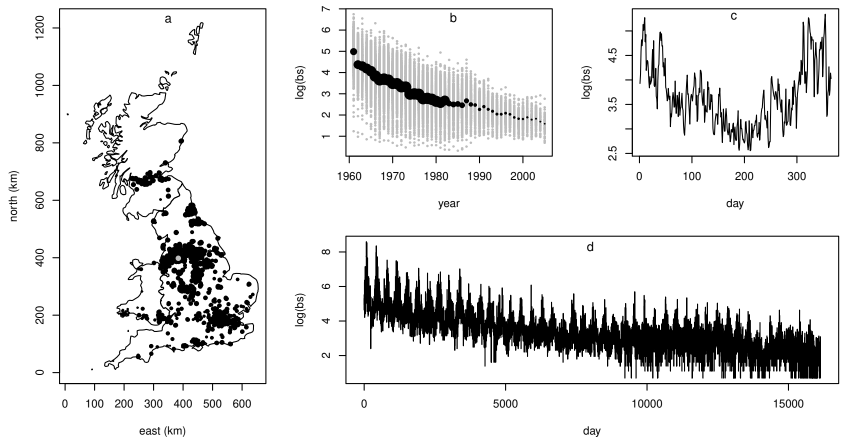
More advanced smooth modelling

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

Bigger model/data methods

- Some data-model size combinations impractical with `gam`, e.g. UK 'black smoke' daily data. $n \approx 10^7$ needs $\approx 10^4$ coef model.



Big model/data strategy

- Iteratively estimate smoothing parameters using REML for working penalized linear models.
  1. As for GLM fitting, Newton optimization of penalized likelihood can be re-cast as iterative fitting of a weighted working penalized linear model.
  2. REML for that working model is fairly easy to optimize and to justify as a large sample approximation to original model REML.
  3. Fastest if just a single Newton $\log(\boldsymbol{\lambda})$ update step is taken for each working model.
- Adjust numerical methods to exploit 'cache friendly' matrix methods, and parallelize.
- Exploit fact that covariates usually take $\ll n$ discrete values, or can be discretized to $O(n^{1/2})$ unique values without statistically important information loss.
- Only available for mean (location) regression models so far, not location scale etc.

Discrete covariate methods

- Formation of $\mathbf{X}^\mathsf{T}\mathbf{W}\mathbf{X}$ is the leading order cost: $O(np^2)$.
- Lang et al.[1] point out that for a single 1D smooth, $f(x)$, the product $\mathbf{X}^\mathsf{T}\mathbf{W}\mathbf{X}$ 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 similarly efficient methods for all the expensive matrix products in the multiple smooth discretised covariate case, both for multiple 1D smooths and for 'tensor product' smooths of multiple covariates.

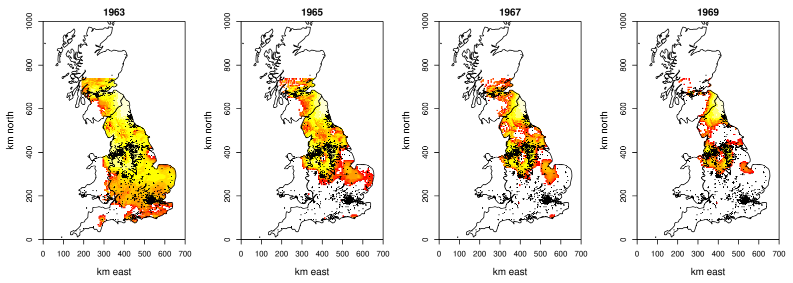
[1] Lang, Umlauf, Wechselberger, Harttgen & Kneib, 2014, Statistics & Computing.

Simple discrete method example

- Suppose a variable x_j has m_j unique values $\bar{x}_j$ such that $x_j(i) = \bar{x}_j(k_j(i))$ where k_j is an appropriate index vector.
- For a single smooth of x_j, 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.
- For example, direct computation of $X_j^\mathsf{T} y$ has $O(np_j)$ cost, but
$$X_j^\mathsf{T} y = \bar{X}_j^\mathsf{T} \bar{y} \text{ where } \bar{y}_l = \sum_{k_j(i)=l} y_i$$
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.

`bam(...,discrete=TRUE)`

- The preceding methods are implemented by function `bam`, with option `discrete=TRUE`.
- `bam` is used like `gam`, although not all features and options are available.
- For the black smoke model it is at least $10^4\times$ faster than `gam` with a tiny fraction of the memory footprint. See `?bam`.



