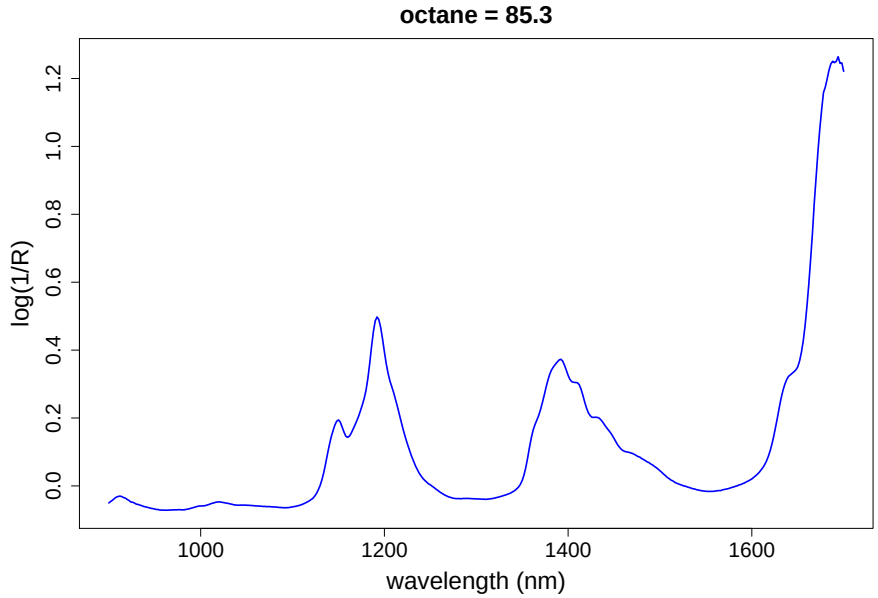
REML estimation of penalized GLMs

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

A model for octane rating

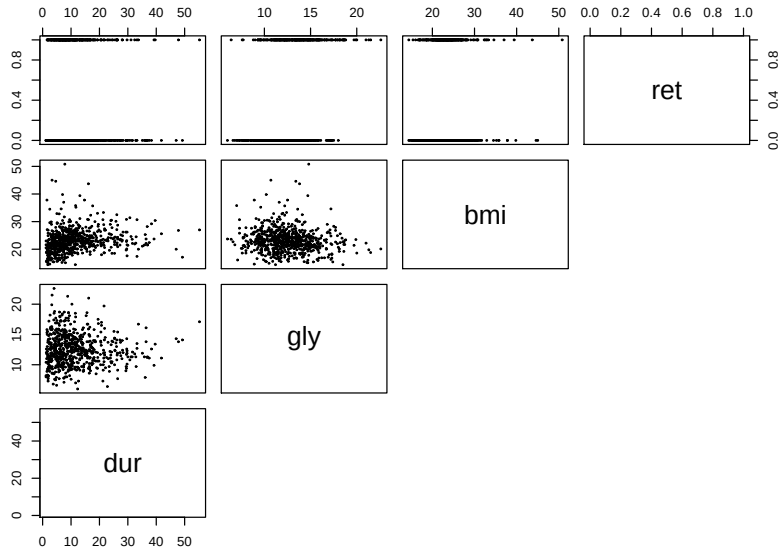


- Aim to predict octane rating from NIR spectrum, $g(\nu)$.
- A possible model is that $\texttt{octane}_i \sim$ Gamma,

$$\mathbb{E}(\texttt{octane}_i) = \int f(\nu) g_i(\nu) d\nu$$

where f is a function to be estimated.

Diabetic retinopathy



- Possible model. . . $\texttt{ret} \sim$ Bernoulli,

$$\text{logit}\{\mathbb{E}(\texttt{ret})\} = f_1(\texttt{dur}) + f_2(\texttt{bmi}) + f_3(\texttt{gly}) + f_4(\texttt{dur}, \texttt{bmi}) + f_5(\texttt{dur}, \texttt{gly}) + f_6(\texttt{gly}, \texttt{bmi})$$

Penalized GLMs

- Examples are special cases of the model

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

- y_i is an exponential family distributed univariate response, $\mathbf{A}$ is a known parametric model matrix, with corresponding coefficients $\boldsymbol{\theta}$, some of which may be random, f_j is a smooth function of one or more predictor variables, and L_{ij} is a known linear functional.
- Such models include GAMs, GAMMs, varying coefficient models and signal regression models.

Practical representation

- To make these models practical, it helps to represent each f_j using a linear basis expansion,

$$f_j(x) = \sum_k^K \gamma_k b_k(x)$$

  where the coefficients, γ_k, are unknown, the basis functions $b_k(x)$ are chosen for convenience, and K is set large enough to avoid underfitting.

- Then the model becomes

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

  where $\mathbf{X}$ is a model matrix containing $\mathbf{A}$ and evaluated versions of $L_{ij}b_k$, and $\boldsymbol{\beta}$ contains $\boldsymbol{\theta}$ and the f_j coefs.

Estimation

- To avoid overfit, estimation is by penalized MLE.

