lm` can be used to fit the model.

```
tk <- seq(min(t),max(t),length=k) ## knots
X<-apply(diag(k),1,function(y) approx(tk,y,t,rule=2)$y)
b <- lm(a ~ X-1)
```

- Here is the result[1] using K=40 evenly spaced t_k^* (knots).

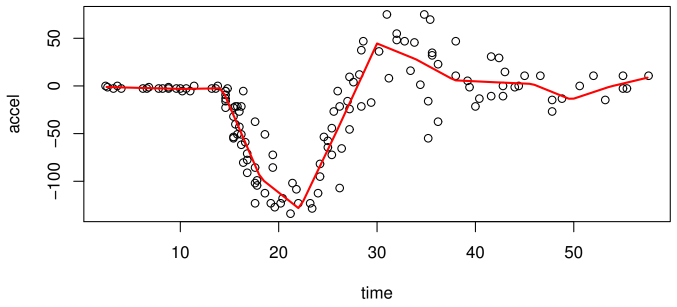


- Far too wiggly! Reduce K

[1]using `plot(t,a); lines(t,X %*% coef(b),col=2)`

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.

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

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 **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 **S**.

- In R. . .

```
D <- diff(diag(k),d=2)
S <- crossprod(D) ## or S <- t(D) %*%  D
```

Penalized fitting

- Now the penalized least squares estimates are

$$\hat{\beta} = \underset{\beta}{\text{argmin}} \sum_i \{a_i - f(t_i)\}^2 + \lambda \mathcal{P}(f)$$

smoothing parameter λ controls the fit-wiggliness tradeoff.

- For computational purposes this is re-written

$$\hat{\beta} = \underset{\beta}{\text{argmin}} \|\mathbf{a} - \mathbf{X}\beta\|^2 + \lambda \beta^\mathsf{T} \mathbf{S} \beta.$$

- Formally,

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

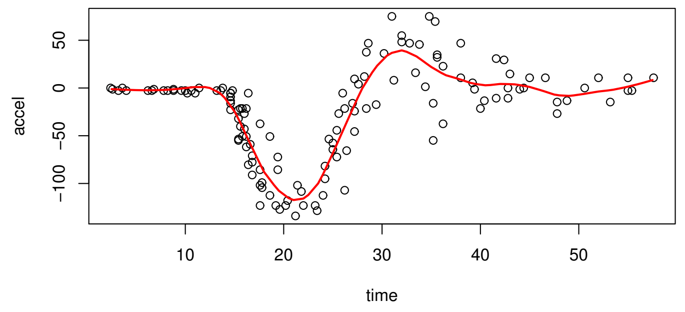
but direct use of this expression has sub-optimal computational stability.

Computing the smooth fit

- In fact

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

- 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 (Demmler-Reinsch) 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

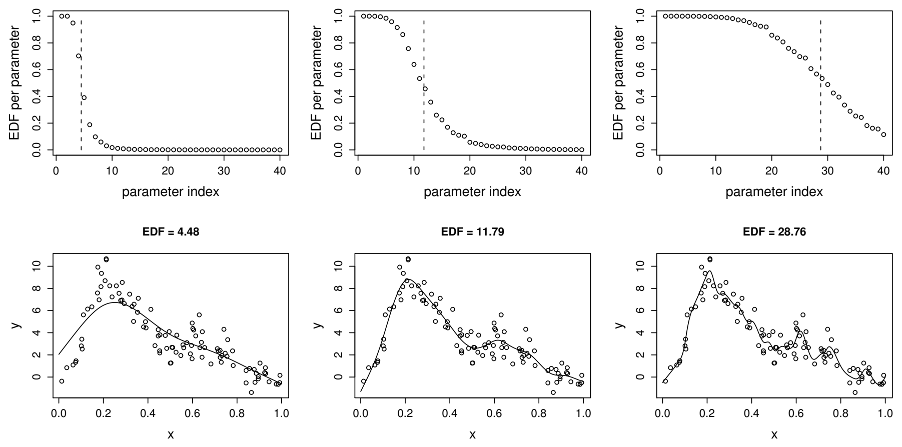
$$\mathbf{R}^{-\mathsf{T}}\mathbf{S}\mathbf{R}^{-1} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\mathsf{T}$$

- Define $\mathbf{P} = \mathbf{U}^\mathsf{T}\mathbf{R}$. And reparameterize $\beta' = \mathbf{P}\beta$.
- 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 j^{th} coefficient, and is bounded between 0 and 1. It gives the EDF for $\hat{\beta}_j$.
- So total EDF is $\sum_j (1 + \lambda\Lambda_{jj})^{-1} = \text{tr}(\mathbf{F})$, 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 (but lower MSE, hopefully).
- 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 $\hat{\sigma}^2 = \|\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}\|^2/\{n - \text{tr}(\mathbf{F})\}$ is useful.

Consequences of the Bayesian model

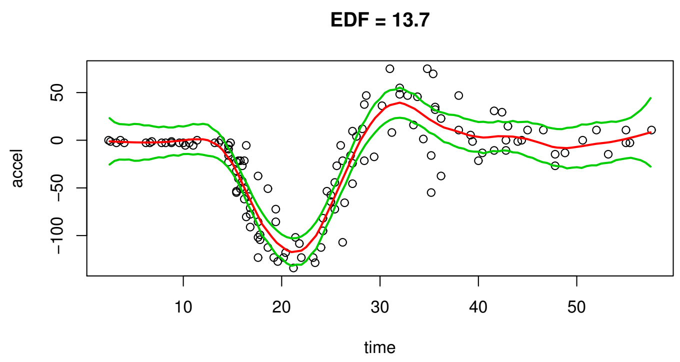
- $\boldsymbol{\beta} \sim N(\mathbf{0}, \mathbf{S}^{-}\sigma^2/\lambda) \Rightarrow \mathbf{f} \sim N(\mathbf{0}, (\mathbf{X}\mathbf{S}\mathbf{X}^{\mathsf{T}})^{-}\sigma^2/\lambda)$, i.e. f is equivalent to a Gaussian random field with covariance matrix $(\mathbf{X}\mathbf{S}\mathbf{X}^{\mathsf{T}})^{-}\sigma^2/\lambda$.
- The Bayesian model has the same structure as a linear mixed model, and can be computed as such (Empirical Bayes).
- 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.

Computing as a standard mixed model

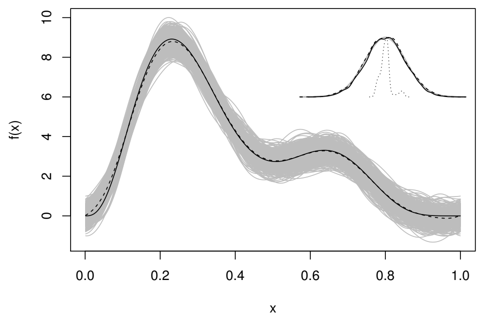
- Recall $\mathbf{D}^{\mathsf{T}}\mathbf{D} = \mathbf{S}$, and set $\boldsymbol{\beta}' = \mathbf{D}_+\boldsymbol{\beta}$ where
$$\mathbf{D}_+ = \begin{bmatrix} \mathbf{I}_2 & \mathbf{0} \\ & \mathbf{D} \end{bmatrix} \text{ (assume order 2 penalty here).}$$
- Then $\mathbf{X}\boldsymbol{\beta} = \mathbf{X}\mathbf{D}_+^{-1}\boldsymbol{\beta}'$ and $\boldsymbol{\beta}^{\mathsf{T}}\mathbf{S}\boldsymbol{\beta} = \sum_{i=3}^{K} \beta_i'^2$.
- Let $\mathbf{X}^*$ denote the first 2 cols of $\mathbf{X}\mathbf{D}_+^{-1}$ and $\mathbf{Z}$ the remainder.
- Let $(\boldsymbol{\beta}^{*T}, \mathbf{b}^{\mathsf{T}}) = \boldsymbol{\beta}'^{T}$. The Bayesian smooth model becomes
$$\mathbf{y} = \mathbf{X}^*\boldsymbol{\beta}^* + \mathbf{Z}\mathbf{b} + \boldsymbol{\epsilon}, \quad \mathbf{b} \sim N(\mathbf{0}, \mathbf{I}\sigma_b^2), \quad \boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{I}\sigma^2).$$
which is a standard linear mixed model.
- Notice that the columns of $\mathbf{X}^*$ form a basis for functions in the null space of the penalty.

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



Nychka's construction

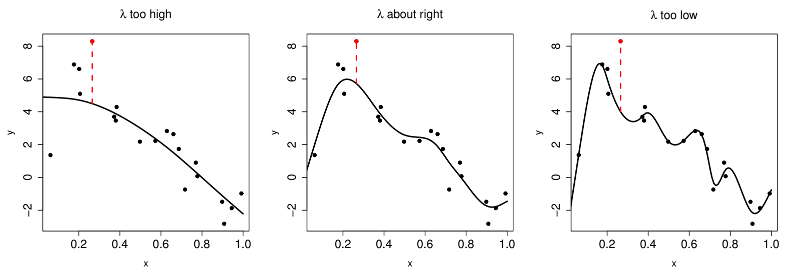


- Figure shows 500 spline estimates (grey) and their mean (dashed) of the truth shown in black.
- Top left shows distribution of sampling error (black) and bias (dotted) if we evaluate at a random x.
- Mean + bias is approximately normal (dashed + grey), so we can construct an interval with accross the function coverage.
- Bayes covariance matrix contains sampling variance + expected squared bias under prior.

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 $\hat{\boldsymbol{\beta}} = \text{argmin}_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \boldsymbol{\beta}^\mathsf{T}\mathbf{S}\boldsymbol{\beta}$ 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 (e.g. as linear mixed model, as above).
  3. Go fully Bayesian by completing the Bayesian model with a prior on λ (requires simulation and not pursued here).

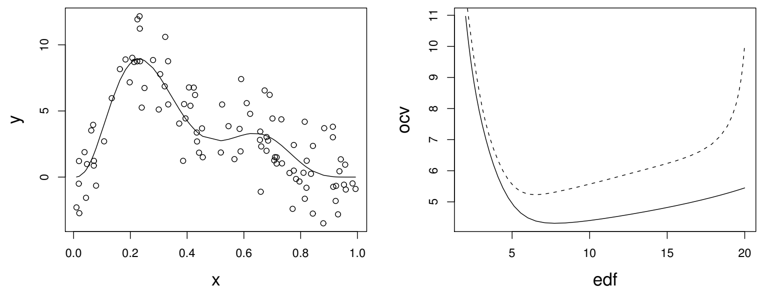
Prediction error: cross validation



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

$$\|\mathbf{Q}\mathbf{y} - \mathbf{Q}\mathbf{X}\boldsymbol{\beta}\|^2 + \lambda\boldsymbol{\beta}^\mathsf{T}\mathbf{S}\boldsymbol{\beta}$$

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

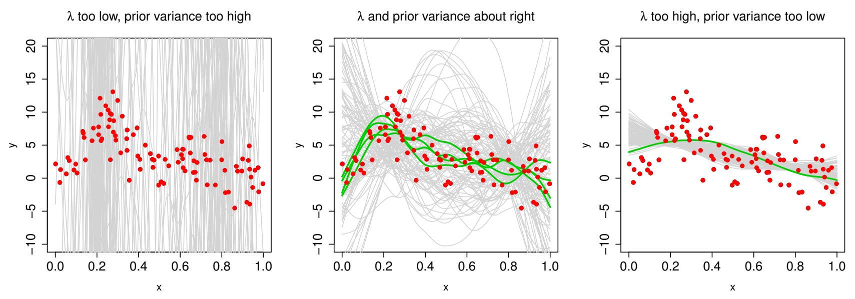
GCV: generalized cross validation

- 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:

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

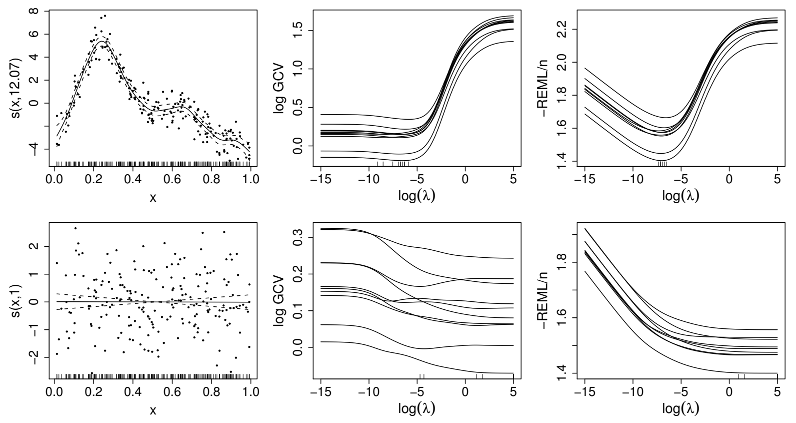
- It is easy to show that $\text{tr}(\mathbf{A}) = \text{tr}(\mathbf{F})$, where $\mathbf{F}$ is the degrees of freedom matrix.
- In addition to invariance, GCV is much easier to optimize efficiently in the multiple smoothing parameter case.

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.
- The methods presented here are not restricted to the piecewise linear smoother, we can subsitute better bases and penalties…

More basis penalty smoothers

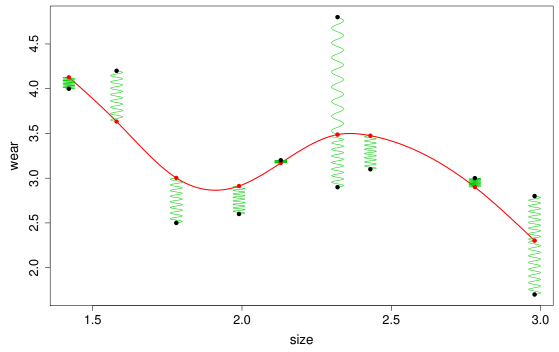
Simon Wood
School of Mathematics, University of Bristol, U.K.

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.
- The smoothers covered are available in R package `mgcv`. I'll give pointers as we go.

Splines

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

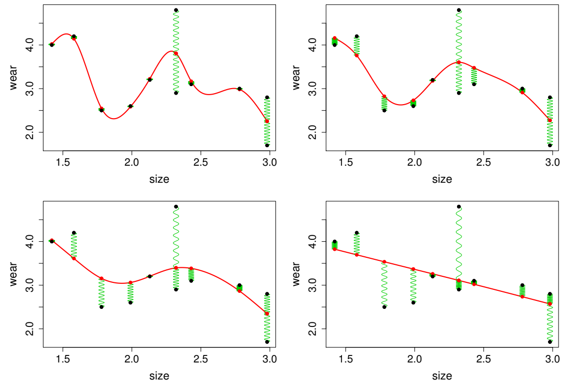


- Mathematically the red curve is the *function* minimizing

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

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.

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

- 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

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

  (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.