Linear functionals of smooths

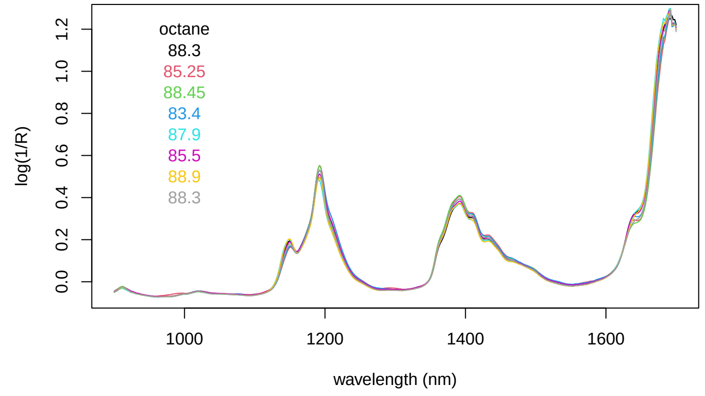
- In the models looked at so far, functions always contributed to the linear predictor by being evaluated at single covariate points.
- But the same methods will work with linear functionals of smooths. For example
  - Consider modelling respiratory hospitalizations as a function of ozone levels over the preceding 5 days. A suitable model term for the ith day might be $\sum_{d=0}^{5} f(\texttt{o3}_{i-d}, d)$ where f is a tensor product smooth of ozone and lag (distributed lag).
  - Sometimes covariates take the form of observations of whole functions, e.g. $x_i(\nu)$, and a suitable contribution to η_i might be
  $$\int f(\nu) x_i(\nu) d\nu$$
  where f is a smooth function (signal regression).
  - Occasionally the response depends not on a point evaluation of f, but on its integral over some range.

Using linear functionals of smooths

- Linear functionals can always be discretized into weighted sums of point evaluations of smooths.
- `mgcv` then allows them to be added to a model using a *summation convention*.
- Specifically, suppose `X` is a matrix of covariate values, and `W` a weight matrix of the same dimension, both with one row per y_i.
- Then `s(X,by=W)` implements the term
$$\sum_j f(X_{ij}) W_{ij}$$
for a smooth function f. e.g. for an integral `X` and `W` would be quadrature points and weights.
- Similarly `s(X,Z,by=W)` is $\sum_j f(X_{ij}, Z_{ij}) W_{ij}$. `te` also useable.

Linear functional signal regression example

- Consider predicting octane from near IR spectra of fuel samples.

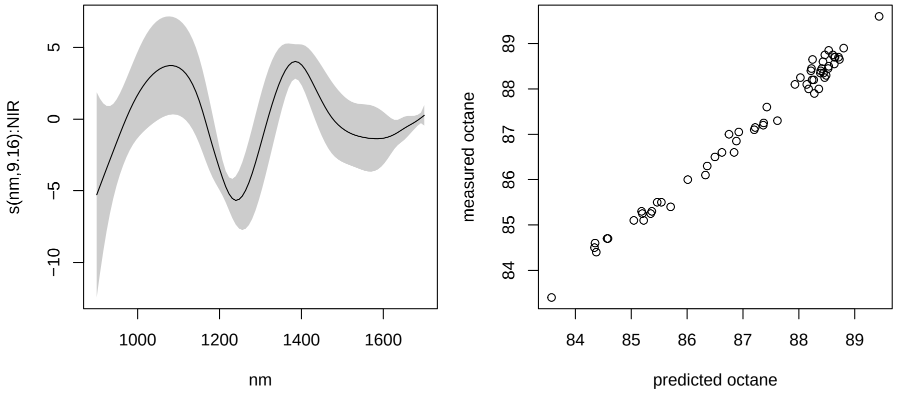


- A model is $\texttt{octane}_\texttt{i} = \sum_j f(\texttt{nm}_{ij})\texttt{NIR}_{ij} + \epsilon_i$, where $\texttt{NIR}_{ij}$ is intensity at wavelength $\texttt{nm}_{ij}$ – jth wavelength for ith sample.

