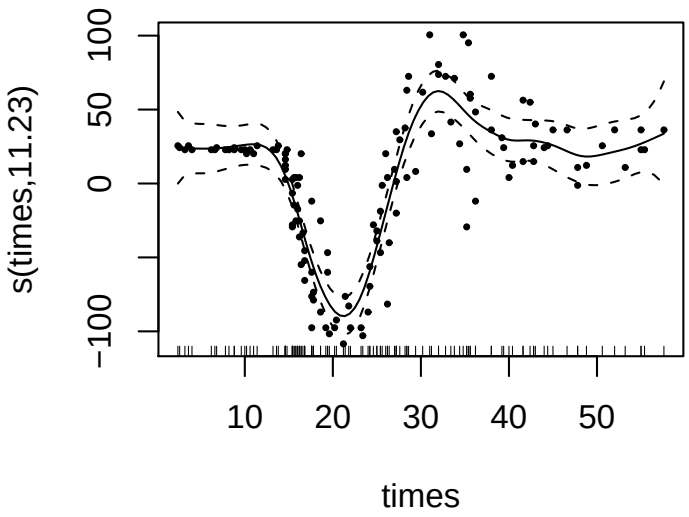
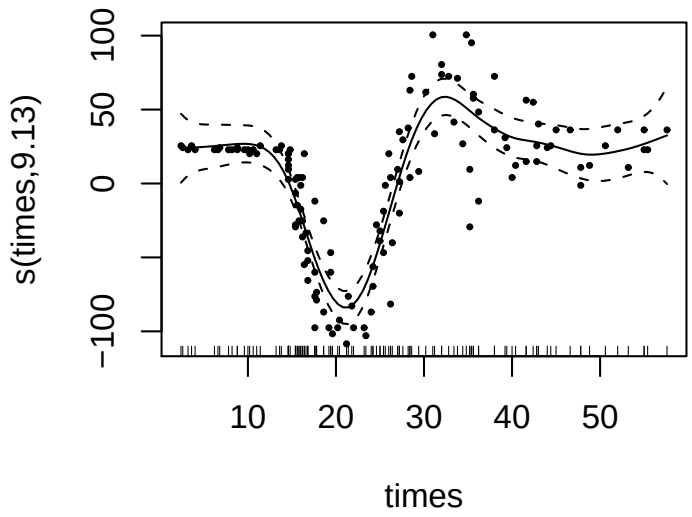
More advanced use of mgcv

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

Fine control of smoothness: gamma

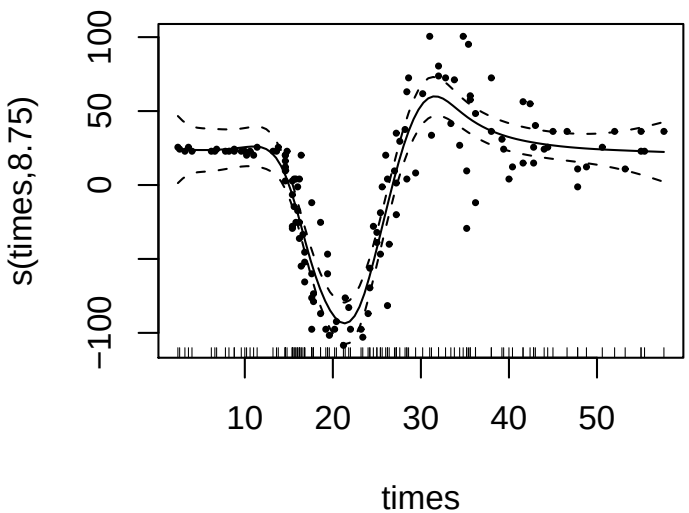
- Suppose that we fit a model but a component is too wiggly.
- For GCV/AIC we can increase the 'cost' of degrees of freedom to be more BIC like. i.e. multiply the EDF by $\log(n)/2$...
- ```
  b <- gam(accel~s(times,k=40),data=mcycle)
  plot(b,residuals=TRUE,pch=20,cex=.5) ## Too wiggly!!
  gamma <- log(nrow(mcycle))/2 ## BIC like penalization
  ## 'gamma' multiplies the EDF in GCV or AIC score...
  b <- gam(accel~s(times,k=40),data=mcycle,gamma=gamma)
  plot(b,residuals=TRUE,pch=20,cex=.5)
  ```

