Basis Penalty Smoothers

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

Estimating functions

- It is sometimes useful to estimate smooth functions from data, without being too precise about the functional form.
- When estimating such unknown functions there are 2 desirable properties
  1. A reasonable approximation to the truth, without too much mis-specification bias.
  2. Reasonable computational efficiency.
- The basis-penalty approach is a reasonable compromise between these objectives.

Bases and penalties

- Write the function to be estimated as

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

  where β_k are coefficients to be estimated, and $b_k(x)$ are known *basis functions* chosen for convenience.
- PROBLEM: If K is large enough definitely avoid underfit, it will probably overfit!
- SOLUTION: During fitting, penalize departure from smoothness with a penalty, e.g....

$$\mathcal{P}(f) = \int f''(x)^2 dx = \boldsymbol{\beta}^{\mathsf{T}} \mathbf{S} \boldsymbol{\beta}$$

Why does $\int f''(x)^2 dx = \beta^{\mathsf{T}} \mathbf{S} \beta$?

- We can write f'' in terms of β and a vector of basis function derivatives

$$f''(x) = \sum_{k}^{K} \beta_k b_k''(x) = \beta^{\mathsf{T}} \mathbf{b}(x)$$

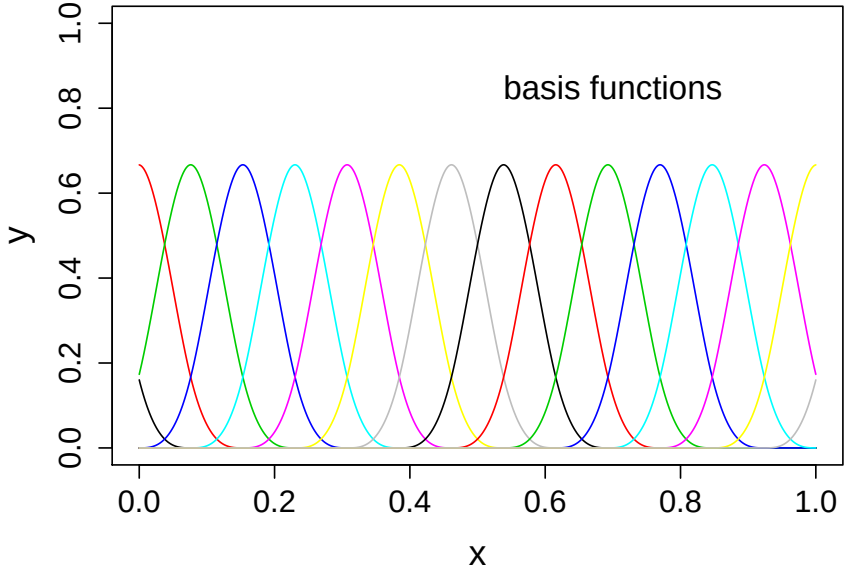
  where $\mathbf{b}(x) = [b_1''(x), b_2''(x), \ldots]^{\mathsf{T}}$.

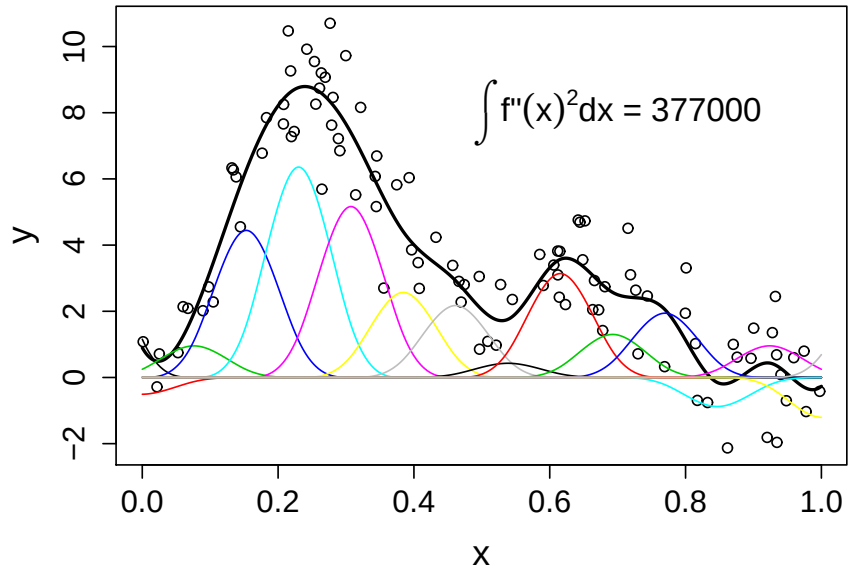
- Then the penalty can also be re-written