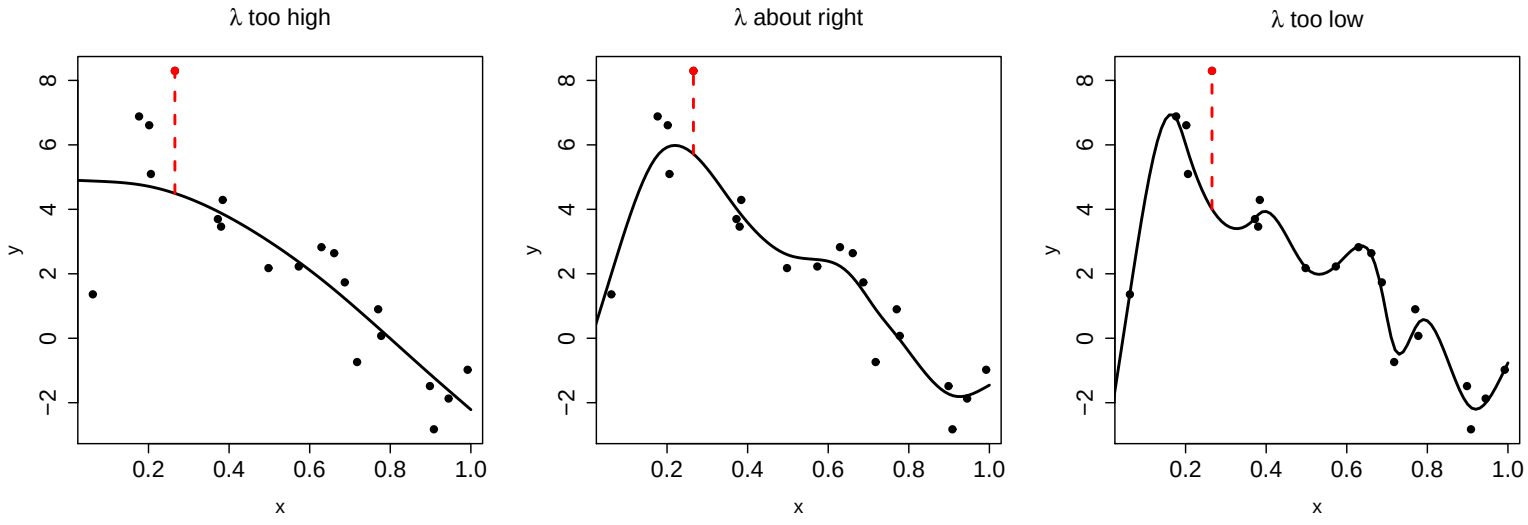
$$\text{minimise} \;\; D(\beta) + \sum \lambda_k \beta^{\mathrm{T}} \mathbf{S}_k \beta \;\; \text{w.r.t.} \;\; \beta$$

  D is model deviance, and the $\lambda_k \beta^{\mathrm{T}} \mathbf{S}_k \beta$ penalize overfit.
- Typically $\beta^{\mathrm{T}} \mathbf{S}_k \beta$ measures 'wiggliness' of f_j, or are random effect shrinkage factors.
- Smoothing parameters λ_k control smoothness of the f_j, or random effect variances.
- There is a Bayesian/ mixed model interpretation: imposition of the penalties is equivalent to using a prior

$$\beta \sim N\left\{\mathbf{0}, \left(\sum \lambda_k \mathbf{S}_k\right)^{-}\right\}.$$

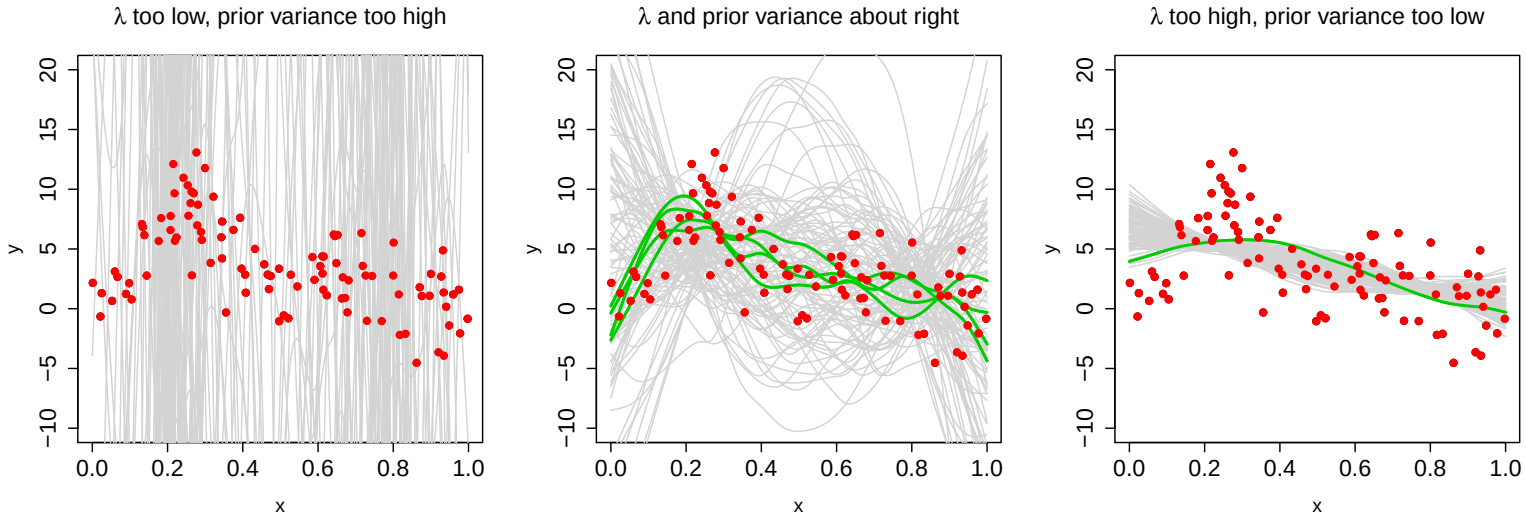
- *Given* $\boldsymbol{\lambda}$, $\hat{\beta}$ is found by penalized IRLS. What about $\boldsymbol{\lambda}$...