Octane model estimation and fit

- Let matrices `nm` and `NIR` have samples in rows, wavelength/ intensity in cols. Then fit with `mgcv`...

```
library(gamair) ## for data
b <- gam(octane~s(nm,by=NIR,k=50),data=gas)
```



Deconvolution example: Covid incidence

- Can't observe Covid infections happening as there are variable delays from infection to disease onset.
- Working out when Covid fatalities[2] were infected provides useful information on how Covid infection rate changed. This is a deconvolution problem.
- Simple model of `deaths` each `day` is $\texttt{deaths}_i \sim \text{Poi}(\mu_i)$

$$\mathbb{E}[\texttt{deaths}_i] = \mu_i = \sum_{d=0}^{d^+} f(\texttt{day}_i - d)\pi(d),$$

- π is infection to death distribution (several studies, including ISARIC); smooth function $f(t)$ is daily new infections.

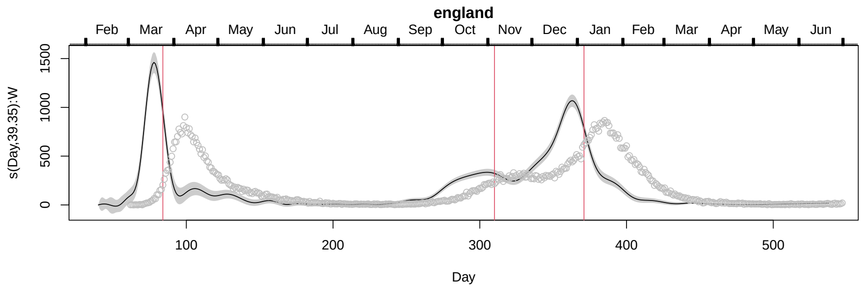
[2]NHS England hospital deaths with Covid.

Covid deconvolution with `gam`

- Modelling zeroes with Poisson and identity link awkward - use normal approximation to Poisson - variance from pilot smooth.

```
b0 <- gam(deaths ~ s(julian,k=k),family=poisson,method="REML")
w <- 1/fitted(b0) # set var proportional to pilot run E(deaths)
b <- gam(deaths ~ s(Day,by=W,k=k,bs="ad",m=8,pc=list(Day=40))+
                   s(dow,k=7,bs="cc")-1,weights=w,
                   method="REML",knots=list(dow=c(0,7)))
```

- Uses matrix summation convention where $\texttt{Day}_{ij} = \texttt{julian}_i - j$ and $\texttt{W}_{ij} = j$. `dow` is day of week (matters in UK).

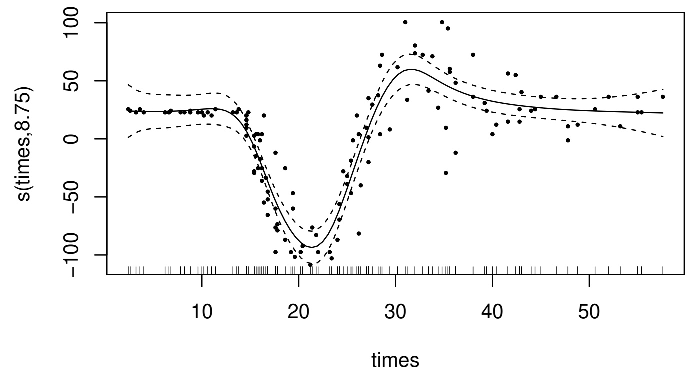


More advanced posterior inference

- Suppose that we are interested in inference about some quantity that is non-linear in the model coefficients. How can we obtain its posterior distribution or a CI for it?
- *Posterior simulation* is a convenient approach. 2 options...
  1. Simulate a large set of i.i.d. coefficient vectors from the Gaussian approximate posterior $N\{\hat{\boldsymbol{\beta}}, (\hat{\boldsymbol{\mathcal{I}}} + \mathbf{S}_\lambda)^{-1}\}$, and compute the quantity of interest from each. From this posterior sample, CIs and the approx. posterior density are computable.
  2. Use the approximate posterior as the basis for proposals in a simple Metropolis Hastings sampler, which simulated samples from the exact posterior of $\boldsymbol{\beta}$.
- If the Gaussian posterior is suspected to be poor?
  1. Again simulate from posterior using Metropolis Hastings, or
  2. use an INLA approximation.