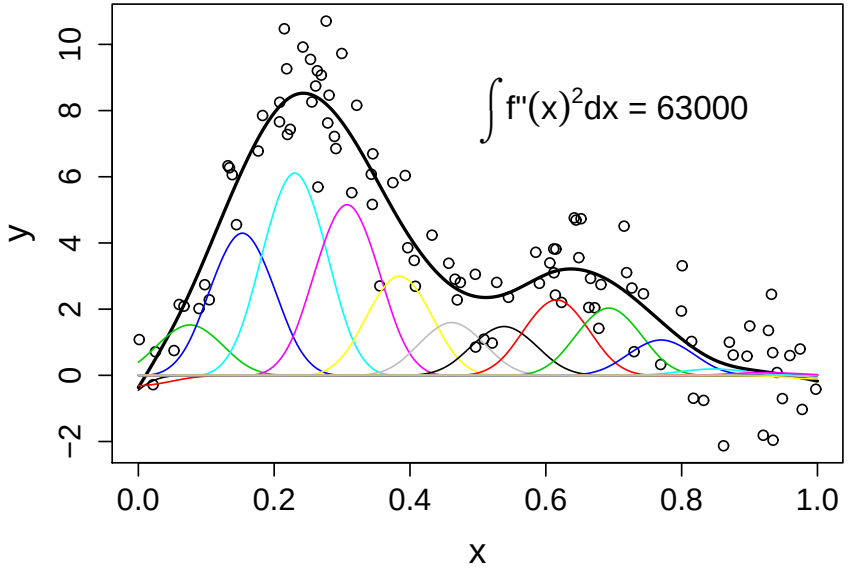
$$\begin{aligned} \int f''(x)^2 dx &= \int \beta^{\mathsf{T}} \mathbf{b}^{\mathsf{T}}(x) \mathbf{b}(x) \beta dx \\ &= \beta^{\mathsf{T}} \int \mathbf{b}^{\mathsf{T}}(x) \mathbf{b}(x) dx \beta = \beta^{\mathsf{T}} \mathbf{S} \beta \end{aligned}$$

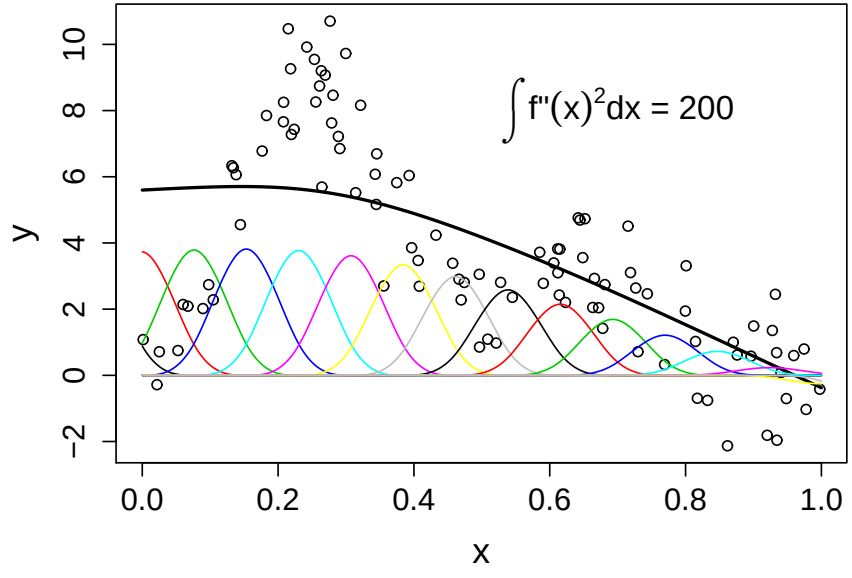
  where $\mathbf{S} = \int \mathbf{b}^{\mathsf{T}}(x) \mathbf{b}(x) dx$, a matrix of fixed coefficients.