General spline theory: some background

- Consider a Hilbert space of real valued functions, f, on some domain τ (e.g. $[0,1]$).
- It is a *reproducing kernel Hilbert space*, $\mathcal{H}$, if evaluation is bounded. i.e. $\exists M$ s.t. $|f(t)| \le M\|f\|$.
- Then the Riesz representation thm says that there is a function $R_t \in \mathcal{H}$ s.t. $f(t) = \langle R_t, f\rangle$.
- Now consider $R_t(u)$ as a function of t: $R(t,u)$

$$\langle R_t, R_s\rangle = R(t,s)$$

  — so $R(t,s)$ is known as *reproducing kernel* of $\mathcal{H}$.
- Actually, to every positive definite function $R(t,s)$ corresponds a unique r.k.h.s.

Smoothing

- RKHS are quite useful for constructing smooth models, to see why consider finding $\hat{f}$ to minimize

$$\sum_i \{y_i - f(t_i)\}^2 + \lambda \int f''(t)^2 dt.$$

- Let $\mathcal{H}$ have $\langle f, g\rangle = \int g''(t)f''(t)dt$.
- Let $\mathcal{H}_0$ denote the RKHS of functions for which $\int f''(t)^2 dt = 0$, with finite basis $\phi_1(t), \phi_2(t)$, say.
- Spline problem seeks $\hat{f} \in \mathcal{H}_0 \oplus \mathcal{H}$ to minimize

$$\sum_i \{y_i - f(t_i)\}^2 + \lambda\|Pf\|^2.$$

Smoothing solution

- $\hat{f}(t) = \sum_{i=1}^n c_i R_{t_i}(t) + \sum_{i=1}^2 d_i\phi_i(t)$. Why?
- Suppose minimizer were $\tilde{f} = \hat{f} + \eta$ where $\eta \in \mathcal{H}$ and $\eta \perp \hat{f}$:
  1. $\eta(t_i) = \langle R_{t_i}, \eta\rangle = 0$.
  2. $\|P\tilde{f}\|^2 = \|P\hat{f}\|^2 + \|\eta\|^2$ which is minimized when $\eta = 0$.
- ...obviously this argument is rather general.
- So if $E_{ij} = \langle R_{t_i}, R_{t_j}\rangle$ and $T_{ij} = \phi_j(t_i)$ then we seek $\hat{c}$ and $\hat{d}$ to minimize

$$\|y - Td - Ec\|_2^2 + \lambda c^{\mathrm{T}} Ec.$$

- RKHS approach is elegant and general, but at $O(n^3)$ cost.

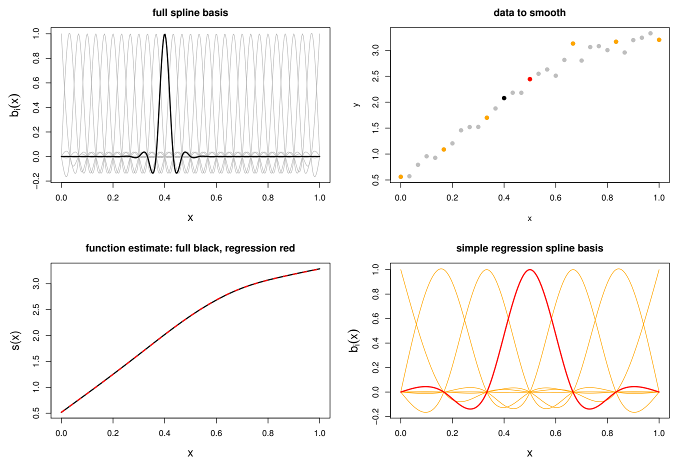
Penalized regression splines

- 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*.

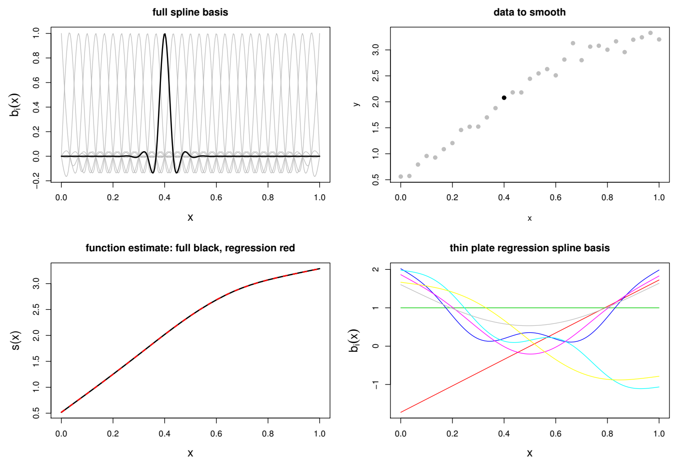
Knot based example: `"cr"`

- 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.

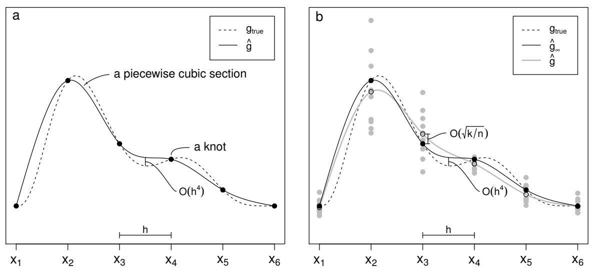


Eigen based example: `"tp"`

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



How small a basis: cubic spline example

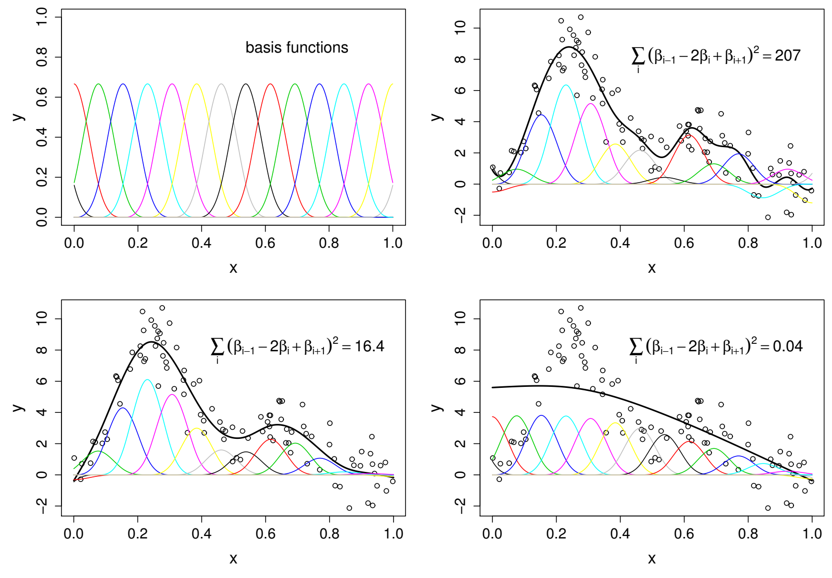


- A cubic interpolating spline $\hat{g}$ matching a function g_{true} at k evenly spaced (h) knots, has $O(h^4) = O(k^{-4})$ approx. error.
- If we observe g_{true} at each knot n/k times with noise (independently) then $\hat{g}$ has $O(\sqrt{k/n})$ sampling error.
- So $k = O(n^{1/9})$ gives optimal asymptotic error rate.
- With penalization use $k = O(n^{1/9}) - O(n^{1/5})$.

P-splines: "ps", "cp" & "bs"

- 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, and they provide sparse bases and penalties, which can be useful in big data and Bayesian settings.
- Actually, the same flexibility is available with derivative based penalties (see `?b.spline` in `mgcv`).

P-spline illustration



An adaptive smoother

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

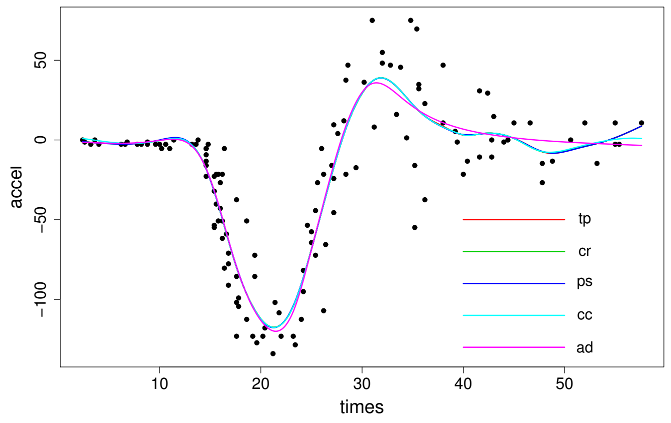
$$\mathcal{P}_a = \sum_{k=2}^{K-1} \omega_k(\beta_{k-1} - 2\beta_k + \beta_{k+1})^2 = \boldsymbol{\beta}^{\mathsf{T}}\mathbf{D}^{\mathsf{T}}\text{diag}(\boldsymbol{\omega})\mathbf{D}\boldsymbol{\beta}$$

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

- 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

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

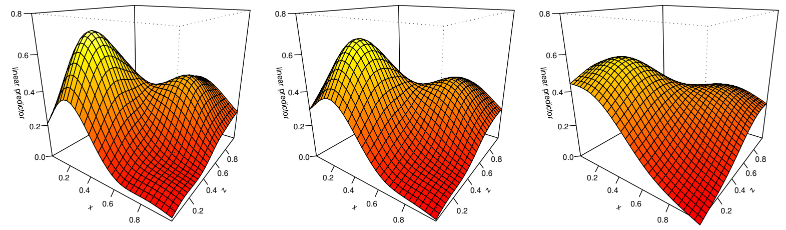
1D smooths compared



- 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

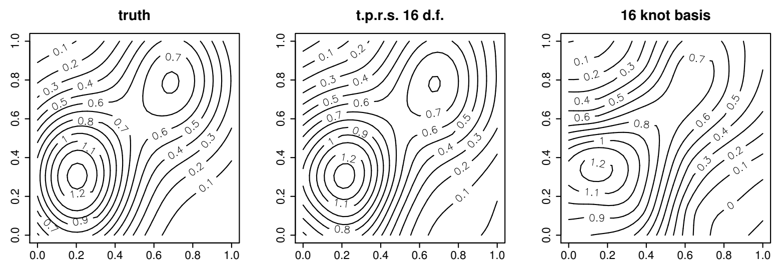
- Full thin plate splines have n parameters and $O(n^3)$ computational cost.
- This drops to $O(k^3)$ if we replace $\mathbf{E}$ by its rank k eigen approximation, $\mathbf{E}_k$, at cost $O(n^2k)$. Big saving if $k \ll n$
- Out of all rank k approximations this one minimizes

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

i.e. the approximation is somewhat optimal, and avoids choosing 'knot locations'.
- For large datasets, randomly subsample n_r data ($k \ll n_r \ll n$) and work out the truncated basis from the subsample, at $O(n_r^2 k)$ cost.

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")`

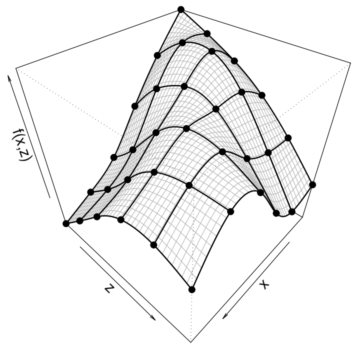
- The $m > d/2$ requirement causes thin plate splines in more than a few dimensions to be impractical, as the null space of the penalty rapidly becomes too high dimensional.
- But thin plate splines are only one special case of the splines introduced by Duchon (1977). He also considered penalties

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

  where $\mathfrak{F}$ denotes Fourier transform and $\boldsymbol{\tau}$ 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

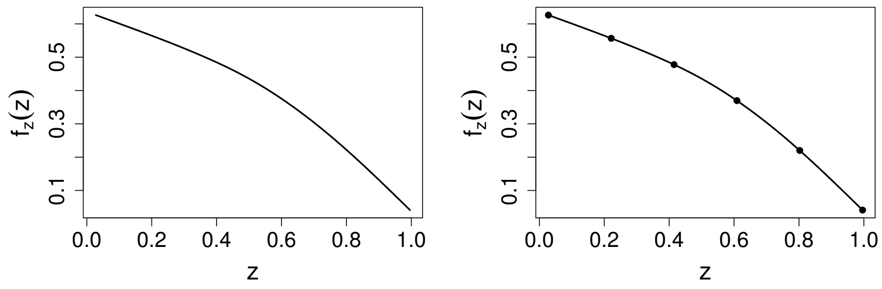
- Carefully constructed tensor product smooths are scale invariant.
- The following recipe for generating them is implemented by `te()` terms in `mgcv` (`t2()` is an alternative construction).
- Consider constructing a smooth of x, z.
- Start by choosing *marginal* bases and penalties, as if constructing 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

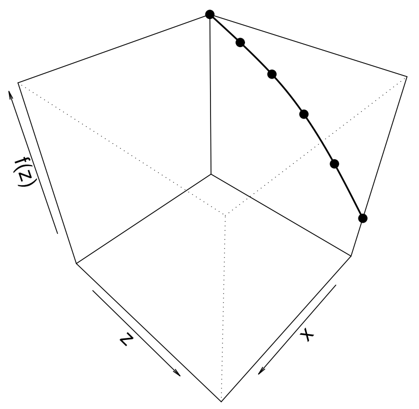
- Suppose we start with $f_z(z) = \sum_{i=1}^6 \beta_j b_j(z)$, on the left.

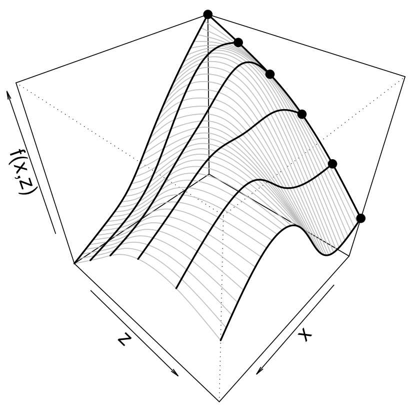


- We can always re-parameterize so that its coefficients are functions heights, at knots (right). Do same for f_x.

Making f_z depend on x

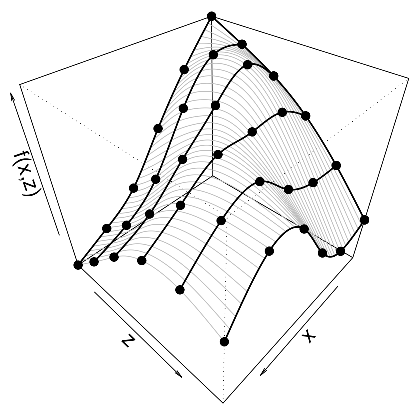
- Can make f_z a function of x by letting its coefficients vary smoothly with x

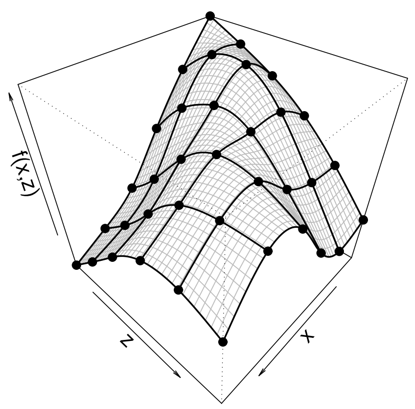




The complete tensor product smooth

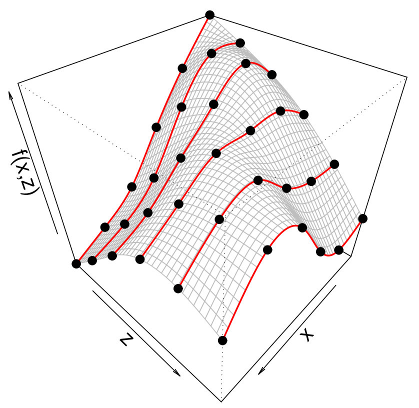
- Use f_x basis to let f_z coefficients vary smoothly (left).
- Construct is symmetric (see right).