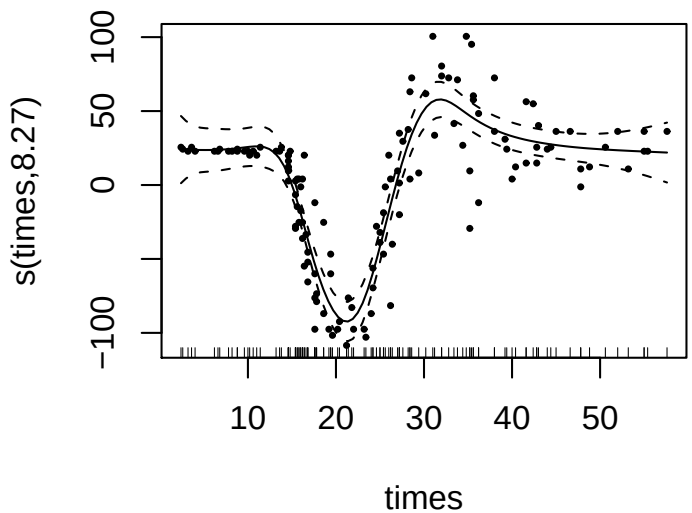



  ```

Fine control of smoothness: sp

- Alternatively the sp argument to gam (or to individual s terms) can be used to fix some smoothing parameters.
- ```
  ## try adaptive...
  b <- gam(accel~s(times,bs="ad"),data=mcycle)
  plot(b,residuals=TRUE,pch=20,cex=.5) ## not adaptive enough?!
  ## Decrease some elements of sp. sp[i] < 0 => estimate sp[i]...
  b <- gam(accel~s(times,bs="ad"),data=mcycle,sp=c(-1,1e-5,1e-5,-1,-1))
  plot(b,residuals=TRUE,pch=20,cex=.5) ## hmm!
  ```




  ```

Posterior inference

- Suppose that we want to make inferences about some non-linear functional of the model. Simulation from the distribution $\boldsymbol{\beta}|\mathbf{y}$ is the answer.
- For example we might be interested in the trough to peak difference in the `mcycle` model just fitted.

```
pd <- data.frame(times=seq(10,40,length=1000))
Xp <- predict(b,pd,type="lpmatrix") ## map coefs to fitted curves
beta <- coef(b);Vb <- vcov(b) ## posterior mean and cov of coefs
n <- 10000
br <- mvrnorm(n,beta,Vb) ## simulate n rep coef vectors from post.
a.range <- rep(NA,n)
for (i in 1:n) { ## loop to get trough to peak diff for each sim
  pred.a <- Xp%*%br[i,]  ## curve for this replicate
  a.range[i] <- max(pred.a)-min(pred.a) ## range for this curve
}
quantile(a.range,c(.025,.975)) ## get 95% CI
    2.5%    97.5%
134.1007 170.0738
```

Posterior simulation versus bootstrapping

- Posterior simulation is very quick.
- It is *much* more efficient than bootstrapping.
- In any case bootstrapping is problematic...
  1. For parametric bootstrapping the smoothing bias causes problems, the model simulated from is biased and the fits to the samples will be yet more biased.
  2. For non-parametric 'case-resampling' the presence of replicate copies of the same data causes undersmoothing, especially with GCV based smoothness selection.
- An objection to posterior simulation is that we condition on $\hat{\boldsymbol{\lambda}}$.
- This is fixable, by simulation of replicate $\boldsymbol{\lambda}$ vectors, and then simulating $\boldsymbol{\beta}$ vectors from the distribution implied by each $\boldsymbol{\lambda}$, but in practice it usually adds little.