Basis & Penalty example



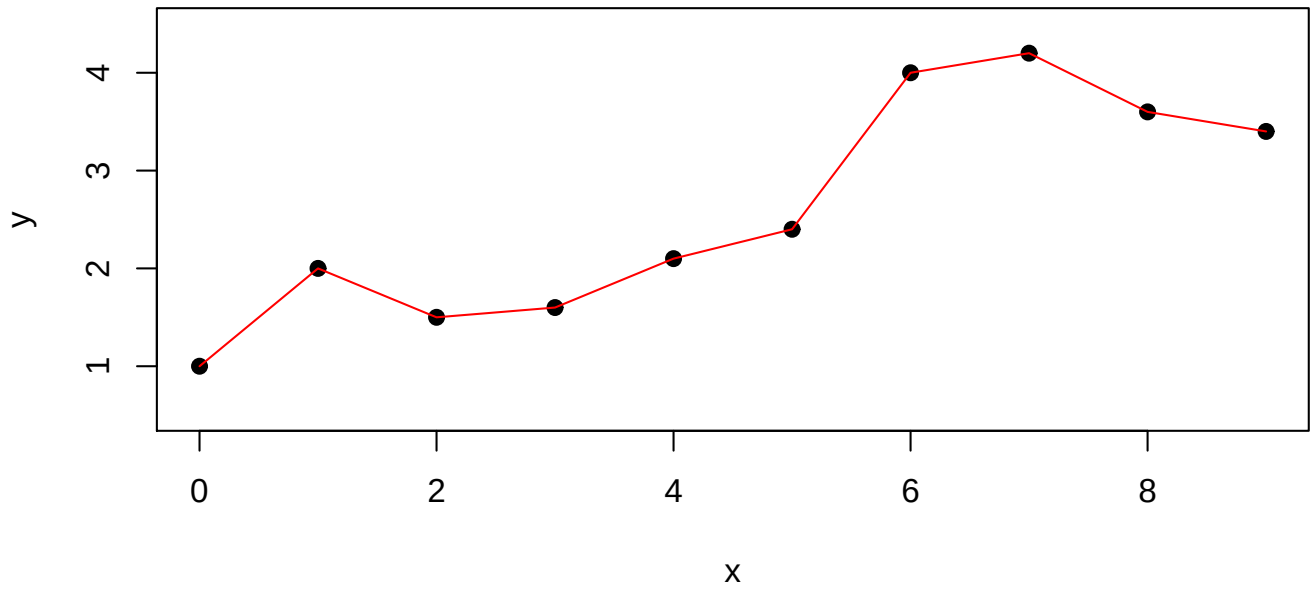






A simple basis

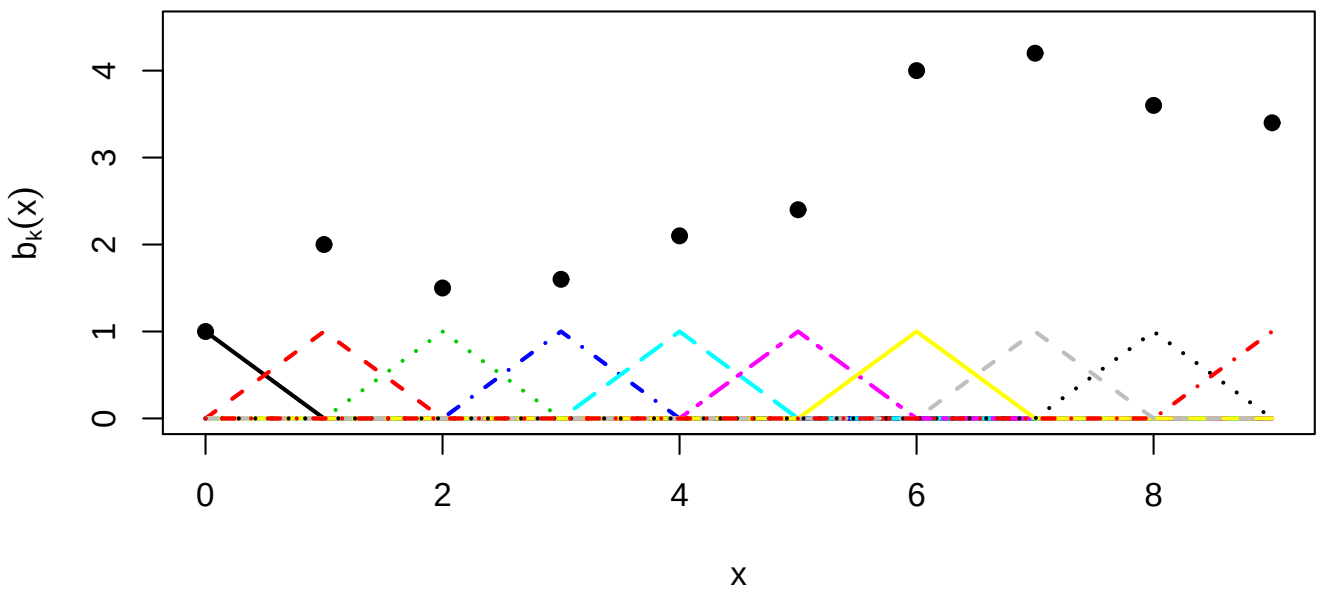
- Before considering smoothing in detail, consider linearly interpolating x^*, y^* data.



- The interpolant (red) can be written $f(x) = \sum_k \beta_k b_k(x)$, where the b_k are *tent funtions*:there is one per data point ($\bullet$).

The tent basis

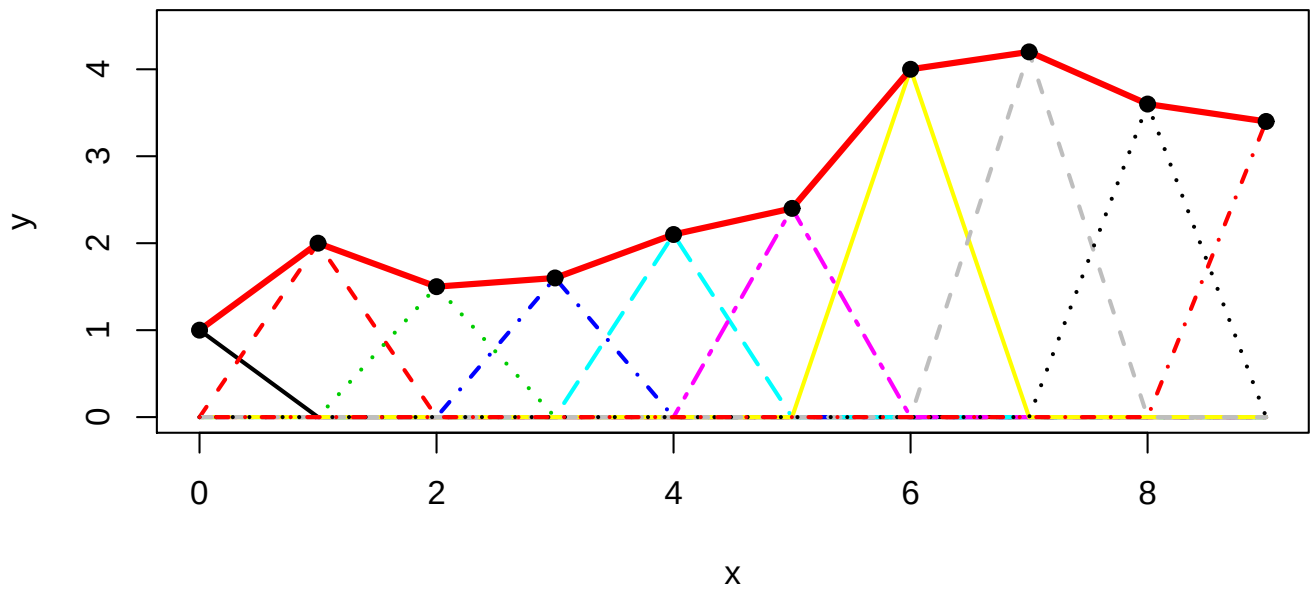
- The k^{th} tent function is 1 at x_k^* and descends linearly to zero at $x_{k\pm1}^*$. Elsewhere it is zero.
- The full set look like this...