How to estimate λ: cross validation



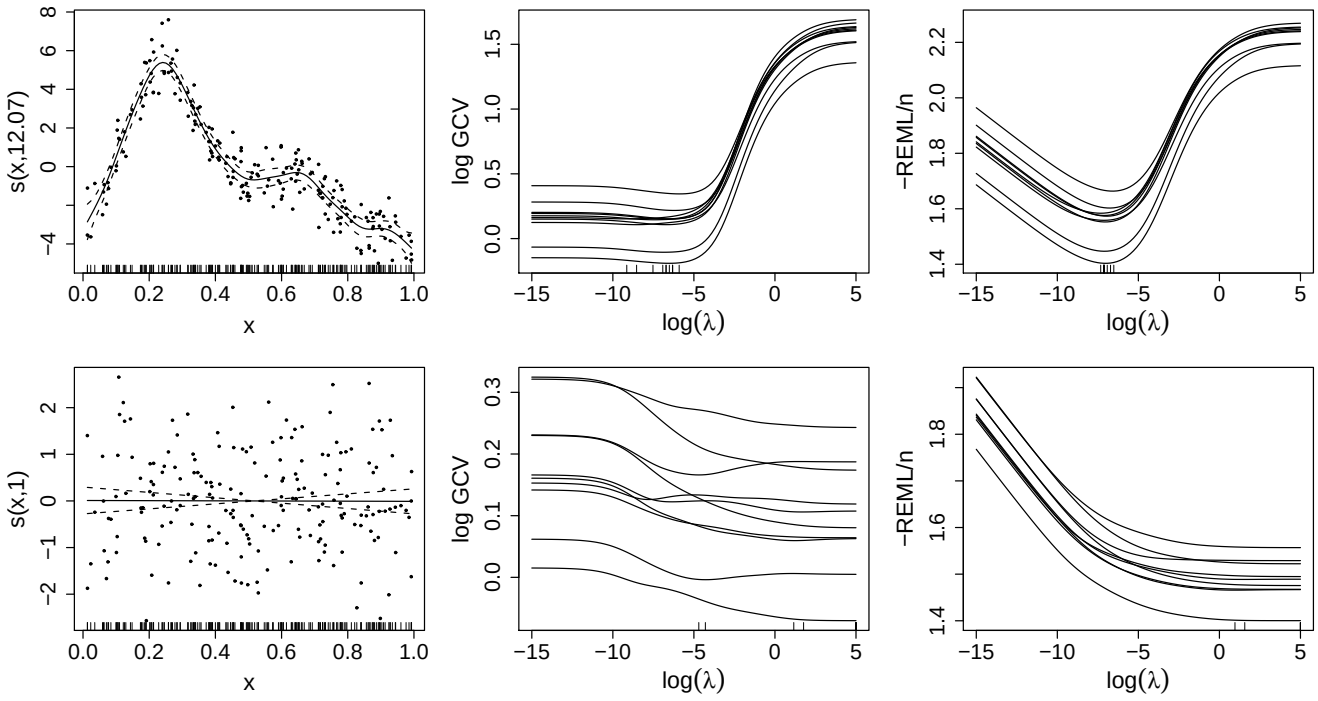
- Leave out one datum at a time, fit model to rest, and find squared error in predicting the left out one.
- Average these squared errors over all data. Find λ to minimize this MSE.
- Invariant version is GCV.

How to estimate λ: marginal likelihood



- Draw β from prior implied by λ. Find average value of likelihood for these draws.
- Choose λ to maximize this average likelihood.
- Formally, maximize $\int f(\mathbf{y}, \beta) d\beta$ w.r.t. λ.

GCV vs. REML comparison



- Asymptotic MSE better for GCV, but GCV $\hat{\lambda}$ converges much more slowly than REML version.
- GCV has greater tendency to multiple minima and severe undersmoothing.

The upshot

- REML or ML based smoothness selection is often preferable to GCV or AIC in practice.
- But the computational methods for REML/ML are substantially less robust than the GCV/AIC equivalents.
- This requires some explanation. . .