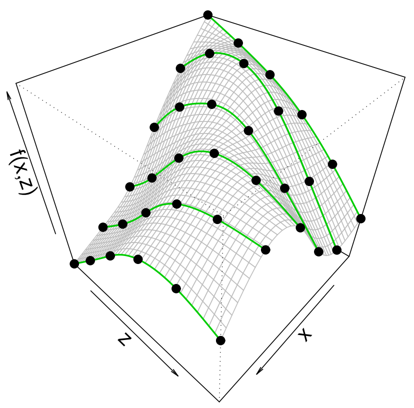




Tensor product penalties - one per dimension

- 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:

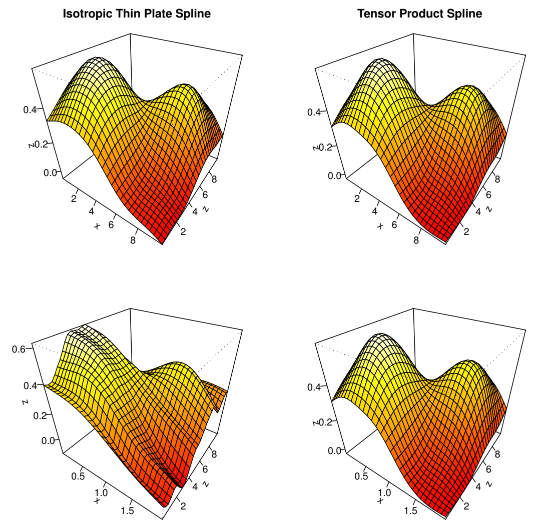
$$f(x,z) = \sum\sum \beta_{ij} b_j(z) a_i(x)$$

- With double penalties

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

- 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

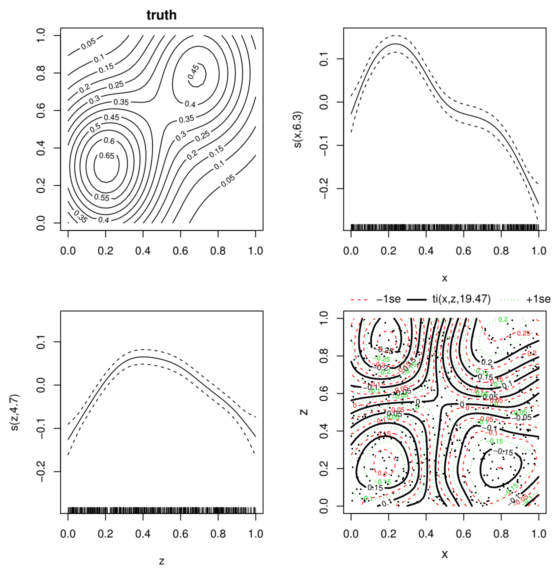


. . . each figure smooths the same data. The only modification is that x has been divided by 5 in the bottom row.

Functional ANOVA and `ti` terms

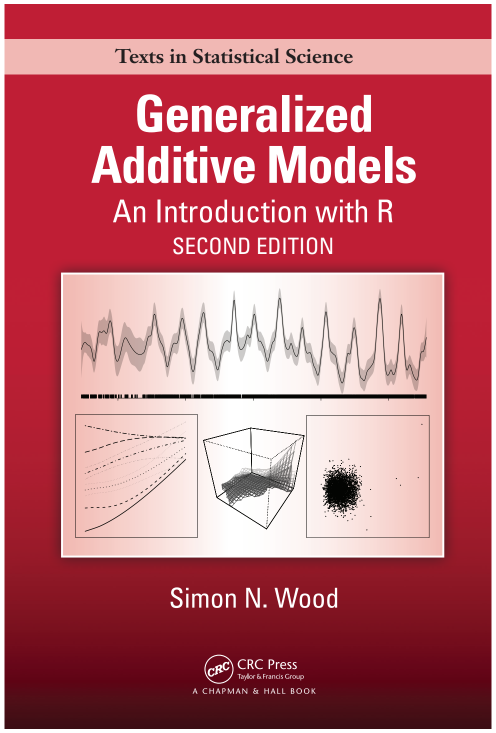
- The basis for a `te` tensor product smooth, $f(x,z)$, contains a subspace of functions of the form $f(x)+f(z)$ (similar applies in higher dimensions).
- This is because the marginal bases include the constant function in their span.
- Applying sum-to-zero constraints to the marginal bases *before* forming the tensor product smooth removes this $f(x)+f(z)$ component. See `ti()` terms in `mgcv`.
- Such a construction facilitates a 'mean effect plus interactions' smooth model.
- e.g. $f(x)+f(z)+f(x,z)$ can be fitted via `ti(x) + ti(z) + ti(x,z)`, in a stably interpretable manner. `t2` smooths are an alternative. See Chong Gu's book 'Smoothing Spline ANOVA'.

`ti` example



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` in `mgcv`).
- Any 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.



Discrete spatial smoothing: Markov random fields

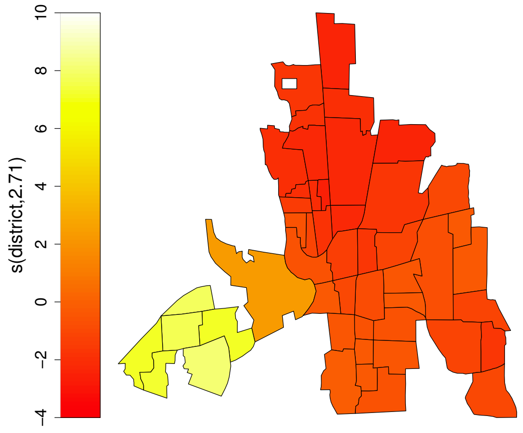
- Sometimes data come allocated to irregular partitions of space (e.g. administrative regions).
- Markov random fields area a popular way of smoothing such data.
- The smooth has a coefficient, γ_i, for each region.
- The neighbouring regions of each region are found, and a quadratic penalty constructed.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

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

- Eigen based rank reduction is effective here.

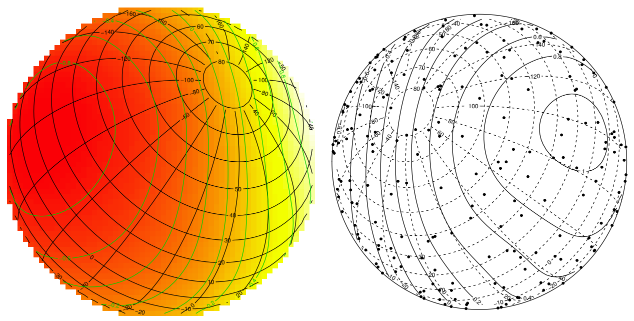
Markov random field illustration

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



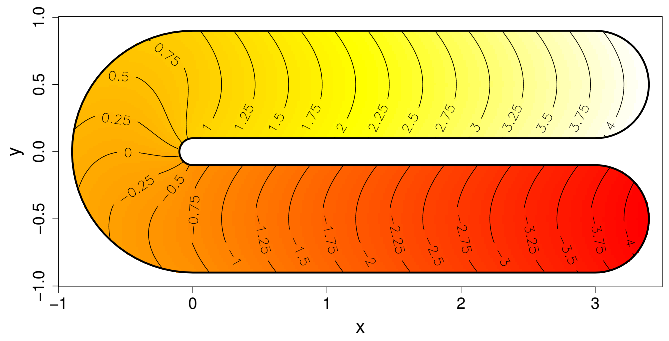
Smoothing on the globe

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



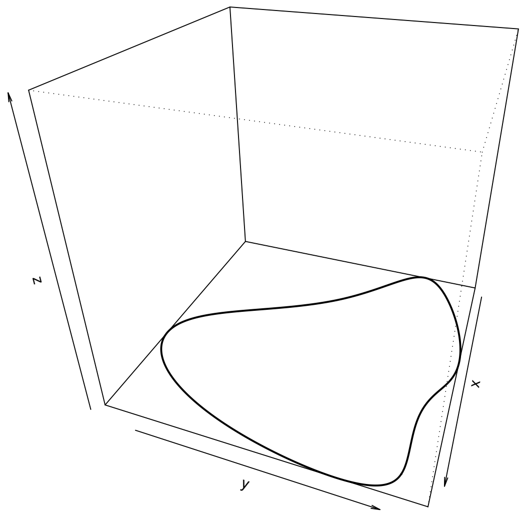
Finite area smoothing

- Suppose now want to smooth samples from this function

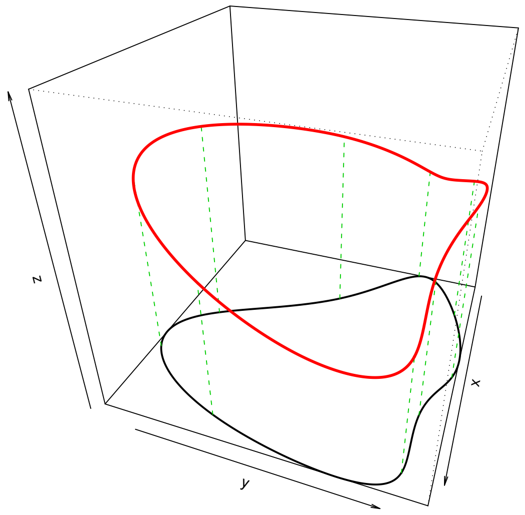


- . . . without ‘smoothing across’ the gap in the middle?
- Let’s use a soap film . . .

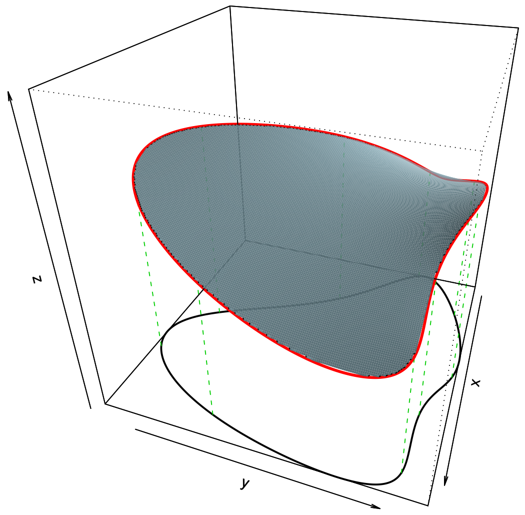
The domain



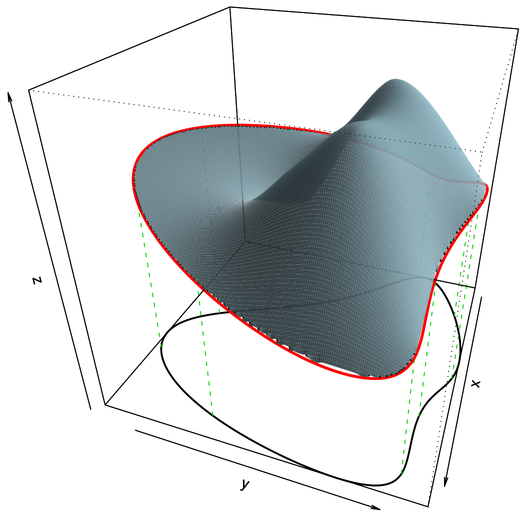
The boundary condition



The boundary interpolating film

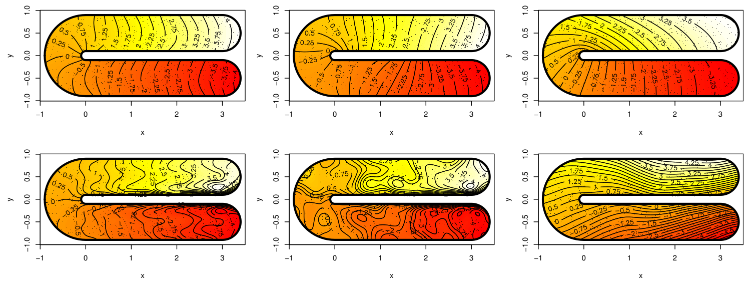


Distorted to approximate data



Soap film smoothers `s(...,bs="so")`

- Mathematically this smoother turns out to have a basis-penalty representation.
- 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, `ti`/`te`/`t2`, is appropriate. (`te/i/2` terms may be isotropic in some marginals.)
- 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.

Generalized Additive Models

Simon Wood
School of Mathematics, University of Bristol, 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 $\hat{\boldsymbol{\beta}} = \text{argmin}_\beta \|\mathbf{y} - \mathbf{X}\beta\|^2 + \lambda\beta^\mathsf{T}\mathbf{S}\beta$.
  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 and beyond...

Additive Models

- Consider the model

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

  - $\mathbf{A}_i$ 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

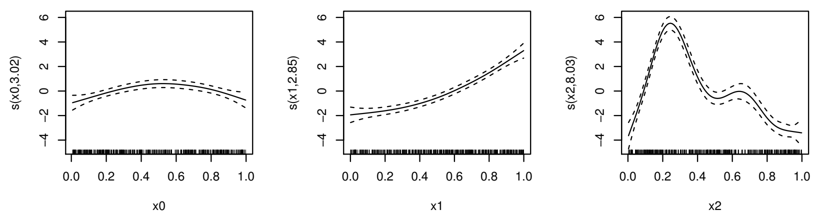
- Choose a basis and penalty for each f_j.
- Let the model matrix for f_j be $\mathbf{X}$ and let $\lambda\boldsymbol{\beta}^\mathsf{T}\mathbf{S}\boldsymbol{\beta}$ be the penalty (more generally $\sum_j \lambda_j\boldsymbol{\beta}^\mathsf{T}\mathbf{S}_j\boldsymbol{\beta}$).
- Reparameterize to absorb the constraint $\mathbf{1}^\mathsf{T}\mathbf{X}\beta = 0$. 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 $\hat{\boldsymbol{\beta}} = \text{argmin}_\beta \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \sum_j \lambda_j \boldsymbol{\beta}^\mathsf{T}\mathbf{S}_j\boldsymbol{\beta}$.
- Again $\boldsymbol{\lambda}$ can be estimated by GCV, REML etc.