- Under this definition of $b_k(x)$, $\beta_k = y_k^*$...

The interpolating basis

- So the interpolating function is represented by multiplying each tent function by its coefficient ($\beta_k = y_k^*$) and summing the results...



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

Prediction matrix

- Suppose that we have a series of points x_k^*, y_k^* to interpolate.
- The x_k^* values define the tent basis, and the y_k^* give the 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

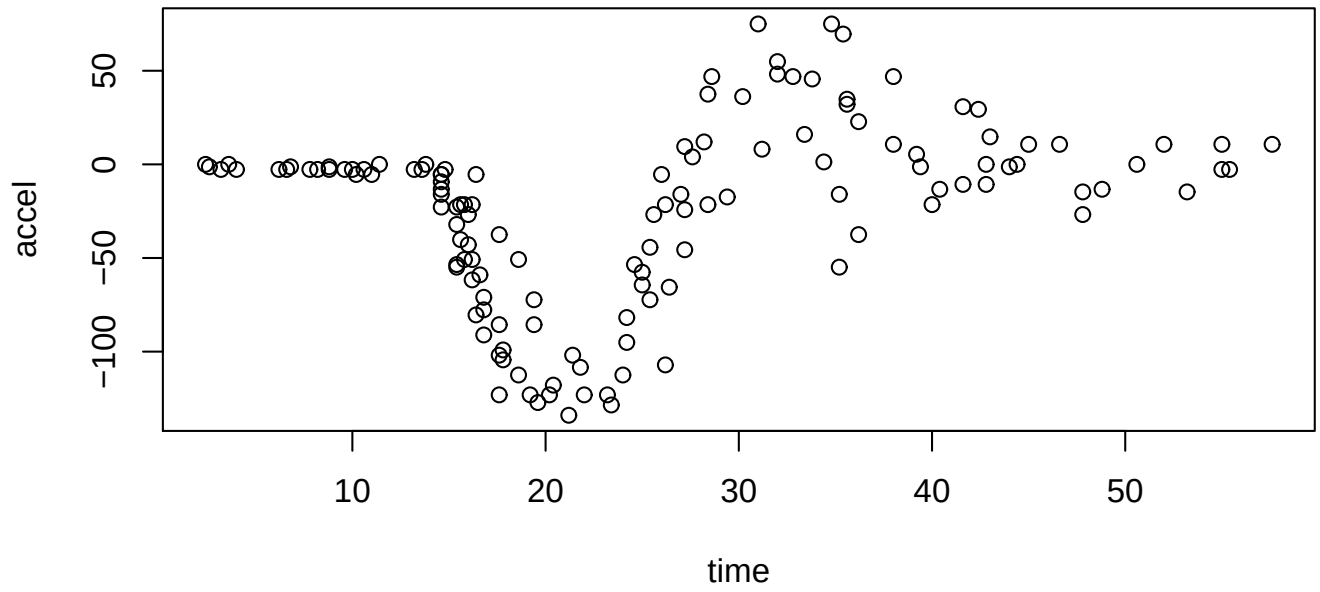
$$\mathbf{f} = \mathbf{X}\beta$$

where the *prediction matrix* is given by

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

Regression with a basis

- Suppose that we want to model these data



- One model is $a_i = f(t_i) + \epsilon_i$, where f is an unknown function.
- We could set $f(t) = \sum_k \beta_k b_k(t)$ where the b_k are tent functions, based on a set of t_k^* values spaced evenly through the range of observed times.

Regression model form

- Writing the model in vector from we have $\mathbf{a} = \mathbf{f} + \boldsymbol{\epsilon}$, where $\mathbf{f}^{\mathsf{T}} = [f(t_1), f(t_2), \ldots]$.
- Let $\mathbf{X}$ be a prediction matrix produced using the tent basis, so that $X_{ij} = b_j(t_i)$. The model becomes

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

- So we have a linear model, with model matrix $\mathbf{X}$.
- This can be estimated by standard linear modelling methods.
- Let's try it out in R.

A simple regression smoother in R

- First an R function for producing tent functions.