Computing $\hat{\boldsymbol{\beta}}|\lambda$ and then $\boldsymbol{\lambda}$

- Given $\boldsymbol{\lambda}$, iterate a PIRLS scheme to convergence...
  1. Form $z_i = \mathbf{X}_i\hat{\boldsymbol{\beta}} + (y_i - \hat{\mu}_i)g'(\hat{\mu}_i)/\alpha_i$ and $w_i = \alpha_i/\{V(\hat{\mu}_i)g'(\hat{\mu}_i)^2\}$.
  2. Minimize $\sum w_i(z_i - \mathbf{X}_i\boldsymbol{\beta})^2 + \sum_k \lambda_k\boldsymbol{\beta}^{\mathrm{T}}\mathbf{S}_k\boldsymbol{\beta}$ w.r.t. $\boldsymbol{\beta}$ to get new estimate of $\hat{\boldsymbol{\beta}}$.

  $\alpha_i = 1 + (y_i - \mu_i)(V'/V_i + g_i''/g_i')$ for Newton or $\alpha_i = 1$ for Fisher scoring.
- Denote the resulting penalized MLE by$\hat{\boldsymbol{\beta}}_\lambda$.
- Two options for $\boldsymbol{\lambda}$ estimate computation...
  1. Single iteration: insert a GCV or REML based $\boldsymbol{\lambda}$ estimation step into the PIRLS.
  2. Nested iteration: let REML or GCV score depend on $\boldsymbol{\beta}$ only through $\hat{\boldsymbol{\beta}}_\lambda$, so each iteratively proposed $\boldsymbol{\lambda}$ value requires a full PIRLS to find $\hat{\boldsymbol{\beta}}_\lambda$.

λ computations compared

- Single iteration (which includes PQL) need not converge to a fixed $\hat{\boldsymbol{\beta}}_{\hat{\lambda}}$.
- Nested iteration is much more tedious to implement, but converges to a fixed $\hat{\boldsymbol{\beta}}_{\hat{\lambda}}$.
- Nested iteration was only available for GCV etc, not REML etc.
- Aim is to rectify this, to make REML based penalized GLM estimation as routine and reliable as parametric GLM estimation.

Laplace approximate REML

- Let $f(\mathbf{y}, \beta)$ be the joint density of the data $\mathbf{y}$ and β.
- To get restricted likelihood, integrate out β approximately...
- Replace $f(\mathbf{y}, \beta)$ by exponential of truncated Taylor expansion of $\log f$, about $\hat{\beta}_\lambda$.
- Approximation is close enough to a Normal density to be integratable. Hooray!
- Twice log approximate restricted likelihood is

$$2l_r = 2l(\hat{\beta}) + \log|\mathbf{S}/\phi|_+ - \hat{\beta}^{\mathrm{T}}\mathbf{S}\hat{\beta}/\phi - \log|\mathbf{H} + \mathbf{S}/\phi| + M_p \log(2\pi).$$

$\mathbf{H}$ is Hessian of $-\log f$, and subscript dropped on $\hat{\beta}_\lambda$.

Computing $\hat{\boldsymbol{\lambda}}$

- l_r maximised w.r.t. $\log \boldsymbol{\lambda}$ by Newton's method.
- Each trial $\log \boldsymbol{\lambda}$ will require a full PIRLS to get $\hat{\boldsymbol{\beta}}_\lambda$.
- Some computational considerations...
  1. $\log |\mathbf{S}/\phi|_+$, where $\mathbf{S} = \sum \lambda_k \mathbf{S}_k$, is very awkward.
  2. $\mathbf{H} = \mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X}/\phi$ if Newton PIRLS is used ($\mathbf{W} = \mathrm{diag}(w_i)$).
  3. Need derivatives of $\hat{\boldsymbol{\beta}}_\lambda$ w.r.t. $\log \boldsymbol{\lambda}$, to get derivatives of l_r for Newton steps. Implicit differentiation gives these easily, if Newton based PIRLS is used.
  4. **But** negative weights can occur for Newton PIRLS, making stable computation difficult.

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,]    1   -1    0    0    0
[2,]   -1    2   -1    0    0
[3,]    0   -1    1    0    0
[4,]    0    0    0    0    0
[5,]    0    0    0    0    0
     [,1] [,2] [,3] [,4] [,5]
[1,]    0    0    0    0    0
[2,]    0    0    0    0    0
[3,]    0    0    1    0    0
[4,]    0    0    0    1    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 $\mathbf{U}\mathbf{D}\mathbf{U}^{\mathrm{T}} = \mathbf{S}_1$. Then $|\mathbf{S}_1 + \lambda\mathbf{S}_2| = |\mathbf{D} + \lambda\mathbf{U}^{\mathrm{T}}\mathbf{S}_2\mathbf{U}|$. 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 $\mathbf{S}$).
- The reparameterization used here also yields the most stable computation of $\hat{\boldsymbol{\beta}}$ and $|\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X} + \mathbf{S}|$, so is always used (recomputed for each trial $\boldsymbol{\lambda}$).