Simple example

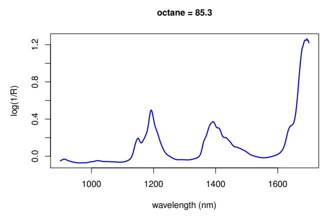
- Response observations y_i and predictors x_{ji}.
- $y_i = f_0(x_{0i}) + f_1(x_{1i}) + f_2(x_{2i}) + \epsilon_i$ where $\epsilon_i \sim N(0, \sigma^2)$.
- `b <- gam(y~s(x0)+s(x1)+s(x2),method="REML")`
  `plot(b)`



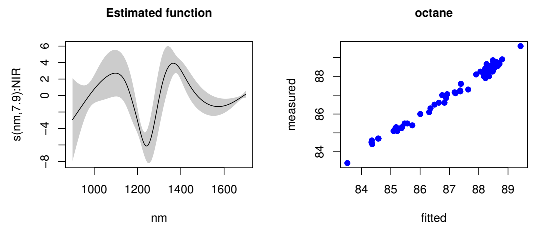
Linear functional generalization

- 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. $L_{ij}f_j = \int k_i(x)f_j(x)dx$, or $L_{ij}f_j = f(x_{ji})z_i$ (k_i and z_i known).
- Dropping j for now, we can discretize $L_if(x) \simeq \sum_k \tilde{L}_{ik}f(x_k)$.
- So $L_if(x) \simeq \sum_k \tilde{L}_{ik}\tilde{\mathbf{X}}_k\boldsymbol{\beta}$, 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 $L_if(x)$ is $\tilde{\mathbf{L}}\tilde{\mathbf{X}}$ (assuming same $x_k s$ for each i). The penalties are just those for f.
- Hence the extended model can be written in the same general form as the simple AM.

Example: scalar on function regression



- $y_i = \alpha + \int f(x)k_i(x)dx + \epsilon_i$
  $f(x)$ is a smooth of wavelength.
- `gam(y ~ s(X,by=K))`
  $X_{ij} = x_j$, $K_{ij} = k_i(x_j)/h$



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)...

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

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

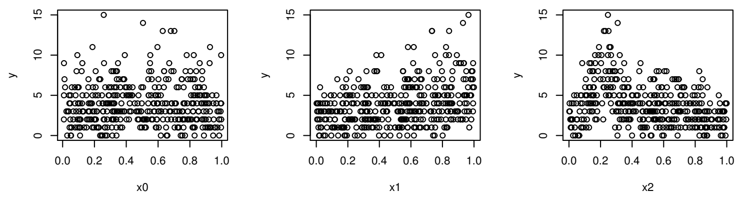
GAM computation: $\hat{\boldsymbol{\beta}}|\mathbf{y}$

- 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

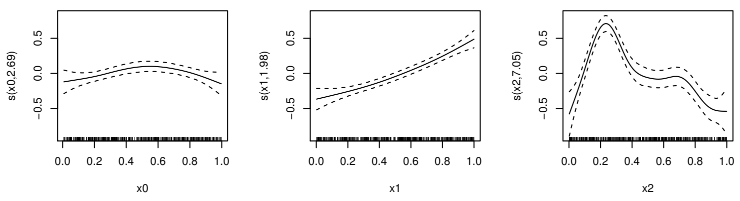
$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\text{argmin}} \sum_i w_i(z_i - \mathbf{X}_i\boldsymbol{\beta})^2 + \sum \lambda_j \boldsymbol{\beta}^{\mathsf{T}}\mathbf{S}_j\boldsymbol{\beta}$$

  and hence revised estimates $\hat{\boldsymbol{\eta}} = \mathbf{X}\hat{\boldsymbol{\beta}}$ and $\hat{\boldsymbol{\mu}} = g^{-1}(\hat{\boldsymbol{\eta}})$.
- $\alpha_i = 1 + (y_i - \hat{\mu}_i)(V_i'/V_i + g_i''/g_i')$ 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.

Example: Poisson regression



- $y_i \sim \text{Poi}(\mu_i)$ where $\log(\mu_i) = f_0(x_{0i}) + f_1(x_{1i}) + f_2(x_{2i})$.
- `gam(y~s(x0)+s(x1)+s(x2),family=poisson(link=log))`



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

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

- $\hat{\phi} = \sum_i w_i(z_i - \mathbf{X}_i\hat{\boldsymbol{\beta}})^2/\{n - \text{tr}(\mathbf{F})\}$ is a Pearson based ϕ estimate (but Fletcher, 2012 Biometrika, better)

$\boldsymbol{\lambda}$ 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

- GCV generalizes to
$$\mathcal{V}_g = nD(\hat{\boldsymbol{\beta}}_\lambda)/\{n - \text{tr}(\mathbf{F})\}^2$$
- Laplace approximate (negative twice) REML is
$$\mathcal{V}_r = \frac{D(\hat{\boldsymbol{\beta}}) + \hat{\boldsymbol{\beta}}^\mathsf{T}\mathbf{S}\hat{\boldsymbol{\beta}}}{\phi} - 2l_s(\phi) \\ + (\log|\mathbf{X}^\mathsf{T}\mathbf{W}\mathbf{X} + \mathbf{S}| - \log|\mathbf{S}|_+) - M_p \log(2\pi\phi),$$

  well founded if $\dim(\boldsymbol{\beta}) = O(n^\alpha)$, $\alpha \le 1/3$ (Shun and McCullagh, JRSSB, 1995), but this result is not sharp.

Nested iteration strategy (Wood, 2011, JRSSB)

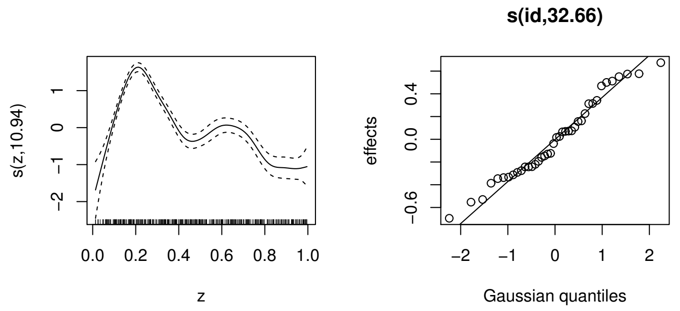
- Optimization wrt $\boldsymbol{\rho} = \log \boldsymbol{\lambda}$ is by Newton's method, using analytic derivatives.
- For each trial $\boldsymbol{\lambda}$ used by Newton's method...
  1. Re-parameterize for maximum numerical stability in computing $\hat{\boldsymbol{\beta}}$ and terms like $\log|\mathbf{S}|_+$.
  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.

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.

Simple GAMM example

- $y_i \sim \text{Poi}(\mu_i)$ where $\log(\mu_i) = f(z_i) + b_{\text{id}(i)}$ if observation i is from group id(i). $\mathbf{b} \sim N(\mathbf{0}, \mathbf{I}\sigma_b^2)$.
- `gam(y ~ s(z,k=20)+s(id,bs="re"),family=poisson, method="REML")`
-
```
gamm(y~s(z,k=20),family=poisson,
     random=list(id=~1))
```
- `gamm4(y~s(z,k=20),family=poisson,random=~(1|id)`



GAM functions in `mgcv`

- `gam`. Main modelling function. Can use all distributions listed in `?family.mgcv`. Tems like `s(...,bs="re")` in model implement simple random effects.
- `bam`. Big data version of `gam`. Exponential family distributions only. See `discrete` and `nthreads` options for bigger and faster still.
- `gamm`. Uses Bayesian duality of smooths and random effects to estimate generalized additive mixed models using `lme`. Random effects as `lme`. See package `gamm4` for `lme4` version.
- `jagam`. Writes JAGS (BUGS dialect) code to fit GAM using JAGS. Idea is to edit code to add more complicated random effect structures, different distributions etc.

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.
  - A common example is ‘spatial confounding’: the model contains a spatial smoother, but some of the covariates also vary smoothly over space.
  - Like co-linearity it is statistically destabilising and complicates interpretation, (in `mgcv` see `?concurvity`).

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

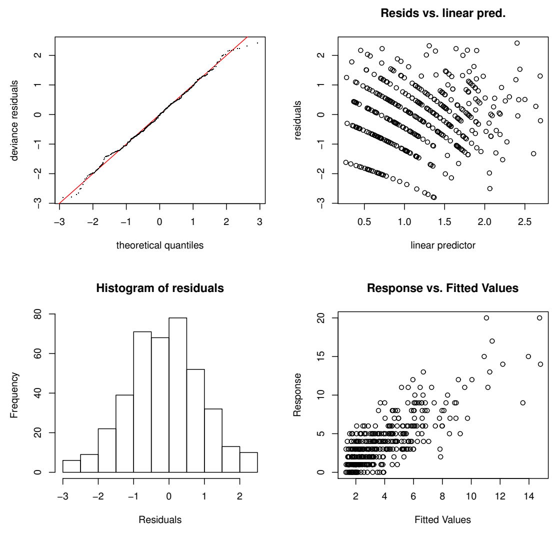
```
> b <- gam(y~s(x0)+s(x1,x2,k=40)+s(x3)+s(x4),
+ family=poisson,data=dat,method="REML")
>
> gam.check(b)

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]
```

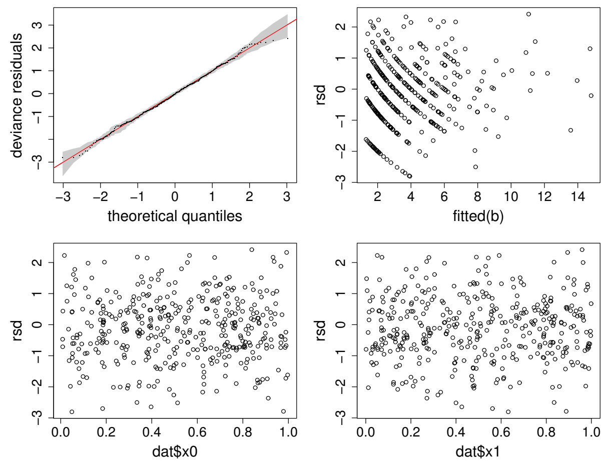
- The printed output is rather detailed information about smoothing parameter estimation convergence.
- 4 residual plots are produced, the first is from `qq.gam`, unless quasi-likelihood is used, in which case we have to fall back on a normal QQ-plot (but anyway don't care about this plot). The rest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

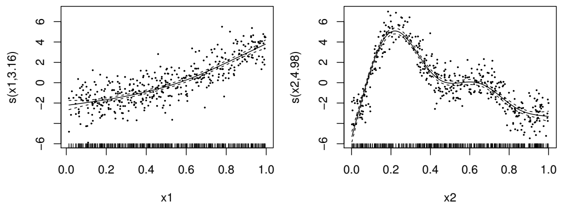


Basis dimension checking

- If the basis dimension for a smooth, $f(z)$, is too small then it will oversmooth, and leave pattern (autocorrelation) in the model residuals with respect z.
- This leads to several checking methods:
  1. Add the residuals to $\hat{f}(z)$, to obtain *partial residuals* and overlay these on a plot of $\hat{f}(z)$, to look for patterns.
  2. Order the residuals w.r.t. z and compute their mean (squared) first differences, $\bar{\Delta}$. Approximate the distribution of $\bar{\Delta}$ under the null hypothesis of zero auto-correlation by randomization of the residuals. Hence compute a p-value for this null. (Similar possible for multi-dimensional z).
  3. Smooth the deviance residuals w.r.t. z using a higher basis dimension (e.g. doubled): a non-flat smooth may be evidence for residual pattern.
- Checks are important if the EDF is close to the basis dimension.

Practical basis dimension check example

```
b <- gam(y ~ s(x1,k=6) + s(x2,k=6),method="REML")
plot(b,pages=1,residuals=TRUE)
```



```
> gam.check(b)
...
          k'   edf k-index p-value
s(x1) 5.000 3.160   0.948    0.23
s(x2) 5.000 4.983   0.827    0.00
> rsd <- residuals(b)
> gam(rsd~s(x2,k=12),method="REML")
...
Estimated degrees of freedom:
9.9  total = 10.9
```

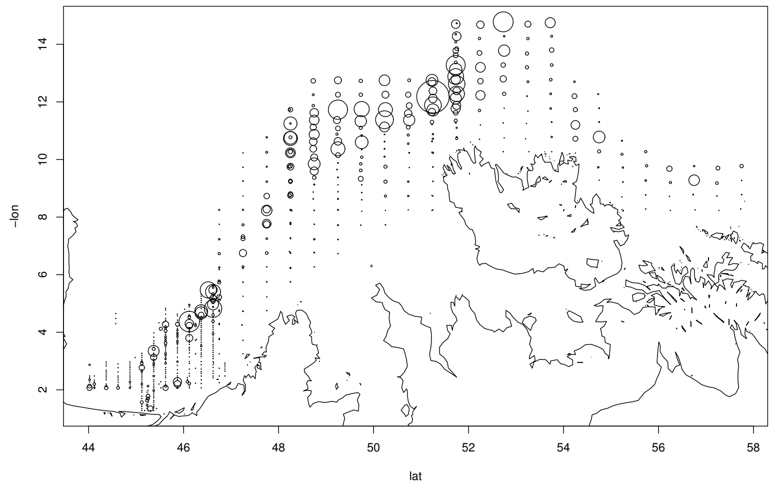
Model selection

- 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 $\boldsymbol{\beta}^{\mathsf{T}}\mathbf{S}\boldsymbol{\beta}$.
- 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 $\mathbf{S} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{\mathsf{T}}$. The last M eigenvalues will be zero. Let $\tilde{\mathbf{U}}$ denote their corresponding eigenvectors.
- $\boldsymbol{\beta}^{\mathsf{T}}\tilde{\mathbf{U}}\tilde{\mathbf{U}}^{\mathsf{T}}\boldsymbol{\beta}$ can be used as an extra penalty on just the component of the term that is unpenalized by $\boldsymbol{\beta}^{\mathsf{T}}\mathbf{S}\boldsymbol{\beta}$.
- Adding such a penalty to all the smooth terms in the model allows smoothing parameter selection to remove terms from the model altogether.

Fish egg modelling example



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 for smooth terms

- 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 $\hat{\mathbf{f}}^{\mathsf{T}}\mathbf{V}_f^{\tau-}\hat{\mathbf{f}}$ 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 $\mathbf{F}$.
- Under the null hypothesis that $\mathbf{f} = \mathbf{0}$ the statistic has a χ^2_τ distribution for integer τ, and a weighted sum of χ^2 distributions otherwise.
- See Wood (2013) Biometrika, 100, 221-228.