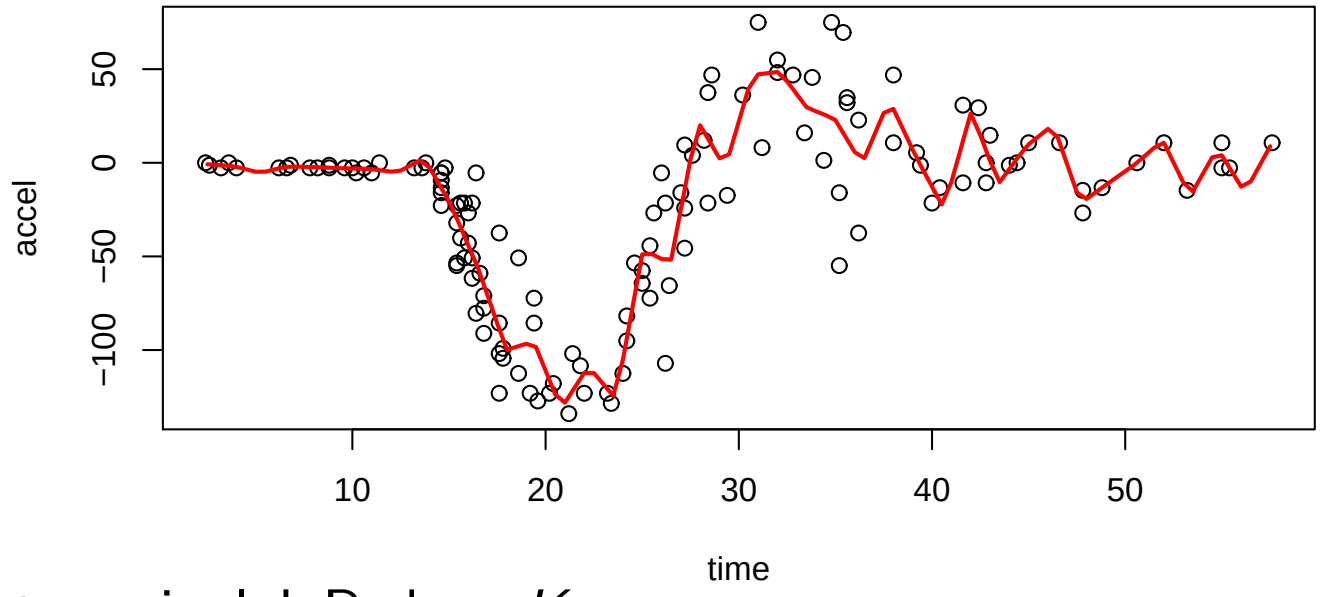
```
tf <- function(x,xk=seq(0,1,length=10),k=1) {
## generate kth tent function from set defined by xk
  yk <- xk*0;yk[k] <- 1
  approx(xk,yk,x)$y
}
```

- And now a function to use it for making prediction/model matrices.

```
tf.X <- function(x,xk=seq(0,1,length=10)) {
## tent function basis matrix given knot sequence xk
  nk <- length(xk); n <- length(x)
  X <- matrix(NA,n,nk)
  for (i in 1:nk) X[,i] <- tf(x,xk,i)
  X
}
```

Fitting the mcycle data

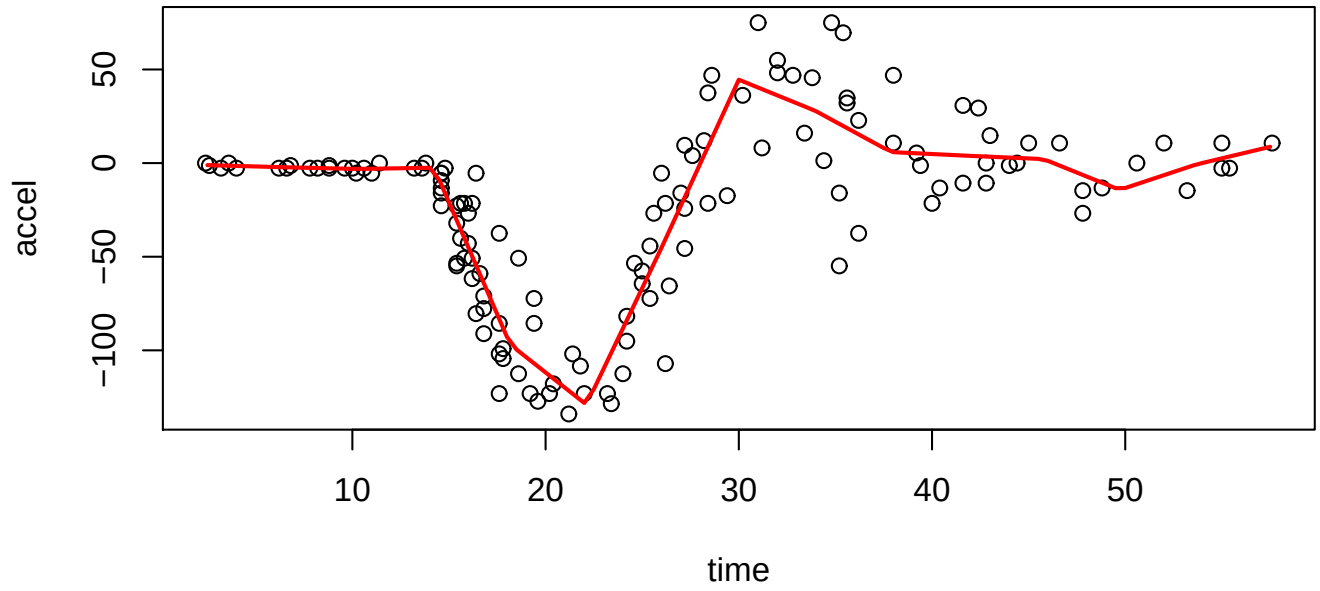
- ```
  K <- 40         ## basis dimension
  t0 <- min(mcycle$times);t1 <- max(mcycle$times)
  tk=seq(t0,t1,length=K)          ## knot sequence
  X <- tf.X(x=mcycle$times,xk=tk) ## model matrix
  b <- coef(lm(mcycle$accel~X-1)) ## fit model
  Xp <- tf.X(x=0:120/2,xk=tk)     ## prediction matrix
  plot(mcycle$times,mcycle$accel,ylab="accel",xlab="time")
  lines(0:120/2,Xp%*%b,col=2,lwd=2)
  ```