Posterior simulation example

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



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

Trough to peak CI

```
library(MASS)
b <- gam(accel~s(times,bs="ad"),data=mcycle,method="REML")
pd <- data.frame(times=seq(10,40,length=1000))
Xp <- predict(b,pd,type="lpmatrix") ## map coefs to fitted curves
beta <- coef(b);Vb <- vcov(b) ## posterior mean and cov of coefs
n <- 10000
br <- rmvn(n,beta,Vb) ## simulate n rep coef vectors from post.
a.range <- rep(NA,n)
for (i in 1:n) { ## loop to get trough to peak diff for each sim
  pred.a <- Xp%*%br[i,]  ## curve for this replicate
  a.range[i] <- max(pred.a)-min(pred.a) ## range for this curve
}
quantile(a.range,c(.025,.975))
    2.5%    97.5%
139.0497 178.4933
```

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

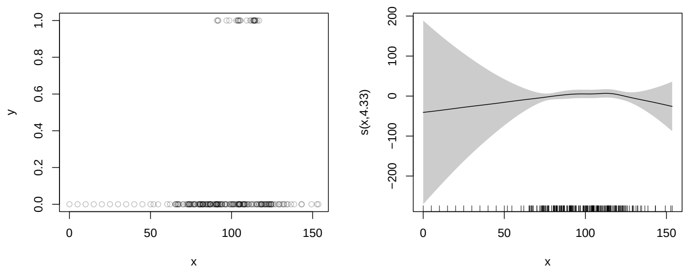
Trough to peak CI by Metropolis Hastings

```
bs <- gam.mh(b,ns=10000,burn=1000,t.df=40,rw.scale=.25,thin=1)
  |======================================================| 100%
fixed acceptance =  0.7804   RW acceptance =  0.1214
br <- bs$bs ## ... then same code as above given bs...
    2.5%    97.5%
138.6949 178.7967
```

- But actually the MH sampling was pointless here. The Gaussian posterior is exact for a Gaussian likelihood.

Posterior Gaussian failure

```
b <- gam(y ~ s(x, k = 15),method = 'REML', family = binomial)
```

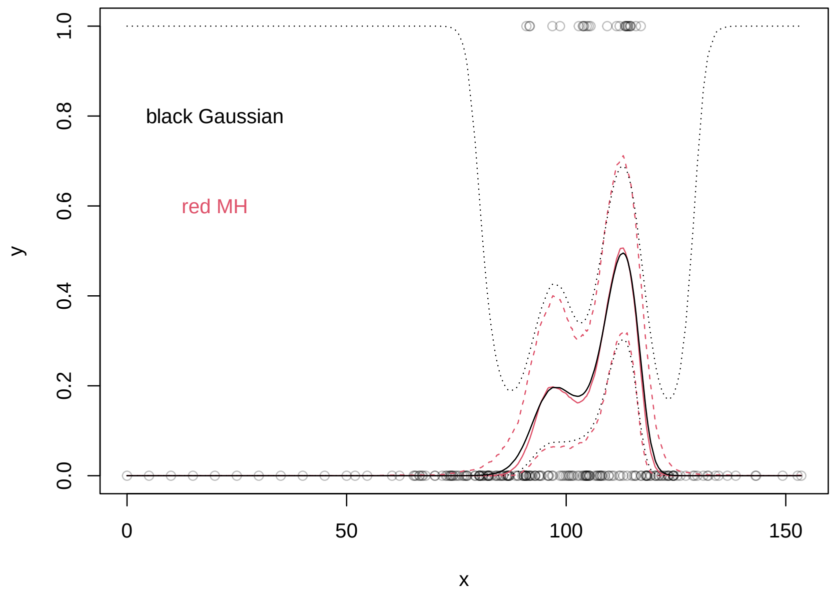


- Right smooth estimated from left binary data.
- Gaussian posterior approximation fails. Symmetry of approximation completely wrong here.

MH posterior sampling solution

```
br <- gam.mh(b,thin=2,ns=2000,rw.scale=.4)$bs
X <- model.matrix(b)
plot(x, y, col = rgb(0,0,0,0.25), ylim = c(0,1))
ff <- X%*%t(br) ## posterior curve sample
linv <- b$family$linkinv ## inverse of logit link
## Get intervals for the curve on the response scale...
fq <- linv(apply(ff,1,quantile,probs=c(.025,.5,.975)))
lines(x,fq[1,],col=2,lty=2);lines(x,fq[3,],col=2,lty=2)
lines(x,fq[2,],col=2);
## Compare to the Gaussian posterior approximation
fv <- predict(b,se=TRUE)
lines(x,linv(fv$fit))
lines(x,linv(fv$fit-2*fv$se.fit),lty=3)
lines(x,linv(fv$fit+2*fv$se.fit),lty=3)
```