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 ' '

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
```

p-values for random effect terms

- The test for smooth terms does not work for random effects — the notion of effective degrees of freedom is much less meaningful, and the approximations break down as the effect tends to zero.
- Tests for random effects/variance components are difficult: the null hypothesis restricts the variance parameter to the edge of the parameter space.
- This problem can be addressed by:
  1. Fix all smoothing/variance parameters other than the parameter of interest at their estimated values.
  2. Re-express the log likelihood ratio statistic for the test in terms of the predicted random effect coefficients of the term.
  3. Under the null, this statistic is distributed as a weighted sum of χ^2 random variables.
- Details are in Wood (2013) Biometrika, 100, 1005-1010.
- The test is exact in the case of a single random Gaussian effect model, and approximate otherwise.

AIC

- There are two possibilities:
  1. *Marginal AIC*. Integrate the penalized terms/ random effects out of the likelihood. Base the AIC on the resulting marginal likelihood: the number of parameters is taken as the number of fixed effects, un-penalized coefficients, smoothing parameters and variance parameters.
  2. *Conditional AIC*. Base the AIC on the likelihood of the penalized and un-penalized model coefficients (including random effect coefficients), replacing the number of parameters with the effective degrees of freedom.
- There are problems with both.
  1. Marginal likelihood underestimates variance components (over-smooths), so the marginal AIC over-favours simple models (REML is not comparable between models with any difference in fixed effect/ un-penalized structure).
  2. Conditional AIC is much too likely to select the over-complicated model, by virtue of the neglect of smoothing/variance parameter uncertainty in the effective degrees of freedom (Greven and Kneib, 2010, Biometrika).

A better AIC

- Let $\boldsymbol{\beta}_d$ be the coefficients minimizing the KL-divergence, and $\boldsymbol{\mathcal{I}}_d$ the information at this point.
- Penalization does not alter AIC derivation up to

$$\text{AIC} = -2l(\hat{\boldsymbol{\beta}}) + 2\mathbb{E}\left\{(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}_d)^{\mathsf{T}}\boldsymbol{\mathcal{I}}_d(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}_d)\right\}$$

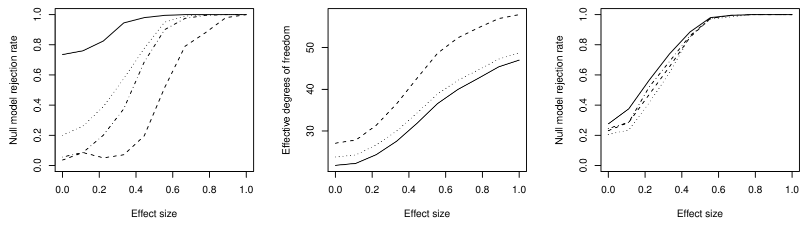
  …but it does alter the expectation term (variance ↓ bias ↑).
- Accounting for penalization we get

$$\text{AIC} = -2l(\hat{\beta}) + 2\text{tr}(\boldsymbol{\mathcal{I}}\mathbf{V}_\beta),$$

  where $\mathbf{V}_\beta$ is the Bayesian posterior covariance matrix.
- So far the trace term is just the EDF, but we can correct $\mathbf{V}_\beta$ for smoothing parameter uncertainty, to get a corrected EDF and AIC.
- Wood, Pya and Säfken (2016, JASA) give the full details.

Better AIC example

- How often AIC selects a term as the effect size of the term increases…



- Left, for a random effect, middle the different effective degrees of freedom for the left plot. Right for a smooth. Dotted is the new AIC computation.

Key help files in `mgcv`

- Best to navigate help using HTML via `help.start()`.
- `?mgcv.package` is an overview, with links to key information.
- Each modelling function has a help page.
- `?plot.gam`, `?predict.gam` `?summary.gam`, `?anova.gam`, `?AIC.gam` and `?gam.check` describe key functions for use with fitted model.
- `?family.mgcv` describes the distributions available.
- `?smooth.terms` describes the range of smoothers.
- `?gam.models` gives some information on specifying models.
- `?random.effects` is a mixed GAM overview.

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.
- Checking is as for a GLM, bit we should also check smoothing basis dimensions.
- Selection can be done using penalties or adaptations of familiar regression modelling techniques.

Other software: R packages and SAS

- `GAMPL` in SAS implements the framework described here.
- `gam` provides Hastie and Tibshirani's original GAM methods.
- `VGAM` and `gamlss` extend the GAM model class using the H& T computational approach.
- `semiPar` is based on Ruppert Wand and Carroll's 2003 book.
- `gss`, `assist` and `bigSplines` implement the smoothing spline ANOVA methods of Wahba, Gu et al.
- `R2BayesX` and `INLA` provide fully Bayesian approaches (INLA is especially good for high rank spatial fields).
- Natalya Pya's `scam` offers shape constrained additive models.
- `refund` provides FDA using `mgcv` as fitting engine.

Additive models beyond exponential families.

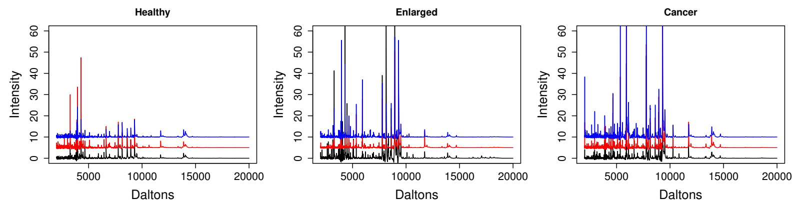
Simon Wood
Mathematical Sciences, University of Bristol, U.K.

Extending the model

- Many extensions have been proposed…
  1. GAM type linear predictor with non-exponential family distribution for independent response variable. Examples: scaled t for heavy tails, beta regression for proportions, ordered categorical models, Tweedie and negative binomial distributions
  2. GAM type linear predictor without a separable log-likelihood. E.g. Cox Proportional hazards and Cox Process.
  3. Models with multiple GAM type linear predictors: E.g. GAMLSS (Rigby & Stasinopoulus, 2005), or multivariate additive models (e.g. Yee and Wild, 1996).
- Let us consider producing a general framework for the extensions allowing us to do (almost) everything that can be done with an exponential family GAM.

Simple example extension

- Here are some mass spectra for samples from patients with normal, enlarged and cancerous prostate.



- Scalar on function regression[1] model for prediction? Category (normal, enlarged or cancer) determined by a logistically distributed latent variable with mean

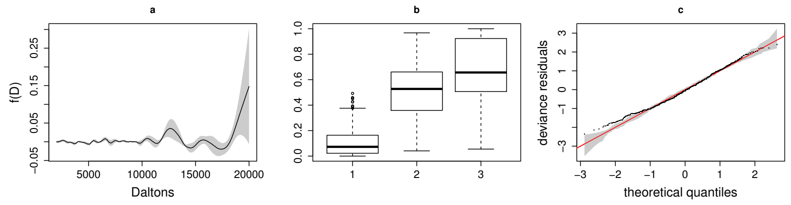
$$\mu_i = \alpha + \int f(D)\nu_i(D)dD.$$

- This is somewhat beyond a standard GAM, but methods can be developed.

[1]AKA 'signal regression'

Ordered categorical signal regression results

- Smooth adaptive f and category cut points to be estimated.
- `library(mgcv)`
  `gam(hs ~ s(D,by=M,bs="ad",k=100),family=ocat(R=3))`
  `D` and `M` are matrices. Each row contains Masses and Spectral intensity, for one subject.
- Results are…



- How is this done, and can we improve on it?

Basic aims

- To produce a general framework for the various GAM extensions that is as useful as the GAM framework. i.e.
  1. Allow numerically reliable smoothing parameter estimation for a wide range of smooths, including those with multiple smoothing penalties.
  2. Allow access to the same inferential machinery that is available for GAMs.
- General framework should reduce implementation of new model classes to the implementation of some standard derivatives of the log likelihood.
- See Wood, Pya & Säfken (2016, JASA) for full details.

Strategy

- Consider models, for data $\mathbf{y}$, in which the log likelihood, l, depends on smooth functions f_j.
- Represent the f_j using intermediate rank basis expansions with quadratic penalties, so that coefficient estimates, given smoothing parameters, $\boldsymbol{\lambda}$, are

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\text{argmax}}\ \mathcal{L}(\boldsymbol{\beta}) = l(\boldsymbol{\beta}) - \frac{1}{2}\sum_j \lambda_j \boldsymbol{\beta}^{\mathsf{T}}\mathbf{S}^j\boldsymbol{\beta}.$$

- Writing $\mathbf{S}_\lambda = \sum_j \lambda_j \mathbf{S}^j$, we can motivate the above by a prior $\boldsymbol{\beta} \sim N(\mathbf{0}, \mathbf{S}_\lambda^-)$.
- Can then estimate $\boldsymbol{\lambda}$ by Marginal Likelihood (REML).

Estimation methods: Laplace approximate ML

- We want to estimate log smoothing parameters as

$$\hat{\rho} \simeq \underset{\rho}{\text{argmax}}\ \ \log \int f(\mathbf{y}|\boldsymbol{\beta}) f(\boldsymbol{\beta}) d\boldsymbol{\beta}$$

- Can't generally do the integral.
- Approximate intergrand by exponential of 2$^{\text{nd}}$ order Taylor approx. of log intergrand (almost a multivariate normal).
- We get the Laplace Approximate ML

$$\mathcal{V}(\boldsymbol{\lambda}) = \mathcal{L}(\hat{\boldsymbol{\beta}}) + \frac{1}{2}\log|\mathbf{S}^\lambda|_+ - \frac{1}{2}\log|\boldsymbol{\mathcal{H}}| + \frac{M_p}{2}\log(2\pi).$$

where $-\boldsymbol{\mathcal{H}}$ is Hessian of penalized log likelihood, $\mathcal{L}$, at $\hat{\boldsymbol{\beta}}$.

Estimation methods: numerical strategy

- Nested Newton optimization…
- Optimize LAML $\mathcal{V}$ w.r.t. $\boldsymbol{\rho}$ by Newton's method.
- Each trial set of $\boldsymbol{\rho}$ values require corresponding $\hat{\boldsymbol{\beta}}$, also by Newton's method. Efficient as starting values excellent after first trial!
- Use implicit differentiation to obtain $\mathrm{d}\hat{\boldsymbol{\beta}}/\mathrm{d}\boldsymbol{\rho}$ etc, and hence derivatives of LAML w.r.t. $\boldsymbol{\rho}$.
- Problem: naïve computation is unstable. Worst parts are the log determinants, e.g. $\log|\sum_j \lambda_j \mathbf{S}^j|_+ \ldots$
- Some $\lambda_j \to \infty$ is legitimate, which can lead to taking logs of 'numerical zeroes' in the log determinant terms.

The problem with $\log |\mathbf{S}|_+$

- Naive $|\mathbf{S}|_+$ evaluation can go badly wrong.
- Consider $\log |\mathbf{S}_1 + \lambda \mathbf{S}_2|_+$ when $\lambda \to 0$

```
> S <- S1 + S2*1e-18; S
     [,1] [,2] [,3]   [,4]   [,5]   [,6]
[1,]    1   -1    0  0e+00  0e+00  0e+00
[2,]   -1    2   -1  0e+00  0e+00  0e+00
[3,]    0   -1    1  0e+00  0e+00  0e+00
[4,]    0    0    0  1e-18 -1e-18  0e+00
[5,]    0    0    0 -1e-18  2e-18 -1e-18
[6,]    0    0    0  0e+00 -1e-18  1e-18
> sum(log(eigen(S)$values[1:4]))    ## naive
[1] -73.39584
> sum(log(eigen(S1)$values[1:2])) + ## true
>      sum(log(eigen(S2)$values[1:2]*1e-18))
[1] -80.69584
> eigen(S)$values  ## why?
[1] 3.000000e+00 1.000000e+00 2.220446e-15 2.000000e-18
[5] 1.000000e-18 1.000000e-18
```

Further problems with $\log |\mathbf{S}|$

- Suppose that the non-zero sub matrices of $\mathbf{S}_i$ need not be full rank, but overlap...

```
> S1;S2
     [,1] [,2] [,3] [,4] [,5]            [,1] [,2] [,3] [,4] [,5]
[1,]    1   -1    0    0    0       [1,]    0    0    0    0    0
[2,]   -1    2   -1    0    0       [2,]    0    0    0    0    0
[3,]    0   -1    1    0    0       [3,]    0    0    1    0    0
[4,]    0    0    0    0    0       [4,]    0    0    0    1    0
[5,]    0    0    0    0    0       [5,]    0    0    0    0    1
> S <- S1 + S2*1e-18
> sum(log(eigen(S)$values))
[1] -115.5355
> sum(log(abs(diag(qr.R(qr(S))))))
[1] -118.3859
```

- Are either of these right? No. $\mathbf{S}_1$ is only rank 2, but its 'numerical footprint' extends beyond the first 2 columns of $\mathbf{S}$, obliterating part of the rank 3 matrix, $\mathbf{S}_2$.

Solution for $\log |\mathbf{S}|$

- Similarity transform to confine $\mathbf{S}_1$ to a block of size corresponding to its rank.
- Consider eigen-decomposition $\mathbf{U}\mathbf{D}\mathbf{U}^{\mathsf{T}} = \mathbf{S}_1$. Then $|\mathbf{S}_1 + \lambda \mathbf{S}_2| = |\mathbf{D} + \lambda \mathbf{U}^{\mathsf{T}}\mathbf{S}_2\mathbf{U}|$. If zero eigenvalues in $\mathbf{D}$ are set to exactly 0, then r.h.s. evaluates correctly.

```
> S
     [,1] [,2] [,3]  [,4]  [,5]
[1,]    1   -1    0 0e+00 0e+00
[2,]   -1    2   -1 0e+00 0e+00
[3,]    0   -1    1 0e+00 0e+00
[4,]    0    0    0 1e-18 0e+00
[5,]    0    0    0 0e+00 1e-18
> es <- eigen(S1); U <- es$vectors
> D <- es$values; D[3:5] <- 0
> Sp <- diag(D) + t(U)%*%S2%*%U * lambda
> sum(log(abs(diag(qr.R(qr(Sp))))))
[1] -124.3396
```

... why did that work, and does it generalize?

- The large elements for $\mathbf{S}_1$ got confined to a 2×2 block, in the similarity transformed version of $\mathbf{S}$