Newton penalized iteratively reweighted least squares

- Newton PIRLS involves minimising

$$\mathcal{S} = \sum w_i (z_i - \mathbf{X}_i \beta)^2 + \beta^{\mathrm{T}} \mathbf{S} \beta$$

  where some w_i can be negative.
- Formal solution $(\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X} + \mathbf{S})\hat{\beta} = \mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{z}$, where $\mathbf{W} = \mathrm{diag}(w_i)$, is far too badly-conditioned to use here.
- For positive w_i, use QR decomposition method on

$$\text{minimise } \mathcal{S} = \left\| \sqrt{\mathbf{W}} \left(\begin{bmatrix} \mathbf{z} \\ \mathbf{0} \end{bmatrix} - \begin{bmatrix} \mathbf{X} \\ \sqrt{\mathbf{S}} \end{bmatrix} \beta \right) \right\|^2 \text{ w.r.t. } \beta$$

- If some $w_i < 0$ use QR on

$$\text{minimise } \mathcal{S} = \left\| \sqrt{\mathrm{abs}\mathbf{W}} \left(\begin{bmatrix} \mathbf{z} \\ \mathbf{0} \end{bmatrix} - \begin{bmatrix} \mathbf{X} \\ \sqrt{\mathbf{S}} \end{bmatrix} \beta \right) \right\|^2 + \text{correction}$$

Derivatives of $\hat{\boldsymbol{\beta}}$ w.r.t. $\boldsymbol{\rho} = \log \boldsymbol{\lambda}$

- Consider single λ. $\hat{\boldsymbol{\beta}}_\lambda$ is the solution to

$$\left.\frac{\mathrm{d}D}{\mathrm{d}\boldsymbol{\beta}}\right|_{\hat{\boldsymbol{\beta}}_\lambda} + 2\lambda \mathbf{S}\hat{\boldsymbol{\beta}} = \mathbf{0}$$

- Differentiating w.r.t. $\rho = \log \lambda$ gives

$$\left.\frac{\mathrm{d}^2 D}{\mathrm{d}\boldsymbol{\beta}\mathrm{d}\boldsymbol{\beta}^{\mathrm{T}}}\right|_{\hat{\boldsymbol{\beta}}_\lambda} \frac{\mathrm{d}\hat{\boldsymbol{\beta}}_\lambda}{\mathrm{d}\rho} + 2\lambda \mathbf{S}\hat{\boldsymbol{\beta}} + 2\lambda \mathbf{S}\frac{\mathrm{d}\hat{\boldsymbol{\beta}}}{\mathrm{d}\rho} = \mathbf{0}$$

- Re-arranging yields...

$$\frac{\mathrm{d}\hat{\boldsymbol{\beta}}}{\mathrm{d}\rho} = -2\lambda \left(\frac{\mathrm{d}^2 D}{\mathrm{d}\boldsymbol{\beta}\mathrm{d}\boldsymbol{\beta}^{\mathrm{T}}} + 2\lambda \mathbf{S} \right)^{-1} \mathbf{S}\hat{\boldsymbol{\beta}}$$

...which is computable from left-overs!

Key points in summary

- Find $\rho = \log(\boldsymbol{\lambda})$ by Newton's method optimization of the log restricted likelihood, l_r.
- For each trial $\boldsymbol{\rho}$ vector ...
  1. Re-parameterize so that $\log|\mathbf{S}|_+$ is computable.
  2. Solve for $\hat{\boldsymbol{\beta}}_\lambda$ by stable Newton PIRLS.
  3. Get the derivatives of $\hat{\boldsymbol{\beta}}_\lambda$ by implicit differentiation.
  4. Slog out the derivatives of l_r itself, from which an update for $\hat{\boldsymbol{\rho}}$ can be computed.
- Full details are exceptionally tedious, but these are the main points.