by variables

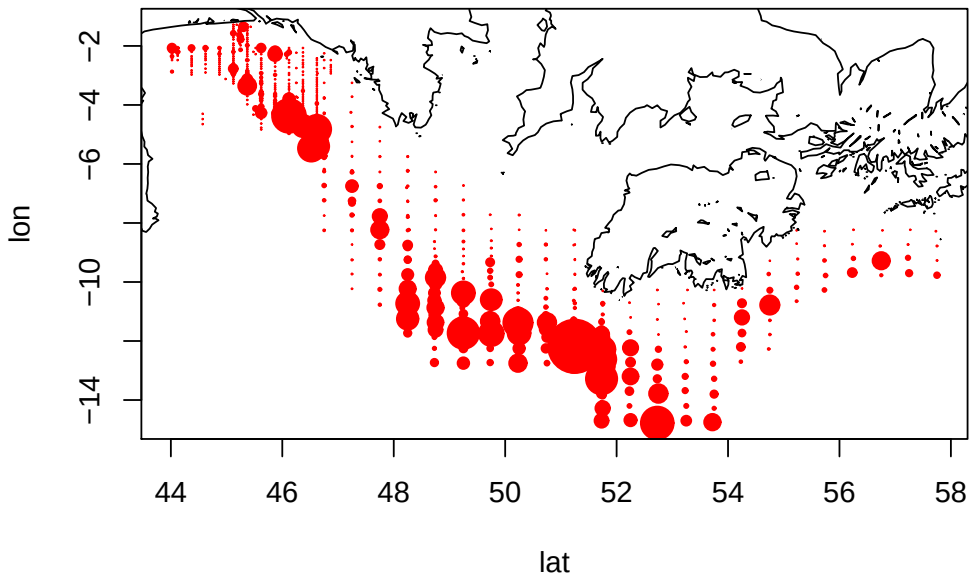
- `mgcv` allows smooths to 'interact' with simple parametric terms and factor variables, using the `by` argument to `s` and `te`.
- Starting with metric by variables, consider the model

$$y_i = \alpha + f(t_i)x_i + \epsilon_i$$

  where f is a smooth function.
- `gam(y ~ s(t,by=x))` would fit this (smooth not centered).
- No extra theory is required. `gam` just has to multiply each element of the i^{th} row of the model matrix for $f(t_i)$ by x_i for each i, and everything else is unchanged.
- Such models are sometimes called 'varying coefficient models'. The idea is that the linear regression coefficient for x_i is varying smoothly with t_i.
- When the smooth term is a function of location then the models are known as 'geographic regression models'.

Example geographic regression

- The dataframe `mack` contains data from a fisheries survey sampling mackerel eggs.

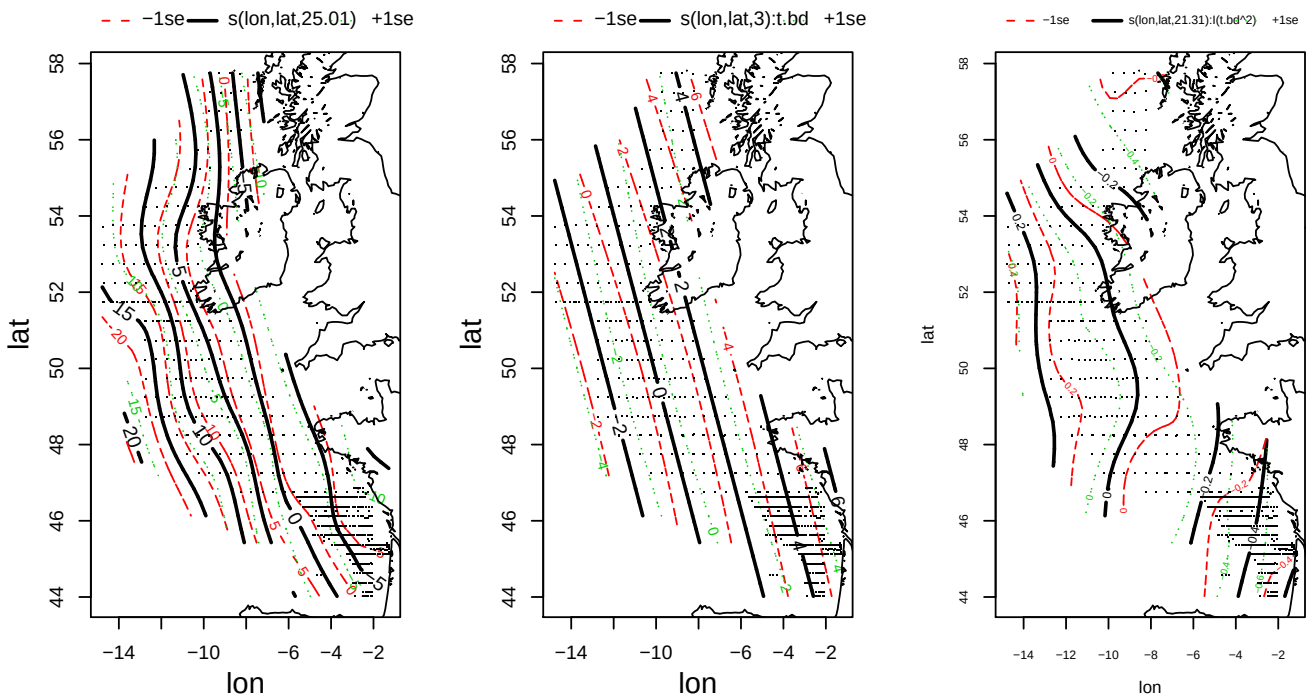


- One model is that egg densities are determined by a quadratic in (transformed) sea bed depth, but that the coefficients of this quadratic vary with location...