```
> Sp
              [,1]          [,2]          [,3]  [,4]  [,5]
[1,]  3.000000e+00 -2.886751e-19 -2.357023e-19 0e+00 0e+00
[2,] -2.886751e-19  1.000000e+00  4.082483e-19 0e+00 0e+00
[3,] -2.357023e-19  4.082483e-19  3.333333e-19 0e+00 0e+00
[4,]  0.000000e+00  0.000000e+00  0.000000e+00 1e-18 0e+00
[5,]  0.000000e+00  0.000000e+00  0.000000e+00 0e+00 1e-18
```

- QR decomposition to get the determinant doesn't mix columns, so the determinant calculation is now stable.
- This approach can be generalized to more than two $\mathbf{S}_j$ matrices (and to rank deficient $\mathbf{S}$).
- Re-parameterization tends to stabilize other computations as well.

Simplified notation

- Having sorted out the instability of $\log|\mathbf{S}|_+$, we can push on with a general method.
- But some expressions required to implement general methods become incomprehensibly complex without a reasonable notation …
  1. Let differentiation with respect to (Greek) parameters be denoted $f^j_\beta = \partial f/\partial\beta_j$, $g^{jk}_{\alpha\beta} = \partial^2 g/\partial\alpha_j\partial\beta_k$ etc.
  2. Greek superscripts without a subscript are labels.
  3. Repeated Roman indices occurring on only one side of an equation should be summed over. e.g. $a_j = b^{\boldsymbol{i}}_\beta w_{\boldsymbol{i}j} = \sum_i b^i_\beta w_{ij}$
- e.g. $\mathcal{L}^{i\,j}_{\beta\beta}$ is the Hessian of the penalized log-likelihood, $p_i = \mathcal{L}^{i\,\boldsymbol{j}}_{\beta\beta}\mathcal{L}^{\boldsymbol{j}}_\beta$ is the product of the Hessian and the gradient.

General case Newton step computation for $\rho = \log\lambda$

1. Reparameterize *each smooth* so that $\log|\mathbf{S}^\lambda|_+$ is stable.
2. Find $\hat{\boldsymbol{\beta}}$ by stabilized Newton method.
3. Drop unidentifiable $\hat{\boldsymbol{\beta}}$ elements at convergence. If necessary repeat Newton optimization to allow other coefficients to adjust. Let $\mathcal{L}^{\hat\beta\hat\beta}_{i\,j}$ be the inverse Hessian of $\mathcal{L}$ at $\hat{\boldsymbol{\beta}}$.
4. Compute $\mathrm{d}\hat\beta_i/\mathrm{d}\rho_k = \mathcal{L}^{\hat\beta\hat\beta}_{i\,\boldsymbol{j}}\lambda_k S^k_{\boldsymbol{jl}}\hat\beta_{\boldsymbol{l}}$.
5. Given $\mathrm{d}\hat{\boldsymbol{\beta}}/\mathrm{d}\rho_k$ compute $l^{i\,j\,k}_{\hat\beta\hat\beta\rho}$.
6. Given $l^{i\,j\,k}_{\hat\beta\hat\beta\rho}$, compute $\mathrm{d}^2\hat{\boldsymbol{\beta}}/\mathrm{d}\rho_k\mathrm{d}\rho_l$.
7. Compute $\mathcal{L}^{\hat\beta\hat\beta}_{\boldsymbol{kj}}l^{\boldsymbol{j}\,\boldsymbol{k}\,pv}_{\hat\beta\hat\beta\rho\rho}$ (model specific, $O(Mnp^2)$).
8. The derivatives of $\mathcal{V}$ can now be computed.

Implicit differentiation example

- $\mathcal{L}^i_{\hat\beta} = l^i_{\hat\beta} - \lambda_{\boldsymbol{k}}S^{\boldsymbol{k}}_{i\boldsymbol{j}}\hat\beta_{\boldsymbol{j}} = 0$
- Differentiating w.r.t. $\rho_k = \log\lambda_k$ yields

$$\mathcal{L}^{i\,k}_{\hat\beta\rho} = l^{i\,\boldsymbol{j}}_{\hat\beta\hat\beta}\frac{\mathrm{d}\hat\beta_{\boldsymbol{j}}}{\mathrm{d}\rho_k} - \lambda_k S^k_{i\boldsymbol{j}}\hat\beta_{\boldsymbol{j}} - \lambda_{\boldsymbol{l}}S^{\boldsymbol{l}}_{i\boldsymbol{j}}\frac{\mathrm{d}\hat\beta_{\boldsymbol{j}}}{\mathrm{d}\rho_k} = 0.$$

- Re-arranging and defining $\mathcal{L}^{\hat\beta\hat\beta}_{i\,j}$ as inverse of $\mathcal{L}^{i\,j}_{\hat\beta\hat\beta}$,

$$\frac{\mathrm{d}\hat\beta_i}{\mathrm{d}\rho_k} = \mathcal{L}^{\hat\beta\hat\beta}_{i\,\boldsymbol{j}}\lambda_k S^k_{\boldsymbol{jl}}\hat\beta_{\boldsymbol{l}},$$

- Now can compute $l^{i\,j\,l}_{\hat\beta\hat\beta\rho} = l^{i\,j\,\boldsymbol{k}}_{\hat\beta\hat\beta\hat\beta}\mathrm{d}\hat\beta_{\boldsymbol{k}}/\mathrm{d}\rho_l$ etc.
- In general require log likelihood derivatives to 4th order.

General case examples

- Cox PH, Cox Process, single index models, MV GAMs…
- Also GAMLSS models (Rigby & Stasinopoulus, 2005)…
  - log likelihood for the i^{th} datum is $l(y_i, \eta^1_i, \eta^2_i, \ldots)$ where the $\eta^k = \mathbf{X}^k\boldsymbol{\beta}^k$ are K linear predictors.
  - Newton estimation of $\hat{\boldsymbol{\beta}}$ + implicit differentiation requires

$$l^{\boldsymbol{j}}_{\beta^l} = l^{\boldsymbol{i}}_{\eta^l}X^l_{\boldsymbol{i}j}, \quad l^{\boldsymbol{j}\,k}_{\beta^l\beta^m} = l^{\boldsymbol{i}\,\boldsymbol{i}}_{\eta^l\eta^m}X^l_{\boldsymbol{i}j}X^m_{\boldsymbol{i}k}.$$

  - First derivatives of $\mathcal{V}$ then require

$$\begin{aligned} l^{\boldsymbol{j}\,k\,p}_{\hat\beta^l\hat\beta^m\rho} &= l^{\boldsymbol{j}\,k\,\boldsymbol{r}}_{\hat\beta^l\hat\beta^m\hat\beta^{\boldsymbol{q}}}\frac{\mathrm{d}\hat\beta^{\boldsymbol{q}}_{\boldsymbol{r}}}{\mathrm{d}\rho_p} = l^{\boldsymbol{i}\,\boldsymbol{i}\,\boldsymbol{i}}_{\hat\eta^l\hat\eta^m\hat\eta^{\boldsymbol{q}}}X^l_{\boldsymbol{i}j}X^m_{\boldsymbol{i}k}X^{\boldsymbol{q}}_{\boldsymbol{ir}}\frac{\mathrm{d}\hat\beta^{\boldsymbol{q}}_{\boldsymbol{r}}}{\mathrm{d}\rho_p} \\ &= l^{\boldsymbol{i}\,\boldsymbol{i}\,\boldsymbol{i}}_{\hat\eta^l\hat\eta^m\hat\eta^{\boldsymbol{q}}}X^l_{\boldsymbol{i}j}X^m_{\boldsymbol{i}k}\frac{\mathrm{d}\hat\eta^{\boldsymbol{q}}_{\boldsymbol{i}}}{\mathrm{d}\rho_p} \end{aligned}$$

  - Second derivatives not much worse. Generically require mixed partials of log density up to 4th order.

Less general case and software

- For a single linear predictor and a log likelihood that is the sum of separate components for each observation, then methods that exploit the regression model structure are better. Details omitted!
- These are the cases where we really have a GAM, except that the response is not exponential family distributed.
- e.g. Beta regression, ordered categorical regression, Tweedie or negative binomial with estimation of all parameters etc.
- Methods implemented in mgcv 1.8-x. e.g.

```
gam(time ~ s(x) + s(z),family=cox.ph,weights=censor)
gam(list(y ~ s(x) + s(z),~ s(v)+s(w)),family=ziplss)
```

- …follow up as for regular GAM.

Example: multinomial prostate screening model

- The ordered categorical prostate screening model was not great: categories not really ordered?
- 3 categories: $y = 0, 1, 2$ (healthy, enlarged, cancer).
- Model - smooth linear predictor, η_j, for all categories except 0.

$$L = \begin{cases} \exp(\eta_y)/\{1 + \sum_j \exp(\eta_j)\} & y > 0 \\ 1/\{1 + \sum_j \exp(\eta_j)\} & y = 0. \end{cases}$$

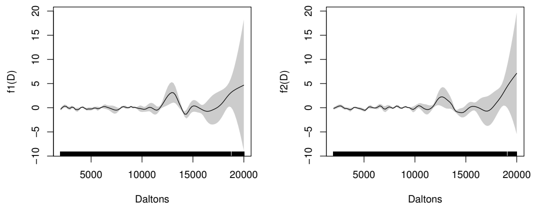
- Use ‘signal regression’ linear predictors for each η_j

$$\eta_{ji} = \alpha_j + \int f_j(D)\nu_i(D)dD.$$

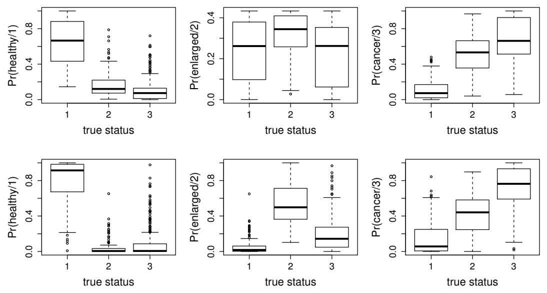
-
  ```
  gam(list(y ~ s(D,by=M),~ s(D,by=M)),
      family=multinom(K=2),data=prostate)
  ```

Multinomial prostate screening results

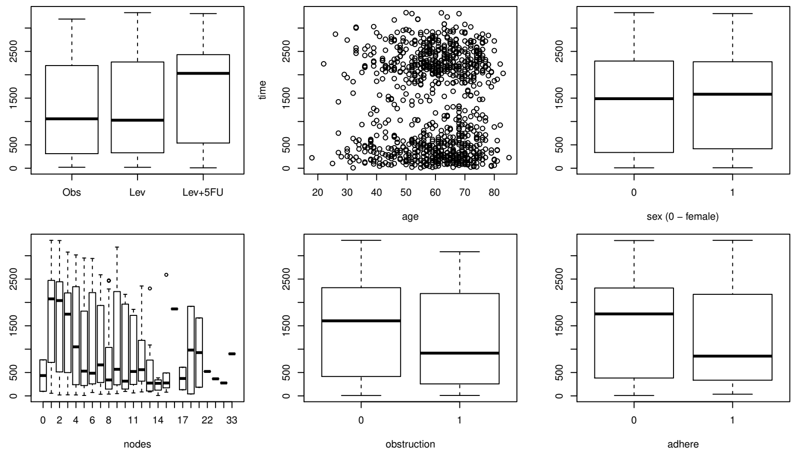
- Estimated coefficient functions



- Classification. Upper: ordered; Lower: multinomial.



Example: Colon cancer survival

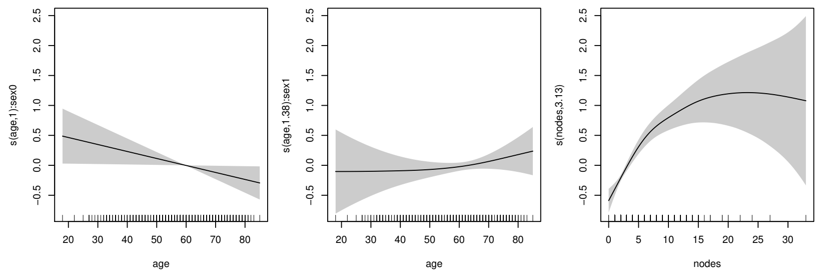


Colon cancer survival times against covariates.

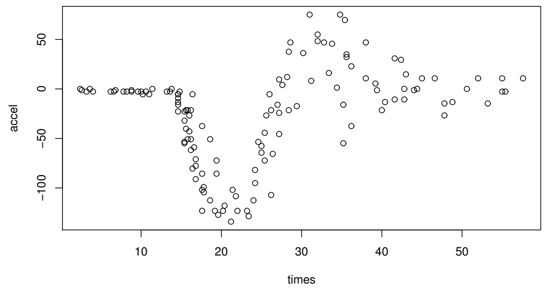
Colon cancer Cox PH model

- Cox PH model with smooth effects for age (by sex) and nodes.
- `gam(time~s(age,by=sex)+sex+s(nodes)+perfor`

```
    +rx+obstruct+adhere,family=cox.ph(),
    data=col1,weights=status)
```



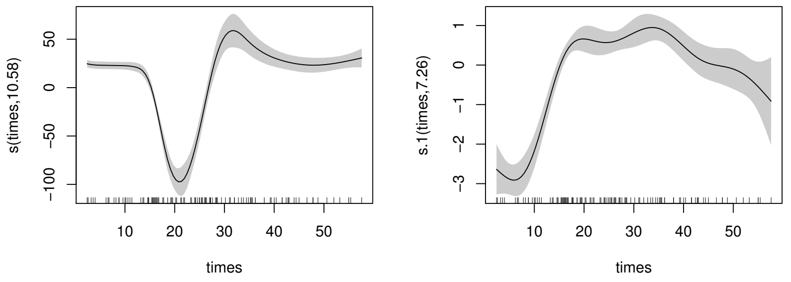
Example: location-scale model



- Mean *and variance* are clearly varying with time
- A model: $y_i = f_1(t_i) + \epsilon_i$, $\epsilon_i \sim N(0, \sigma_i^2)$, $\log \sigma_i = f_2(y_i)$.

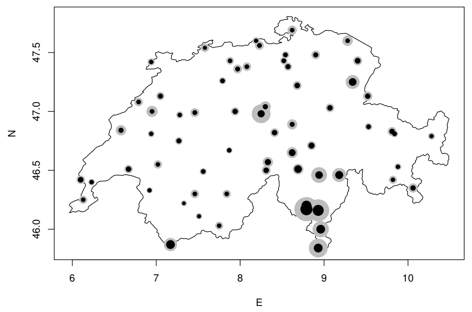
Location-scale fitting

```
b <- gam(list(accel~s(times,k=20,bs="ad"),~s(times)),
            data=mcycle,family=gaulss())
plot(b,pages=1,scale=0,scheme=1)
```



Another example: Swiss rainfall extremes

- Annual maximum 12-hour rainfall from 65 Swiss weather stations, 1981-2015. Black is average over years, grey is maximum across years.

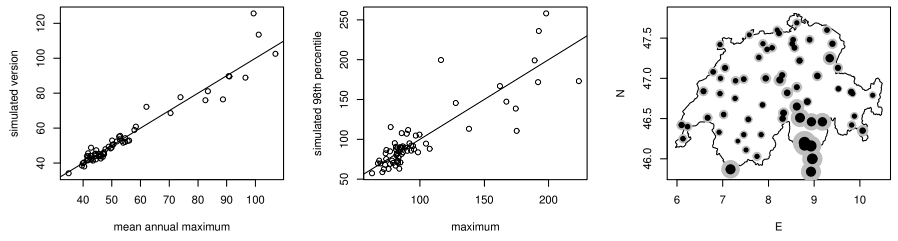


- GEV model with predictors for location, scale and shape