- Far too wiggly! Reduce K
  ```

Reducing K

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



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

Smoothing

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

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

Evaluating the penalty

- To get the penalty in convenient form, note that

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

  by definition of $\mathbf{D}$

- Hence

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

  by definition of $\mathbf{S}$.

Penalized fitting

- Now the penalized least squares estimates are

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

  smoothing parameter λ controls the fit-wiggliness tradeoff.

- For computational purposes this is re-written

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

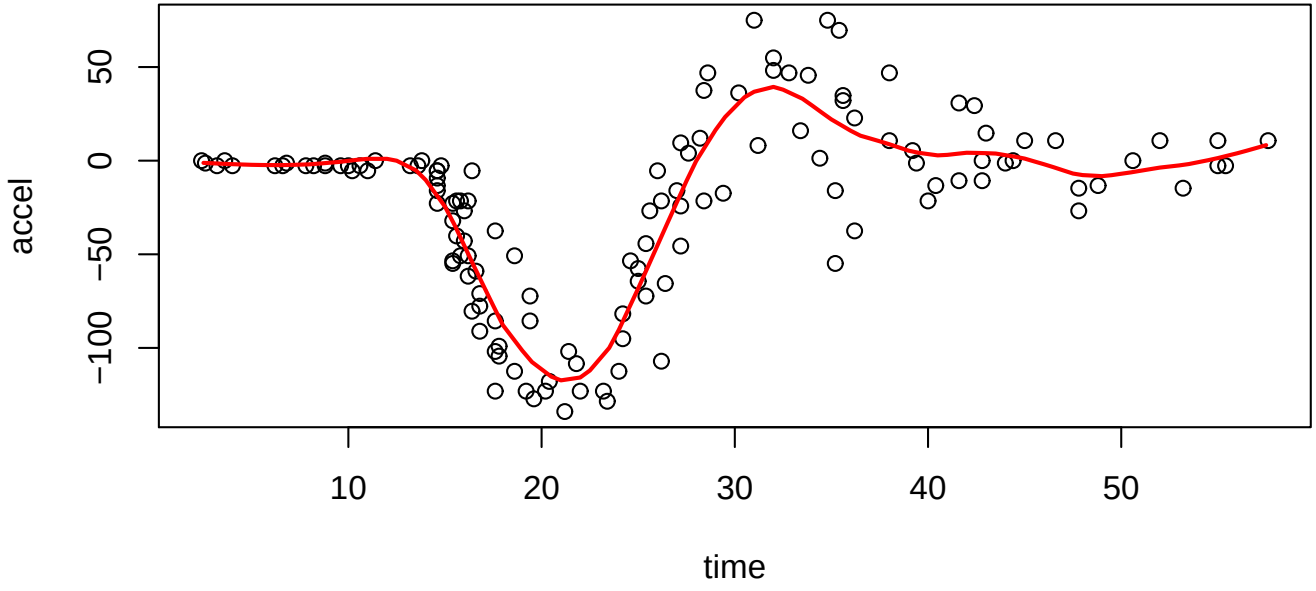
- In fact we can re-write again

$$\hat{\boldsymbol{\beta}} = \arg\min_{\beta} \left\| \begin{bmatrix} \mathbf{a} \\ \mathbf{0} \end{bmatrix} - \begin{bmatrix} \mathbf{X} \\ \sqrt{\lambda}\mathbf{D} \end{bmatrix} \boldsymbol{\beta} \right\|^2$$

- . . . least squares for an augmented linear model!

Penalized mcycle fit

```
D <- diff(diff(diag(K))) ## t(D)%*%D is penalty coef matrix
sp <- 2 ## square root smoothing parameter
XD <- rbind(X,D*sp)  ## augmented model matrix
y0 <- c(mcycle$accel,rep(0,nrow(D))) ## augmented data
b <- lm(y0~XD-1)     ## fit augmented model
plot(mcycle$times,mcycle$accel,ylab="accel",xlab="time")
lines(0:120/2,Xp%*%coef(b),col=2,lwd=2)
```



Diversion: avoiding the penalty

- The penalized fit is much nicer than the regression fit, but much theory will be needed to select λ and account for the fact that we penalized. Couldn't we avoid the penalty?
- To some extend the answer is yes!
- Given the penalty we can re-parameterize in such a way that the basis functions are orthogonal, and arranged in decreasing order of penalization.
- i.e. we can express the original smooth in terms of a set of basis functions than are orthonormal, and where the k^{th} is smoother than the $(k-1)^{\text{th}}$, according to the penalty.
- We can use such a basis unpenalized, and perform smoothness selection by deciding how many basis functions to drop, starting from the first. Formal hypothesis testing or AIC can be used to aid the decision.

The natural basis

- Let a smoother have model matrix $\mathbf{X}$ and penalty matrix $\mathbf{S}$.
- Form QR decomposition $\mathbf{X} = \mathbf{QR}$, followed by symmetric eigen-decompostion

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

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

Natural basis regression smoothing