MH posterior sampling results



The basic INLA idea

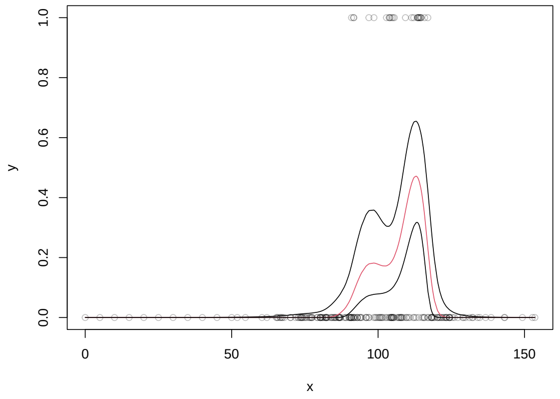
- The key idea in INLA is

$$\pi(\beta_i|\mathbf{y}) = \frac{\pi(\boldsymbol{\beta},\mathbf{y})}{\pi(\boldsymbol{\beta}_{-i}|\beta_i,\mathbf{y})} \simeq \frac{\pi(\tilde{\boldsymbol{\beta}},\mathbf{y})}{\pi_{gg}(\tilde{\boldsymbol{\beta}}_{-i}|\beta_i,\mathbf{y})} = \tilde{\pi}(\beta_i|\mathbf{y})$$

  where π_{gg} is some Gaussian approximation to $\pi(\tilde{\boldsymbol{\beta}}_{-i}|\beta_i,\mathbf{y})$ and $\tilde{\boldsymbol{\beta}}$ maximizes the joint density subject to constraint $\tilde{\beta}_i = \beta_i$.
- $\tilde{\pi}(\beta_i|\mathbf{y})$ approximation much better in tails than π_G, because:
  1. we only evaluate the Gaussian approximation at its mean, not out in its inaccurate tails.
  2. the approximation error enters multiplicatively, rather than growing into the tails
  3. a univariate marginal is easy to renormalize.
- $\tilde{\boldsymbol{\beta}}$ relatively cheap to obtain numerically, but have to base π_{gg} on approximation to exact Hessian, otherwise too costly.
- INLA also integrates over smoothing parameters - less important.

mgcv ginla code

```
G <- gam(y ~ s(x, k = 15),fit=FALSE, family = binomial)
b <- gam(G=G,method="REML")
xp <- quantile(x,prob=seq(0,1,length=15))
## matrix mapping coefs to function values at 15 points...
Xp <- predict(b,newdata=data.frame(x=xp),type="lpmatrix")
inap <- ginla(G,Xp)
## [slightly involved 90% CI plotting code omitted]
```



Neighbourhood Cross Validation and Autocorrelation

- Could use other fit measures, not just log likelihood.
- Given some loss function $\mathcal{L}$

$$\hat{\boldsymbol\beta} = \underset{\boldsymbol\beta}{\text{argmin}} \sum\nolimits_i^n \mathcal{L}(y_i, \boldsymbol\beta) + \sum\nolimits_k \lambda_k \boldsymbol\beta^\mathsf{T} \mathbf{S}_k \boldsymbol\beta$$

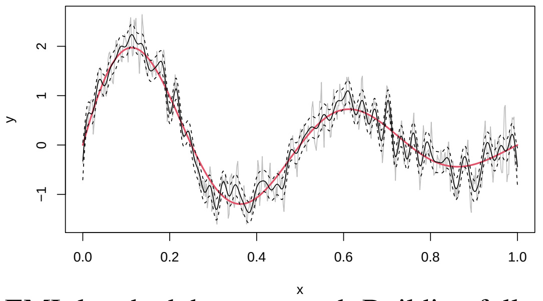
  - MAP, if $\mathcal{L}$ is negative log likelihood.
- Could cross validate. If $\hat{\boldsymbol\beta}^{[-i]}$ is $\hat{\boldsymbol\beta}$ when y_i omitted from fit,

$$\hat{\boldsymbol\lambda} = \underset{\boldsymbol\lambda}{\text{argmin}} \sum_i^n \mathcal{L}(y_i, \hat{\boldsymbol\beta}^{[-i]})$$

- Works with any smooth loss (e.g. ELF loss for quantile regression), can use link with Jackknife to get uncertainty of $\hat{\boldsymbol\beta}$.

CV autocorrelation Problem

- In the presence of short range un-modelled autocorrelation cross validation 'overfits'.
- e.g. $y_i = f_t(x_i) + (e_{i-1} + e_i + e_{i+1})/3$, where e_i are i.i.d., fitted using GCV ignoring correlation (f_t in red)...