```
gam(list(exra~s(nao)+s(elev)+clim.reg+s(N,E),
    ~s(year)+s(elev)+clim.reg+s(N,E)
    ~clim.reg),family=gevlss,data=swer)
```

Swiss rainfall posterior simulation

- Draw model coefficients from MVN approximate posterior, and then draw annual extreme rainfall from implied GEV distribution for each station (very fast).



Smooth additive quantile regression

- The framework presented here can be extended to quantile regression.
- Idea is to replace the log likelihood with a smoothed version of the ‘pinball’ loss function used for quantile regression.
- Formally this uses the Bayesian belief updating framework of Bissiri et al (2016, JRSSB, 78(5)).
- We have to estimate an overall ‘learning rate’ parameter balancing smoothness prior and loss — this is done by a bootstrap calibration step.

Summary

- Smooth models can be extended well beyond the exponential family, and well beyond strictly additive settings.
- The extensions fit readily into a unified computational framework, that allows efficient and reliable smoothness estimation, and re-use of most of the inferential tools available for GAMs.
- A number of practically useful examples of these extensions are available in `mgcv`.
- Extensions to other loss functions also seem to be possible.

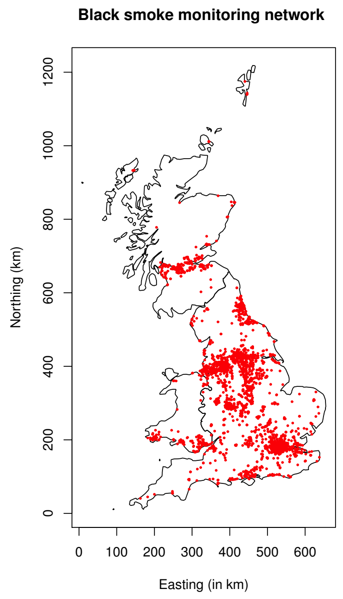
Smooth additive models for large datasets

Simon Wood
School of Mathematics, University of Bristol, U.K.

Example motivation: London smog 1952

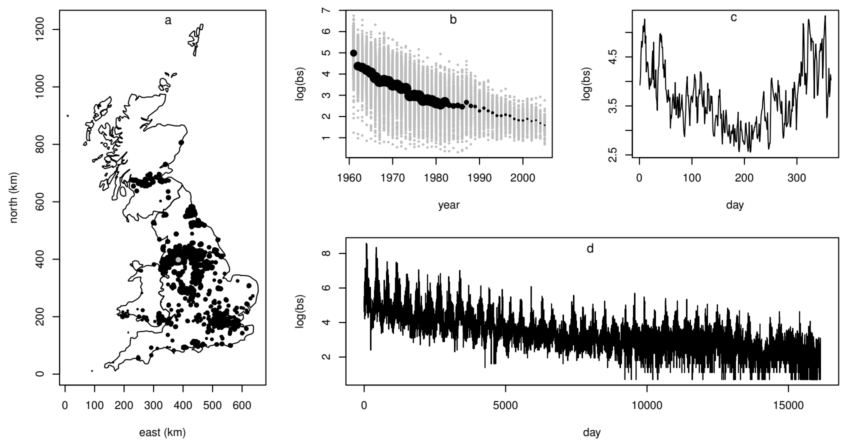
- 5-9 Dec 1952.
- 4-12 thousand premature deaths.
- Black smoke (particulates) and sulphur from domestic coal fires.
- Clean air act 1956.
- Monitoring from 1961.

Black smoke monitoring…



- 4 decades of daily ‘black smoke’ monitoring at a variable subset of the 2400+ stations shown.
- Started in 1961 to monitor air pollution (then mostly from coal), in wake of 1950s smog deaths.
- Epidemiological studies need estimates of *daily* exposure away from stations.
- $O(10^7)$ measurements and suitable smooth latent Gaussian models have $O(10^4)$ coefficients with 10-30 variance parameters.

Daily BS data



Black smoke previous work

- Shaddick & Zidek[1] modelled annual average black smoke:

$$\log(b_i) = f_1(e_i, n_i) + f_2(e_i, n_i)y_i + f_2(e_i, n_i)y_i^2 + \epsilon_i$$

  using INLA[2] for inference.
- Modelling reduces the impact of preferential sampling.
- But annual averages fail to capture the acute pollution events, on the time scale of days, that accelerate deaths, and are likely to be of long term health significance.
- A daily model for all the 10^7 data is appealing, but potentially expensive (e.g. no chance with INLA on a 24 core 128Gb machine).

[1] 2014, Spatial Statistics
[2] Rue, Martino, Chopin, 2009

The model class: recap

- Concentrate on models of the form $y_i \sim \text{EF}(\mu_i, \phi)$

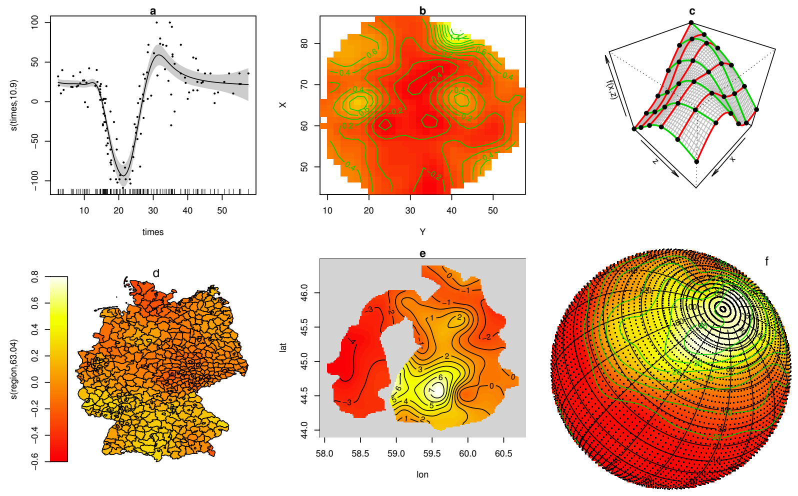
$$g(\mu_i) = \mathbf{A}_i\boldsymbol{\theta} + \sum_j f_j(x_{ji}) + \mathbf{Z}_i\mathbf{b}$$

  $\mathbf{A}, \mathbf{Z}$ are model matrices, f_j are smooth functions, $\boldsymbol{\theta}$ is parameters, $\mathbf{b}$ contains independent Gaussian random effects. g is a known link function. x_j may be vector.
- Represent smooth functions, f, using spline basis expansions with coefficients $\boldsymbol{\beta}$

$$f(x) = \sum_{k=1}^{K} \beta_k b_k(x)$$

- …and define a quadratic smoothing penalty, e.g. $\int f''^2 dx = \boldsymbol{\beta}^{\mathrm{T}}\mathbf{S}\boldsymbol{\beta}$ (suggesting $K = O(n^{1/9-1/5})$).

Wide range of basis penalty smoothers available



Estimation of model coefficients (recap)

- Given bases for the smooths, the model can be written as

$$y_i \sim \text{EF}(\mu_i, \phi), \quad g(\mu_i) = \mathbf{X}_i\boldsymbol{\beta}$$

  where $\mathbf{X}$ $(n \times p)$ contains $\mathbf{A}$, $\mathbf{Z}$ and evaluated basis functions, and $\boldsymbol{\beta}$ is a combine coefficient vector.
- The combined penalty can be written $\sum_j \lambda_j \boldsymbol{\beta}^{\mathrm{T}}\mathbf{S}_j\boldsymbol{\beta}$, where the λ_j are *smoothing parameters*.
- Then $\hat{\boldsymbol{\beta}} = \text{argmax}_{\boldsymbol{\beta}} l(\boldsymbol{\beta}) - \sum_j \lambda_j \boldsymbol{\beta}^{\mathrm{T}}\mathbf{S}_j\boldsymbol{\beta}/2$.
- …this *penalized log likelihood* can be given Bayesian motivation using the prior[3]

$$\boldsymbol{\beta} \sim N\{\mathbf{0}, (\sum_j \lambda_j \mathbf{S}_j)^{-}\}.$$

[3] which also covers simple Gaussian random effects.

A fitting algorithm (PQL/performance iteration)

- The GLM likelihood means that $\boldsymbol{\beta}$ can be estimated by penalized iteratively re-weighted least squares.
- i.e. we iteratively use penalized least squares to fit a *working linear model*:

$$\mathbf{z} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \quad \text{cov}(\boldsymbol{\epsilon}) = \mathbf{W}^{-1}\phi, \quad E(\boldsymbol{\epsilon}) = \mathbf{0}$$

  where $z_i = g'(\hat{\mu}_i)(y_i - \hat{\mu}_i) + \hat{\eta}_i$, $W_{ii}^{-1} = V(\hat{\mu}_i)g'(\hat{\mu}_i)^2$ and $\eta_i = g(\mu_i)$. V is a known and determined by EF.
- Or we use the prior on $\boldsymbol{\beta}$ and iteratively estimate

$$\mathbf{z} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \quad \boldsymbol{\beta} \sim N(\mathbf{0}, \mathbf{S}_\lambda^-) \quad \boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{W}^{-1}\phi),$$

  including $\boldsymbol{\lambda}$, as a linear mixed model (this is PQL[4]). $\mathbf{S}_\lambda = \sum \lambda_j \mathbf{S}_j$. $\boldsymbol{\lambda}$ estimation uses REML.

[4]Breslow & Clayton, 1993, JASA

Justifying the REML step and a big data method

- Let $\mathbf{QR} = \sqrt{\mathbf{W}}\mathbf{X}$ and $\mathbf{f} = \mathbf{Q}^{\mathrm{T}}\sqrt{\mathbf{W}}\mathbf{z}$. Assume $\phi = 1$.
- Then $\mathbf{f} = \mathbf{R}\boldsymbol{\beta} + \mathbf{e}$, $\boldsymbol{\beta} \sim N(\mathbf{0}, \mathbf{S}_\lambda^-)$ and, by CLT, $\mathbf{e} \sim N(\mathbf{0}, \mathbf{I})$.
- If $\hat{\boldsymbol{\beta}}_\lambda = \text{argmin}_\beta \|\mathbf{f} - \mathbf{R}\boldsymbol{\beta}\|^2 + \boldsymbol{\beta}^{\mathrm{T}}\mathbf{S}_\lambda\boldsymbol{\beta}$ then

$$-2l_r = \|\mathbf{f} - \mathbf{R}\hat{\boldsymbol{\beta}}_\lambda\|^2 + \hat{\boldsymbol{\beta}}_\lambda^{\mathrm{T}}\mathbf{S}_\lambda\hat{\boldsymbol{\beta}}_\lambda + \log|\mathbf{R}^{\mathrm{T}}\mathbf{R} + \mathbf{S}_\lambda| + \log|\mathbf{S}_\lambda|_+$$

- But to within an additive constant this is identical to $-2l_r$ for working model $\mathbf{z} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, $\boldsymbol{\beta} \sim N(\mathbf{0}, \mathbf{S}_\lambda^-)$, $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{W}^{-1})$.
- Similar argument if $\phi \neq 1$ using Pearson estimator.
- This justification also gives a *low memory footprint big data algorithm*[5]. We iterate:
  1. Iterative blockwise QR accumulates $\mathbf{R}$ and $\mathbf{f}$.
  2. Smoothing parameters and model coefficients are obtained by fitting the Gaussian model for $\mathbf{f}$.

[5]Wood, Goude and Shaw, 2015, JRSSC

QR updating for $\mathbf{R}$ and $\mathbf{f}$

- Let $\mathbf{X} = \begin{bmatrix} \mathbf{X}_0 \\ \mathbf{X}_1 \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} \mathbf{y}_0 \\ \mathbf{y}_1 \end{bmatrix}$
- Form $\mathbf{X}_0 = \mathbf{Q}_0\mathbf{R}_0$ & $\begin{bmatrix} \mathbf{R}_0 \\ \mathbf{X}_1 \end{bmatrix} = \mathbf{Q}_1\mathbf{R}$.
- Then $\mathbf{X} = \mathbf{QR}$ where $\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}\mathbf{Q}_1$.
- Also $\mathbf{f} = \mathbf{Q}^{\mathrm{T}}\mathbf{y} = \mathbf{Q}_1^{\mathrm{T}}\begin{bmatrix} \mathbf{Q}_0^{\mathrm{T}}\mathbf{y}_0 \\ \mathbf{y}_1 \end{bmatrix}$
- Applying these results recursively, $\mathbf{R}$ and $\mathbf{f}$ can be accumulated from sub-blocks of $\mathbf{X}$ without forming $\mathbf{X}$ whole.
- Above applies equally well to weighted $\mathbf{X}$ and $\mathbf{z}$.

$\mathbf{X}^{\mathrm{T}}\mathbf{X}$ updating and Cholesky

- $\mathbf{R}$ is also the Cholesky factor of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$.
- Partitioning $\mathbf{X}$ row-wise into sub-matrices $\mathbf{X}_1, \mathbf{X}_2, \ldots$, we have

$$\mathbf{X}^{\mathrm{T}}\mathbf{X} = \sum_j \mathbf{X}_j^{\mathrm{T}}\mathbf{X}_j$$

  which can be used to accumulate $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ without needing to form $\mathbf{X}$ whole.
- At same time accumulate $\mathbf{X}^{\mathrm{T}}\mathbf{y} = \sum_j \mathbf{X}_j^{\mathrm{T}}\mathbf{y}$.
- Cholesky decomposition gives $\mathbf{R}^{\mathrm{T}}\mathbf{R} = \mathbf{X}^{\mathrm{T}}\mathbf{X}$, and $\mathbf{f} = \mathbf{R}^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$.[6]
- This is twice as fast as QR approach, but less numerically stable.

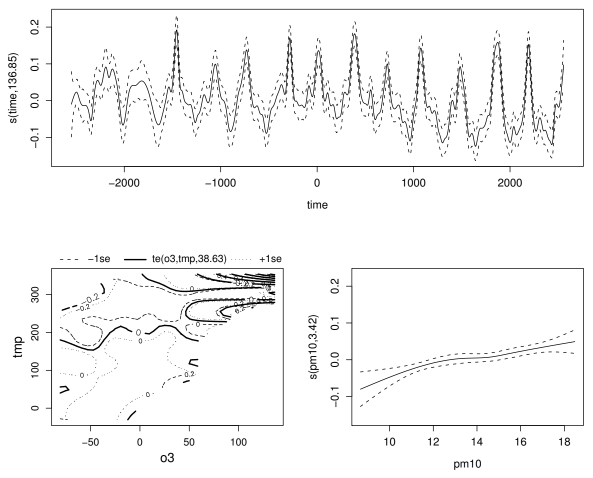
[6]A slight variant deals with the case of rank deficient $\mathbf{X}$.

Death & Air pollution example

- Around 5000 daily `death` rates, for Chicago, along with `time`, `ozone`, `pm10`, `tmp` (last 3 averaged over preceding 3 days). Peng and Welty (2004).
- Appropriate GAM is: $\texttt{death}_\texttt{i} \sim$ Poi,

  $$\log\{\mathbb{E}(\texttt{death}_i)\} = f_1(\texttt{time}_i) + f_2(\texttt{ozone}_i, \texttt{tmp}_i) + f_3(\texttt{pm10}_i).$$
- f_1 and f_3 penalized cubic regression splines, f_2 tensor product spline.
- Results suggest a *very strong* ozone - temperature interaction.