- ```
  qrx <- qr(X)
  RD <- forwardsolve(t(qr.R(qrx)),t(D))
  es <- eigen(RD%*%t(RD),symmetric=TRUE)
  Xs <- qr.qy(qrx,rbind(es$vectors,matrix(0,nrow(X)-nrow(RD),nrow(RD))))
  b <- lm(mcycle$accel~Xs-1) ## unpenalized full fit
  summary(b)
  ...
  Coefficients:
        Estimate Std. Error t value Pr(>|t|)
  Xs1     0.8267    23.5508   0.035   0.9721
  Xs2    -8.2636    23.5508  -0.351   0.7265
  .         .          .       .        .
  Xs30   -5.8426    23.5508  -0.248   0.8046
  Xs31  -10.4568    23.5508  -0.444   0.6581
  Xs32 -126.5964    23.5508  -5.375 5.64e-07 ***
  Xs33  117.9980    23.5508   5.010 2.58e-06 ***
  Xs34  136.8142    23.5508   5.809 8.70e-08 ***
  Xs35  260.9345    23.5508  11.080  < 2e-16 ***
  .         .          .       .        .
  ```
- Since basis is orthogonal, I can legitimately look for the low to high p-value cut off.
- This suggests deleting the first 31 basis functions (out of 40).
  ```