Implementation: `mgcv`

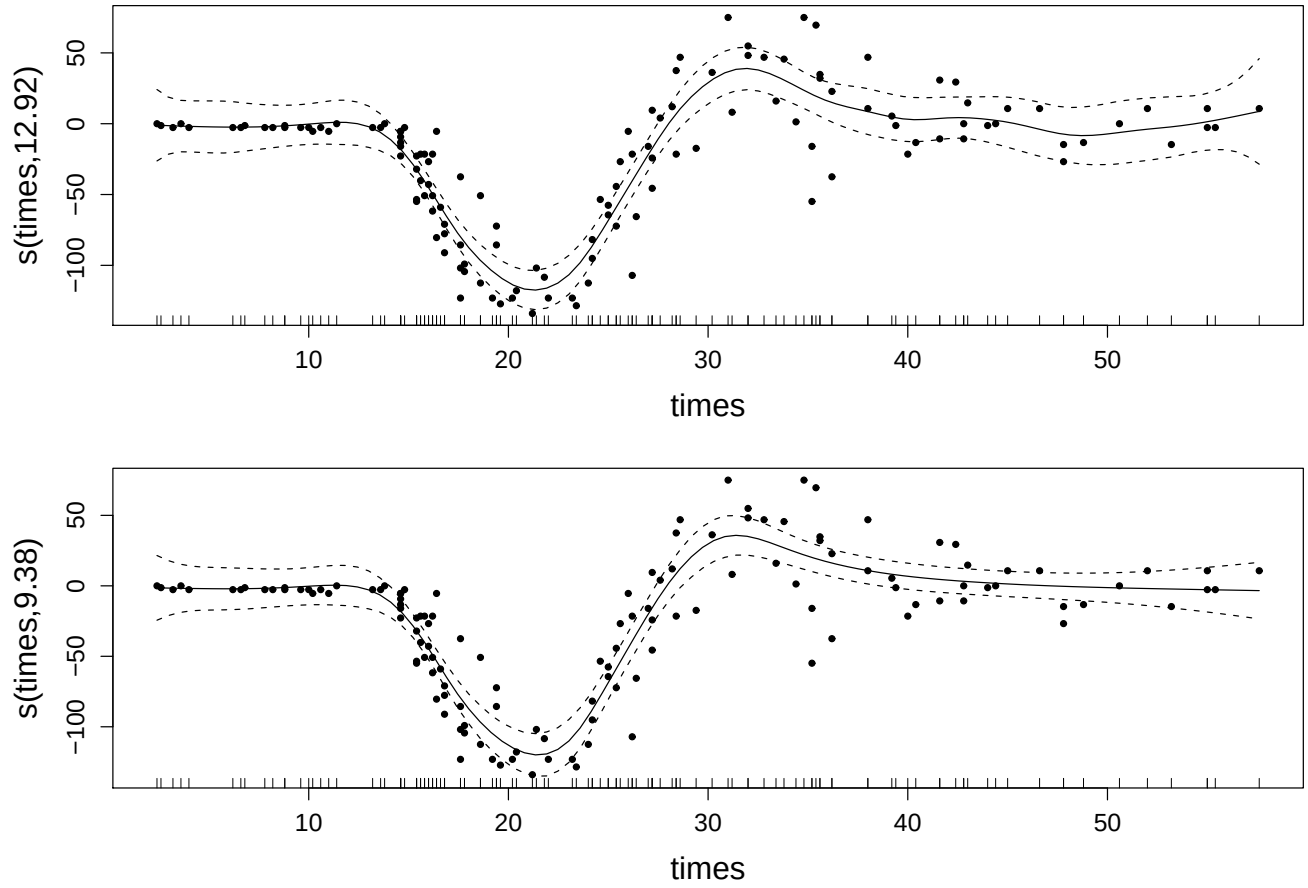
- R package `mgcv` implements this REML based estimation method.
- Consider example GAMM. $y_i \sim$ Poi,

$$\log \mathbb{E}(y_i) = f_1(x_i)t_i + f_2(v_i, w_i) + \gamma_k, \quad \gamma_k \sim N(0, \sigma_g^2)$$

  if observation i from level k of factor variable, `group`.
- `mgcv::gam` can fit this

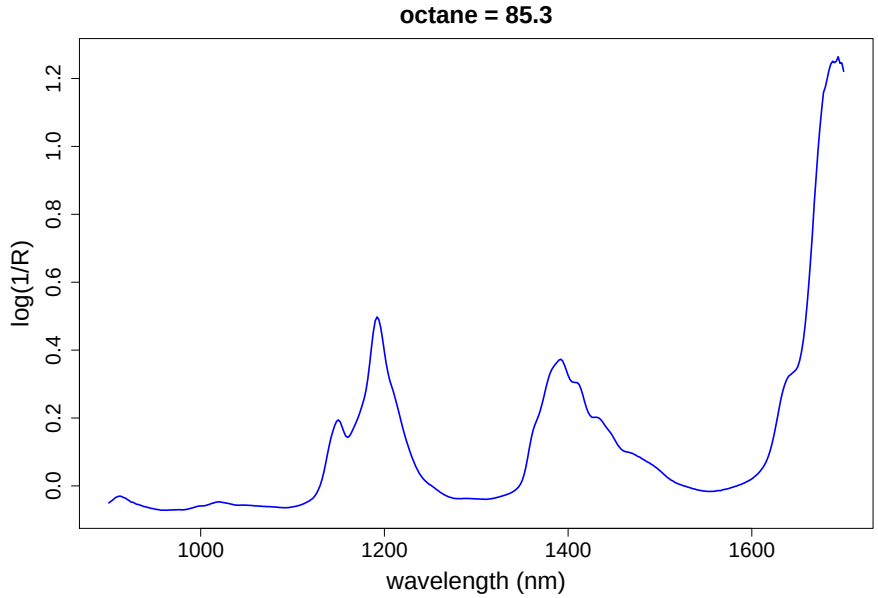
```
gam(y~s(x,by=t)+s(v,w)+s(group,bs="re"),
    family=poisson,method="REML")
```

Example: adaptive smoothing



- Top is normal spline smooth.
- Lower is an adaptive spline smooth, where penalty is split up, so that degree of smoothness can vary with time.