Death & Air pollution Chicago results



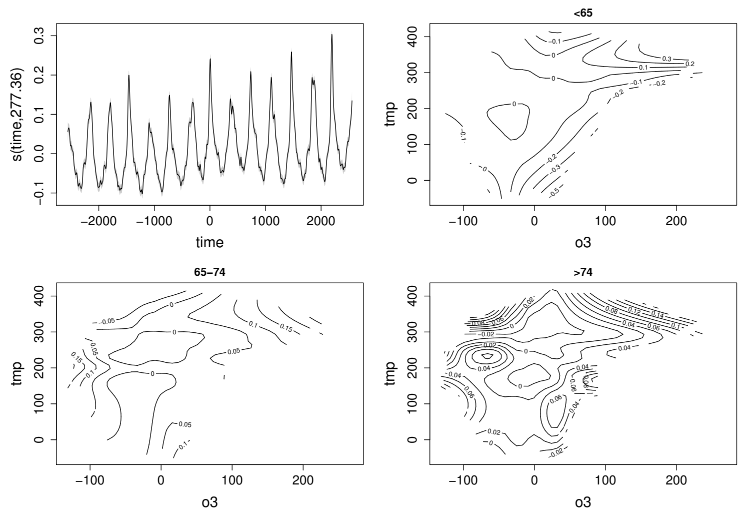
Death & Air pollution: all cities

- To test the truth of ozone-temp interaction it would be good to fit to the equivalent data for around 100 other cities, simultaneously.
- Model is

  $$\log\{E(\texttt{death}_i)\} = \gamma_j + \alpha_k + f_k(\texttt{o3}_i, \texttt{temp}_i) + f_4(t_i)$$

  if observation i is from city j and age group k (there are 3 age groups recorded).
- Model has 802 coefs and is estimated from 1.2M data.
- Fitting takes 12.5 minutes using 4 cores of a \$600 PC.
- Same model to just Chicago, takes 11.5 minutes by previous methods.

Death & Air pollution all cities results

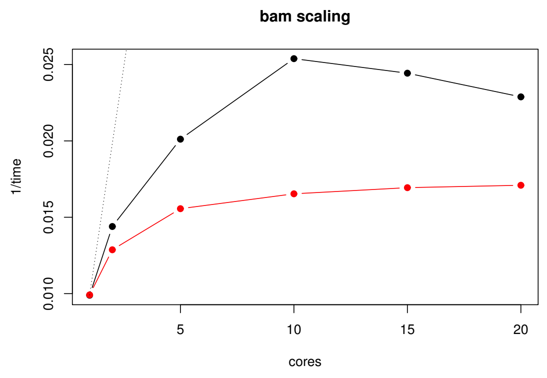


…and Black Smoke?

- Unfortunately the methods described so far would still take a couple of months of computing time to fit a reasonable model for the black smoke data.
- Initially we tried to address this issue by simple minded parallelization of the methods described above, and purchase of a 24 core workstation.
- This was not altogether successful.
- It is quite instructive to look at why.

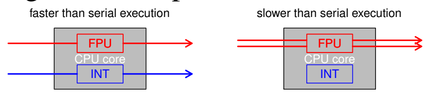
Parallelizing the GAM fit method

- **Aim**: parallelize preceding big additive model methods.
- **Hope**: 24 cores turns one day of computing into 1 hour.
- **Reality**: what happened when we parallelized all steps of flop cost $\geq O(p^3)$ in preceding (`mgcv::bam`) method…

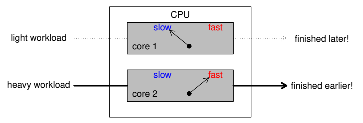


The messy realities of parallel computing

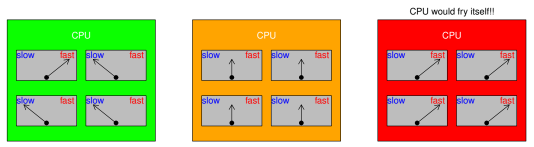
1. Hyper-threading can make parallel slower than serial…



2. Dynamic core clock speed management for power efficiency can make low work thread take most time.



3. Thermal limits: n cores are not n times faster than 1 core.



4. A floating point operation (flop) may take one or two CPU cycles, retrieving a number from memory 10 times that.
   - **Numerical computation is memory bandwidth limited**.

Memory bandwidth, Cache, block algorithms

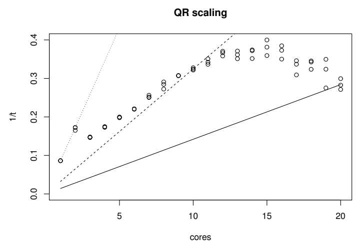
- Cache is small fast access memory between CPU and main memory.
- Big speed up if most flops involve data already in Cache.
- Consider two 10^6 flop computations
  1. $\mathbf{C}$ is a 1000×1000 matrix, and $\mathbf{y}$ a 1000-vector. Compute $\mathbf{Cy}$. Each of 10^6 elements of $\mathbf{C}$ read once, no re-use.
  2. $\mathbf{A}$ and $\mathbf{B}$ are both 100×100 matrices. Form $\mathbf{AB}$. Repeatedly revisits the 2×10^4 elements of $\mathbf{A}$ and $\mathbf{B}$.

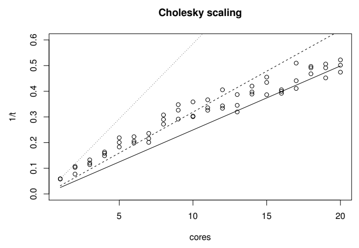
  …provided $\mathbf{A}$ and $\mathbf{B}$ fit in Cache, 2 is *much* faster.
- Structure algorithms around Cache friendly blocks! e.g.

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{11}\mathbf{B}_{11} + \mathbf{A}_{12}\mathbf{B}_{21} & \mathbf{A}_{11}\mathbf{B}_{12} + \mathbf{A}_{12}\mathbf{B}_{22} \\ \mathbf{A}_{21}\mathbf{B}_{11} + \mathbf{A}_{22}\mathbf{B}_{21} & \mathbf{A}_{21}\mathbf{B}_{12} + \mathbf{A}_{22}\mathbf{B}_{22} \end{bmatrix}$$

Parallel block pivoted QR & Cholesky

- There are parallelizable block pivoted QR[7] and Cholesky[8] algorithms published,
- Here's how they scale with openMP parallelization…





- Pivoted QR has alot of unavoidable matrix-vector operations (instead of matrix-matrix).
- …this caused the poor scaling of the GAM fitting method.

[7]Quintana-Ortí, Sun and Bishop, 1998, SIAM J. Sci. Comp.
[8]Lucas 2004, LAPACK working paper

Scalable fitting: Cholesky only?

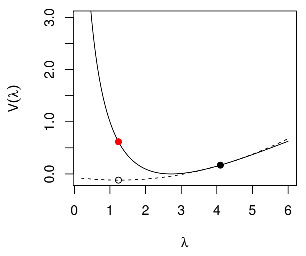
- The (restricted) marginal likelihood used for $\boldsymbol{\lambda}$ estimation

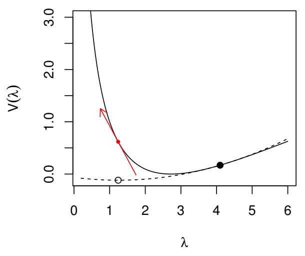
$$\mathcal{V}(\boldsymbol{\lambda}) = \frac{\|\sqrt{\mathbf{W}}(\mathbf{z}-\mathbf{X}\hat{\beta}_\lambda)\|^2 + \hat{\beta}_\lambda^{\mathrm{T}}\mathbf{S}_\lambda\hat{\beta}_\lambda + k}{2\phi} + \frac{n-M_p}{2}\log(2\pi\phi) + \frac{\log|\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X}+\mathbf{S}_\lambda| - \log|\mathbf{S}_\lambda|_+}{2}$$

  is difficult because of the $\log|\cdot|$ terms: naive computation can be equivalent to taking logs of numerical zeroes!
- Stable $\log|\cdot|$ requires pivoted QR (& eigen) decomposition, but we would prefer Cholesky only to get scalability.
- Simple idea: avoid QR and eigen-decompostions by avoiding evaluating log determinants during optimization.

Gradient only Newton optimization

- Newton step, Δ, uses only 1st and 2nd derivatives of $\mathcal{V}$.
- Function values only required for step control
  - 'halve Δ if $\mathcal{V}(\lambda+\Delta) > \mathcal{V}(\lambda)$.'
- Instead 'halve Δ if $\nabla\mathcal{V}^{\mathrm{T}}\Delta > 0$'.





- This eliminates the 'log(0) problem'. e.g., if $\rho = \log\lambda$,

$$\frac{\partial\log|\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X}+\sum\lambda_j\mathbf{S}_j|}{\partial\rho_j} = \mathrm{tr}\left\{(\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X}+\sum\lambda_j\mathbf{S}_j)^{-1}\mathbf{S}_j\lambda_j\right\}$$

Simple idea 2

1. Conventional PQL finds best fit variance/smoothing parameters for working model.
2. Waste of effort to find *best* fit when working model will anyway be discarded at next iteration!
3. Just find improved smoothing parameters by taking one Newton step each iteration.
4. With care we can still step control properly (can even prove convergence for some cases).
5. Actual iteration omitted for reasons of tediousness…

Simple idea 3: discrete covariate methods

- The methods so far scale like the pivoted Cholesky (rather than QR) and turn months of computing into days-weeks.
- Formation of $\mathbf{X}^\mathrm{T}\mathbf{W}\mathbf{X}$ is the leading order cost: $O(np^2)$.
- Lang et al.[9] point out that for a single 1D smooth, $f(x)$, the product $\mathbf{X}^\mathrm{T}\mathbf{W}\mathbf{X}$ is very efficiently computable if x has only $m \ll n$ discrete values.
- As statisticians we should be prepared to discretise x to $m = O(\sqrt{n})$ bins.
- It is possible to find (novel) efficient computational methods for the multiple discretised covariate case, both for multiple 1D smooths and for 'tensor product' smooths of multiple covariates.

[9] Lang, Umlauf, Wechselberger, Harttgen & Kneib, 2014, Statistics & Computing.

Simple discrete method example

- For a single smooth, its $n \times p_j$ model matrix becomes

$$X_j(i,l) = \bar{X}_j(k_j(i),l)$$

where $\bar{\mathbf{X}}_j$ is an $m_j \times p_j$ matrix evaluating the smooth at the corresponding gridded values.

- Then, for example

$$X_j^\mathrm{T} y = \bar{X}_j^\mathrm{T}\bar{y} \;\text{ where }\; \bar{y}_l = \sum_{k_j(i)=l} y_i$$

Cost: $O(n) + O(m_j p_j)$ – for $m_j \ll n$ this a factor of p_j saving.

- In general all required (cross)products are a factor of p_j more efficient, where p_j is the largest (marginal) basis dimension involved in the term.

Black smoke modelling

- Method based on parallel Cholesky, determinant free iteration and discretization of covariates implemented in `mgcv` function `bam(...,discrete=TRUE)`.
- The current 'best' daily black smoke model is

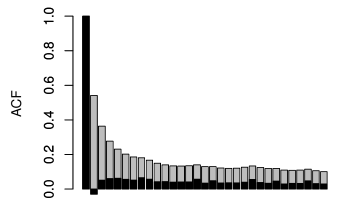
$$\begin{aligned}\log(\texttt{bs}_i) = &f_1(\texttt{y}_i) + f_2(\texttt{doy}_i) + f_3(\texttt{dow}_i)\\ &+ f_4(\texttt{y}_i,\texttt{doy}_i) + f_5(\texttt{y}_i,\texttt{dow}_i) + f_6(\texttt{doy}_i,\texttt{dow}_i)\\ &+ f_7(\texttt{n}_i,\texttt{e}_i) + f_8(\texttt{n}_i,\texttt{e}_i,\texttt{y}_i) + f_9(\texttt{n}_i,\texttt{e}_i,\texttt{doy}_i) + f_{10}(\texttt{n}_i,\texttt{e}_i,\texttt{dow}_i)\\ &+ f_{11}(\texttt{h}_i) + f_{12}(\texttt{T}_i^0,\texttt{T}_i^1) + f_{13}(\bar{\texttt{T1}}_i,\bar{\texttt{T2}}_i) + f_{14}(\texttt{r}_i) + \alpha_{k(i)} + b_{\text{id}(i)} + e_i\end{aligned}$$

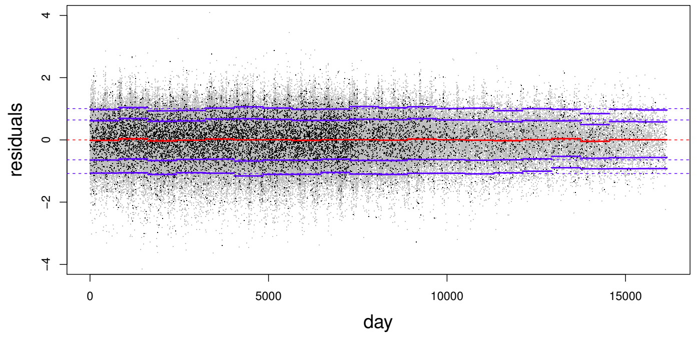
The model has around 10^4 coefficients and $r^2 = 0.79$.

- With the new parallel discretised methods fit time is < 1 hour. We estimate that previous methods would have required > 1 month. Memory footprint is about 15Gb.

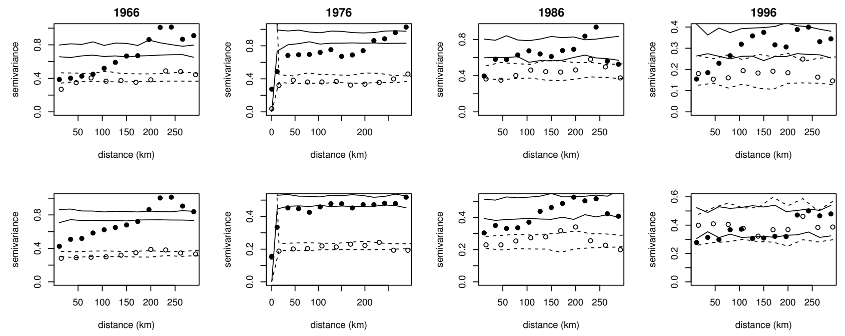
Daily black smoke model - 10 day timestep

BS residuals in time





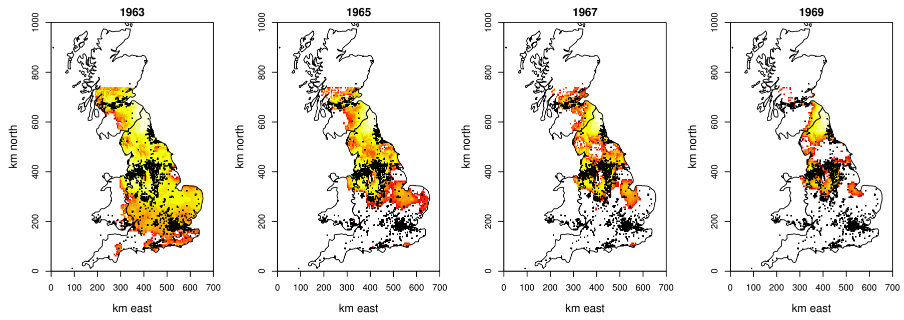
BS variograms in space



- Black solid is raw data, open is model residuals.
- Lines are permutation based reference intervals.
- Top row, day 40. Bottom row, day 200.

Aside: penalty choice matters - a bad model movie

Derived maps — Posterior exceedance probabilities



- Map shows average daily probability of exceeding current EU daily limit, for 4 years in the 1960s.
- Notice how the switch from coal to gas for domestic heating cleans up the London area quite quickly.
- The northern industrial city areas take longer.

Conclusions

- Possible to obtain three orders of magnitude computational speed-up in large additive model fitting by combining
    1. A new iterative algorithm for REML based additive model estimation. Requires only Cholesky decomposition, and avoids evaluation of unstable log determinants.
    2. OpenMP parallelization of modern pivoted block Cholesky.
    3. New methods to efficiently compute all the model matrix products required in fitting, based on discretization of covariates (including tensor product smooths).
- The methods facilitate the first *daily* models of UK black smoke densities based on the multi-decade daily data from the UK black smoke monitoring network.
- See Wood, Li, Shaddick and Augustin (2017) JASA.