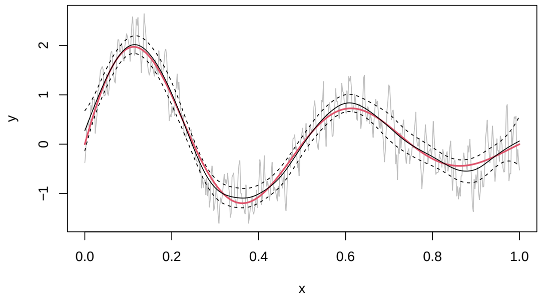
- REML less bad, but not good. Building full autocorrelation model often awkward, expensive and a distraction.

NCV autocorrelation solution

- Cross validate by leave-out-neighbours (e.g. Chu & Marron 1991 AOS; Roberts et al. 2017 Ecography).

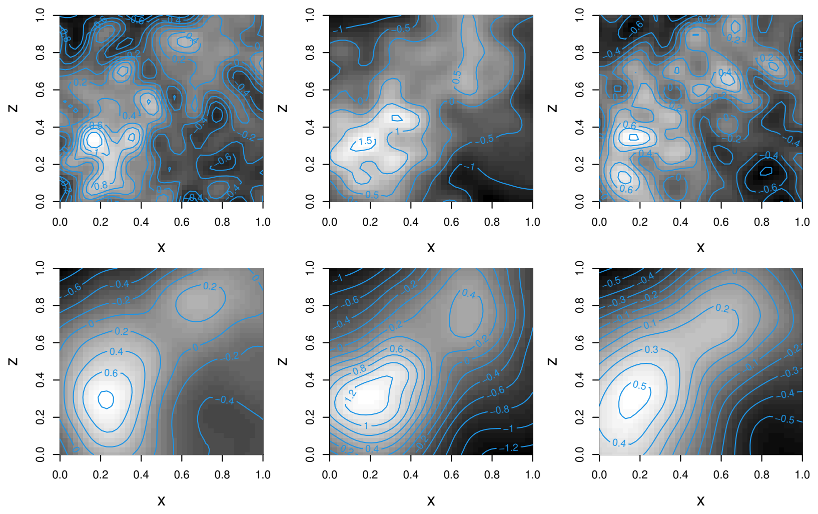
$$\hat{\boldsymbol\lambda} = \underset{\boldsymbol\lambda}{\text{argmin}} \sum_i^n \mathcal{L}(y_i, \hat{\boldsymbol\beta}^{-\texttt{nei}(i)})$$

- Solves the problem, but at high computational cost in general.



- It turns out that for smooth regression models there's a cheap and accurate ($O(n^{-2})$) approximation.

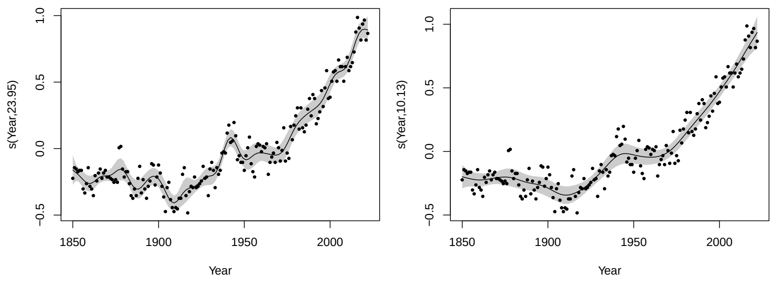
Works for spatial as well



NCV in `gam`

```
b0 <- gam(TA~s(Year,k=30),data=gt,method="NCV") ## leave one out
n <- nrow(gt)
## k[m[i-1]:m[i]] indexes neighbours of i to drop
k <- -1:1 + rep(1:n,each=3); k <- k[3:length(k)-1]
nei <- list(k=k,m= cumsum(c(2,rep(3,n-2),2)))

b <- gam(TA~s(Year,k=30),data=gt,method="NCV",nei=nei)
par(mfrow=c(1,2),mar=c(4,4,1,1))
plot(b0,rug=FALSE,residuals=TRUE,pch=19,cex=.5,scheme=1)
plot(b,rug=FALSE,residuals=TRUE,pch=19,cex=.5,scheme=1)
```



A few other packages (not exhaustive)

- `qgam` for smooth quantile regression.
- `scam` for shape constrained GAMs.
- `mgcViz` for `ggplot` based GAM visualization and checking, including for big data.
- `INLA` for the full power of the INLA method.
- `gamlss` and VGAM for GAMLSS and multivariate models.
- `gss` for smoothing spline ANOVA.
- `GAMPL` in SAS, for `mgcv` style GAMs in SAS.