Example: A model for octane rating

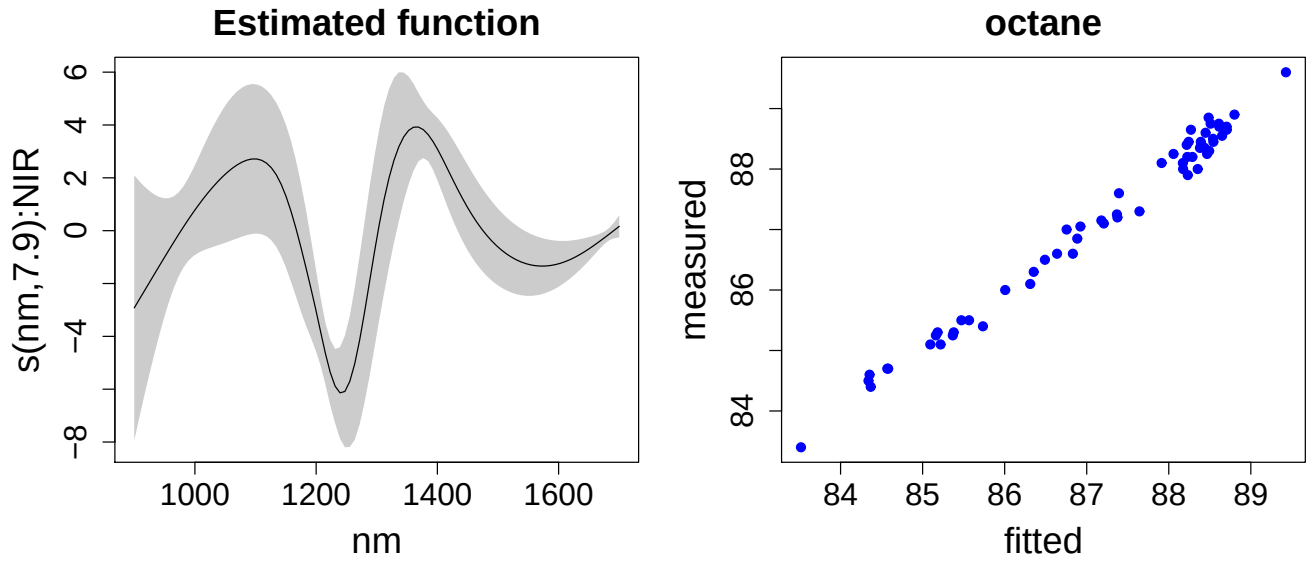


- Aim to predict octane rating from NIR spectrum, $g(\nu)$.
- A possible model is that $\texttt{octane}_i \sim$ Gamma,

$$\log \mathbb{E}(\texttt{octane}_i) = \int f(\nu) g_i(\nu) d\nu$$

where f is a function to be estimated.

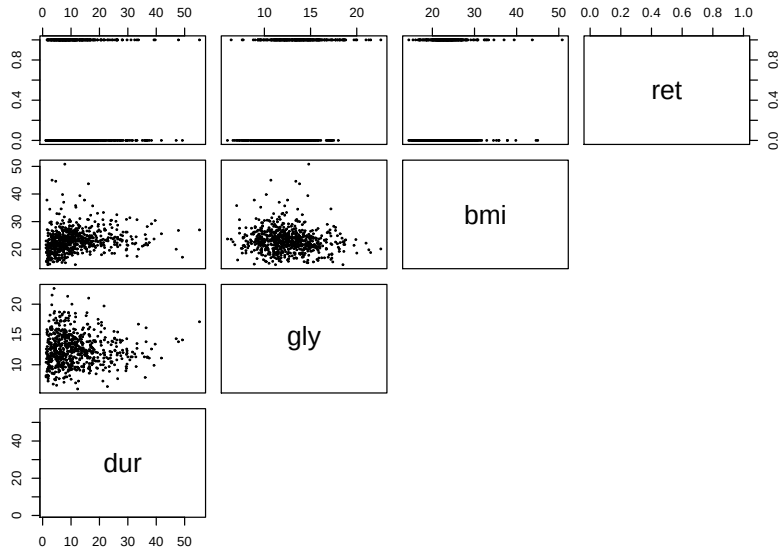
Octane model estimates



- If `nm` is a matrix each row of which contains the spectral wavelengths and `NIR` is a corresponding matrix of spectral measurements, then model is fit by

```
gam(octane~s(nm,by=NIR,bs="ad"),
    family=Gamma(log))
```