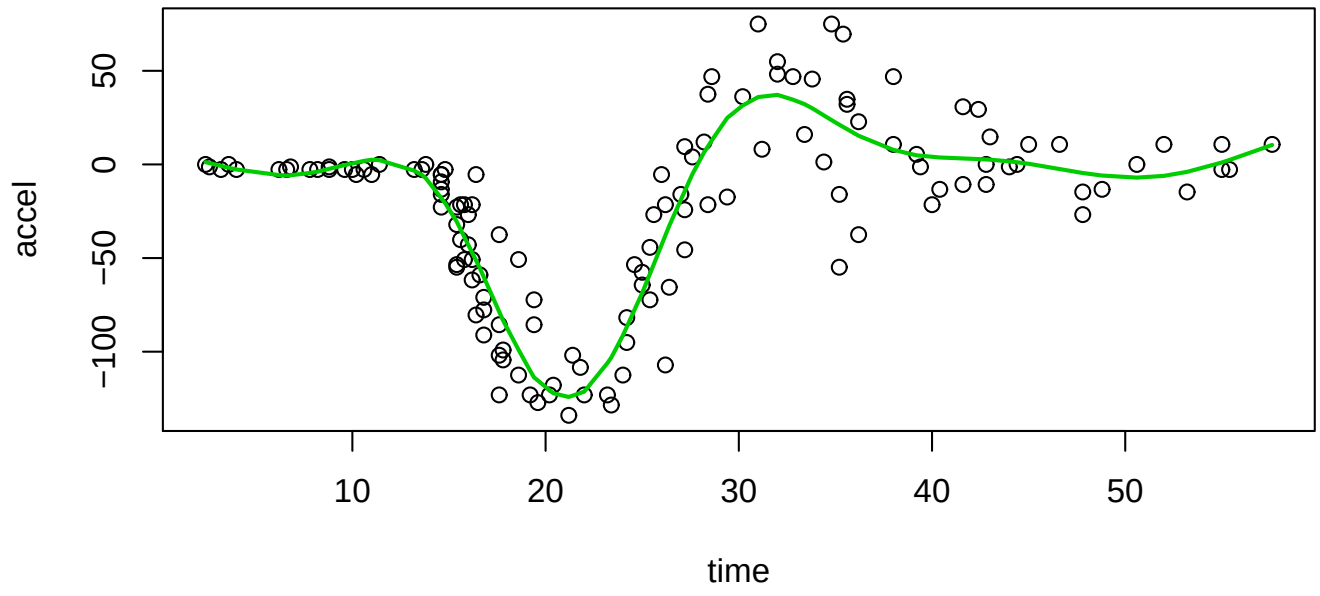
NP refit

- ```
  > b0 <- lm(mcycle$accel~Xs[,-c(1:31)]-1)
  > anova(b0,b)
  Analysis of Variance Table

  Model 1: mcycle$accel ~ Xs[, -c(1:31)] - 1
  Model 2: mcycle$accel ~ Xs - 1
    Res.Df   RSS Df Sum of Sq      F Pr(>F)
  1    124 62336
  2     93 51582 31     10755 0.6255 0.9304
  > AIC(b0,b)
     df      AIC
  b0 10 1215.381
  b  41 1252.194
  ```
- The reduced model b0 is clearly better.
- Since this is just an ordinary linear model it is perfectly legitimate to use AIC and F-ratio testing here.
  ```

The NP fit

- `plot(mcycle$times,mcycle$accel,ylab="accel",xlab="time")`
  `lines(mcycle$times,fitted(b0),col=3,lwd=2)`



- . . . a nice smooth fit without having to leave the inferential framework of standard linear models.
- This would work just as well in a GLM.
- However with multiple correlated predictors model selection would not be so straightforward.

Back to penalized fitting!

Effective Degrees of Freedom

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

Smoothing bias

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

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

- Smooths are baised!
- This is the price paid for reducing bias when you don't know the functional form in advance.
- The bias makes frequentist inference difficult (including bootstrapping!).

A Bayesian smoothing model

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

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

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

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

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

Using the $\beta|\mathbf{y}$ distribution

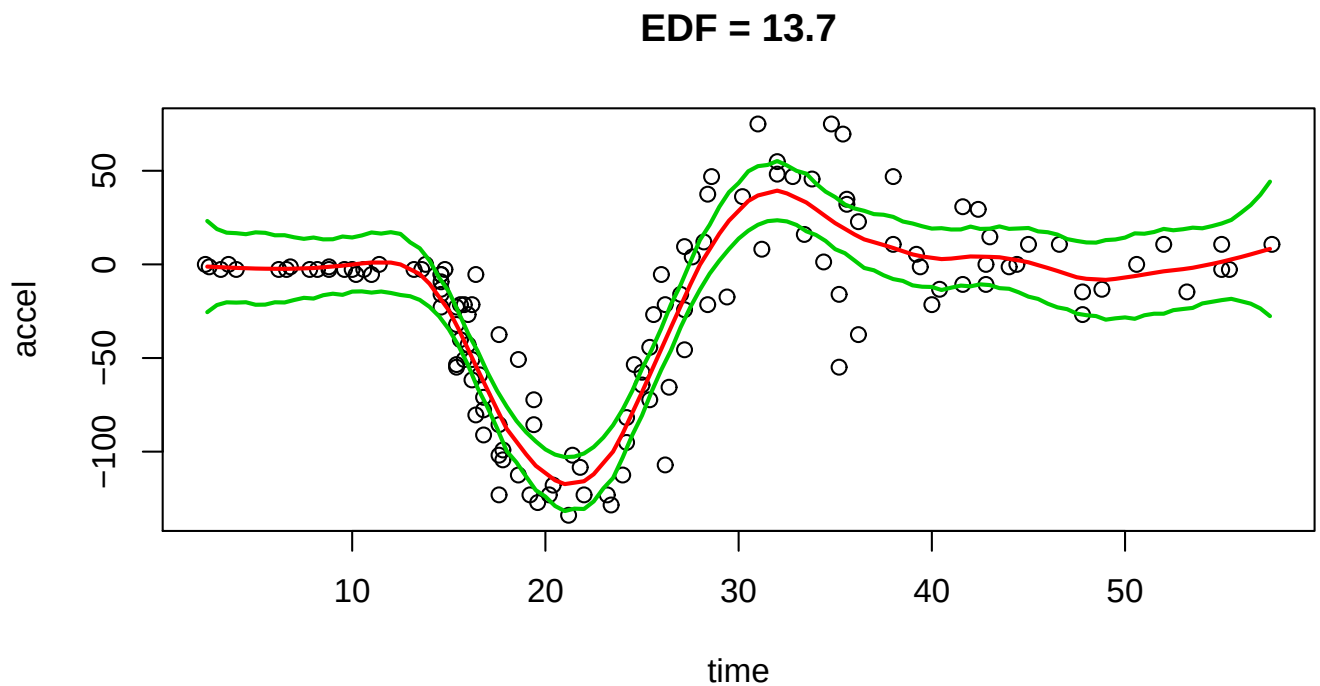
- It is very cheap to simulate from the distribution of $\beta|\mathbf{y}$, allowing easy Bayesian inference about any quantity derived from the model.
- Confidence/credible intervals for the smooth can be constructed without simulation. 'Across the function' they have very good frequentist properties. (Nychka, 1988)
- Nychka's idea
  - Construct an interval $C(x)$ so that if x is chosen randomly, $\Pr\{f(x) \in C(x)\} = .95$ (say).
  - Choosing x randomly turns bias$\{\hat{f}(x) - f(x)\}$ into a zero mean random variable.
  - Basing $C(x)$ on the sum of sampling variability and 'random bias' yields the Bayesian intervals (approximatly).
  - So by construction, the Bayesian intervals have close to nominal 'across the function' coverage.

P-values

- You might want to test whether the a smooth could be replaced by a constant (or other parametric term).
- This sits uncomfortably with the Bayesian approach, but there are several alternatives:
  1. Functions in the null space of the penalty are not penalized by smoothing, so under the null the parameter estimates are unbiased, and a purely frequentist approach can be taken, however the correct degrees of freedom to use is problematic.
  2. Given Nychka's result on the good frequentist properties of Bayesian intervals, it is possible to try and 'invert' the intervals to obtain p-values.
  3. If testing is really a key part of the analysis, it may be better to simply use unpenalized models, via the natural parameterization trick.

Adding a CI to mcycle fit

```
V <- solve(t(XD)%*%XD)        ## (X'X+\lambda S)^{-1}
ldF <- rowSums(V*(t(X)%*%X)) ## diag(F)
n <- nrow(X)
sig2 <- sum(residuals(b)[1:n]^2)/(n-sum(ldF))
V <- V*sig2  ## posterior covariance matrix
## get s.e. of predicted curve
f.se <- rowSums((Xp%*%V)*Xp)^.5 ## diag(Xp%*%V%*%t(Xp))
t <- 0:120/2; f <- Xp%*%coef(b)
lines(t,f+2*f.se,col=3,lwd=2)
lines(t,f-2*f.se,col=3,lwd=2)
title(paste("EDF =",round(sum(ldF),digits=2)))
```



Time to consider λ selection...