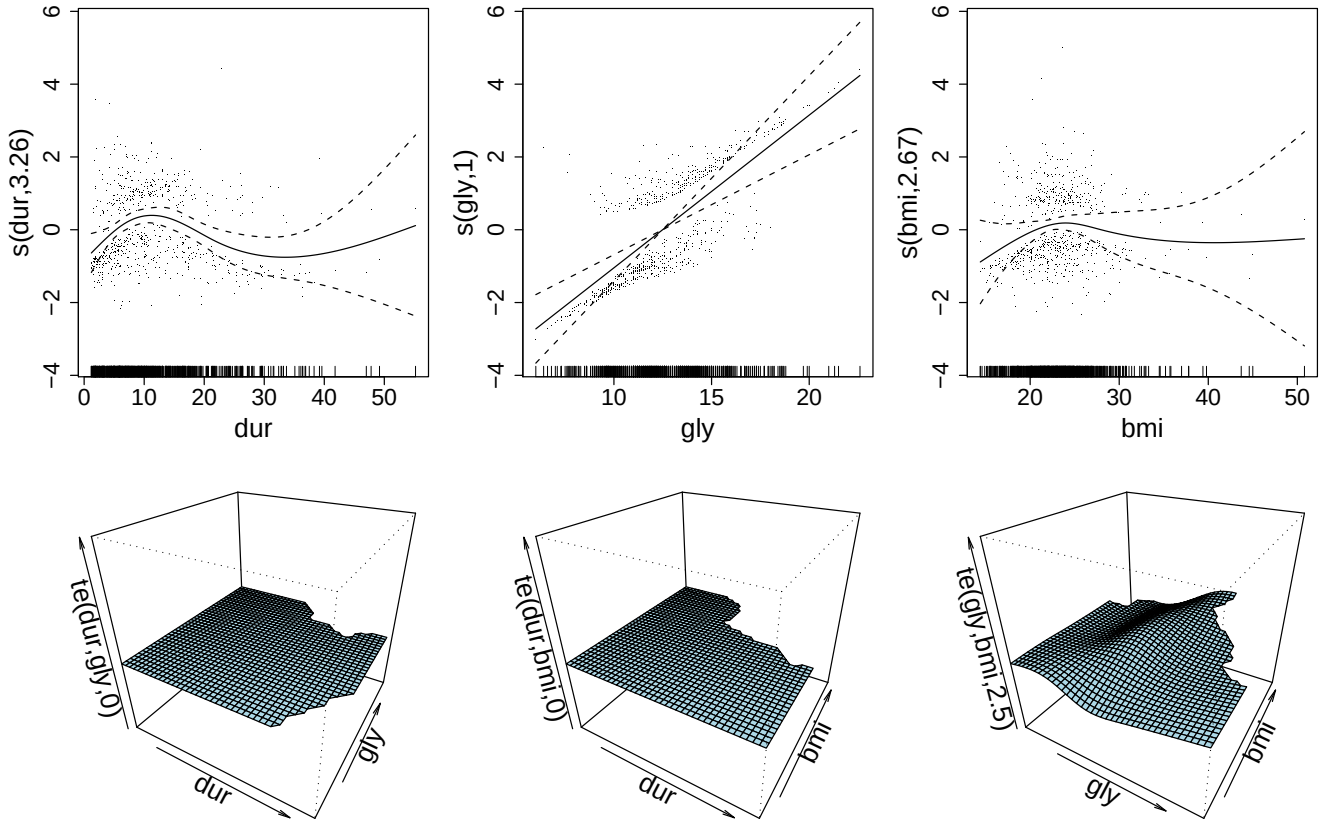
Example: Diabetic retinopathy



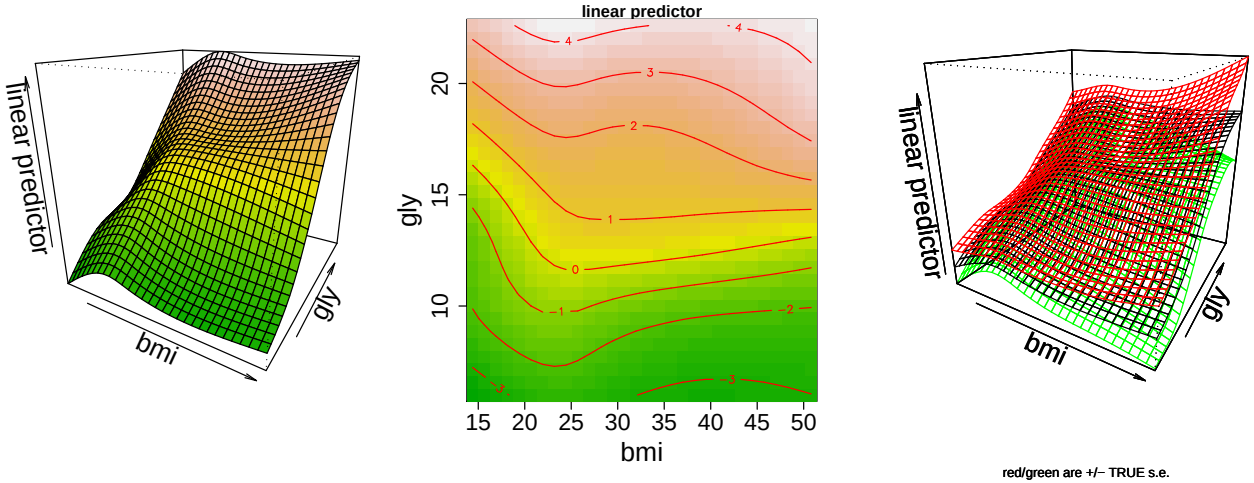
- Possible model. . . $\texttt{ret} \sim$ Bernoulli,

$$\text{logit}\{\mathbb{E}(\texttt{ret})\} = f_1(\texttt{dur}) + f_2(\texttt{bmi}) + f_3(\texttt{gly}) + f_4(\texttt{dur}, \texttt{bmi}) + f_5(\texttt{dur}, \texttt{gly}) + f_6(\texttt{gly}, \texttt{bmi})$$

Retinopathy estimates



Retinopathy estimates: interaction



Advert

- Wood, S.N. (2011) Fast stable REML and ML estimation of semiparametric GLMs. JRSSB 73(1):1-34