Mackerel model fit

- The model can be fitted by

```
mack$log.na <- log(mack$net.area)
mack$t.bd <- (mack$b.depth)^.25
b <- gam(egg.count ~ offset(log.na) + s(lon,lat) + s(lon,lat,by=t.bd)+
                     s(lon,lat,by=I(t.bd^2)),
         data=mack,family=Tweedie(p=1.1,link=log),method="ML")
for (i in 1:3) { plot(b,select=i);lines(coast)}
```



Concurvity

- Notice how uncertain the contours are in the previous model.
- There is a *concurvity* problem with the model.
- The covariate `t.bd` is itself very well modelled as a smooth function of the other covariates `lon` and `lat` ...

```
> b.conc <- gam(t.bd~s(lon,lat,k=50),data=mack,method="ML")
> summary(b.conc)
...
             edf Ref.df     F p-value
s(lon,lat) 46.83  48.82 100.3  <2e-16 ***
...
R-sq.(adj) =  0.885   Deviance explained = 89.4%
```

- Logically this means that all the model terms could be fairly well approximated by smooth functions of location...
- This is a difficult issue to deal with.

Smooth-factor interactions

- Occasionally a smooth-factor interaction is required.
- The `by` variable argument to `s` and `te` permits this.
- Iff `d` is a factor, and `x` is continuous then

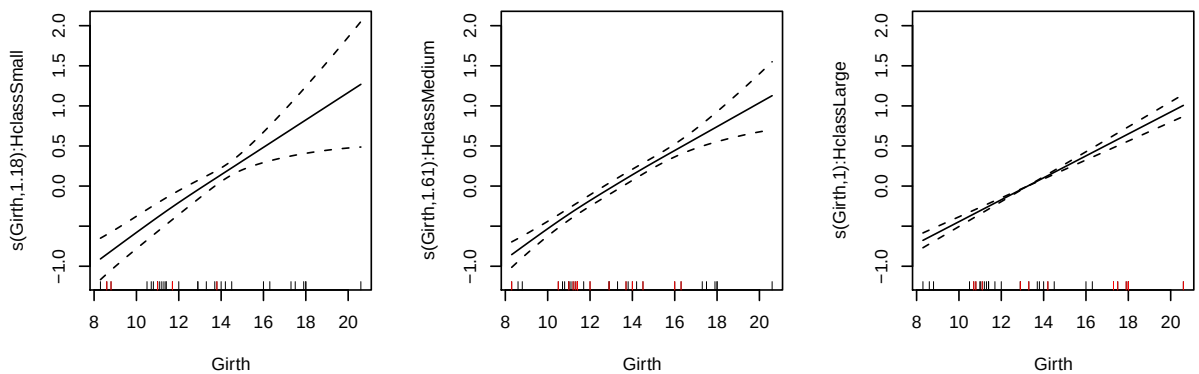
  `s(x,by=d)`

  creates a separate (centered) smooth of `x` for each level of `d`.
- To force the smooths to all have the same smoothing parameter, set the `id` argument to something, e.g.

  `s(x,by=d,id=1)`
- Note that the smooths are all subject to centering constraints. With a metric by variable this would not the case (unless the `by` variable is a constant, which it should not be).

Factor by example

- As an example recall the discrete Height class version of the `trees` data. We could try the model $\log E(\texttt{Volume}_i) = f_k(\texttt{Girth}_i)$ if tree i is height class k.
- ```
  ct5 <- gam(Volume~s(Girth,by=Hclass) + Hclass,
             family=Gamma(link=log),data=trees,method="REML")
  par(mfrow=c(1,3));plot(ct5)
  ```



- Notice that with factor `by variables` the smooths have centering constraints applied, hence the need for the separate `Hclass` term in the model.
  ```

The summation convention

- `s` and `te` smooth terms accept *matrix* arguments and `by` variables to implement general $L_{ij}f_j$ terms.
- If $\mathbf{X}$ and $\mathbf{L}$ are $n \times p$ matrices then

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

  evaluates $L_{ij}f_j = \sum_k f(X_{ik})L_{ik}$ for all i.
- For example, consider data $y_i \sim \mathrm{Poi}$ where

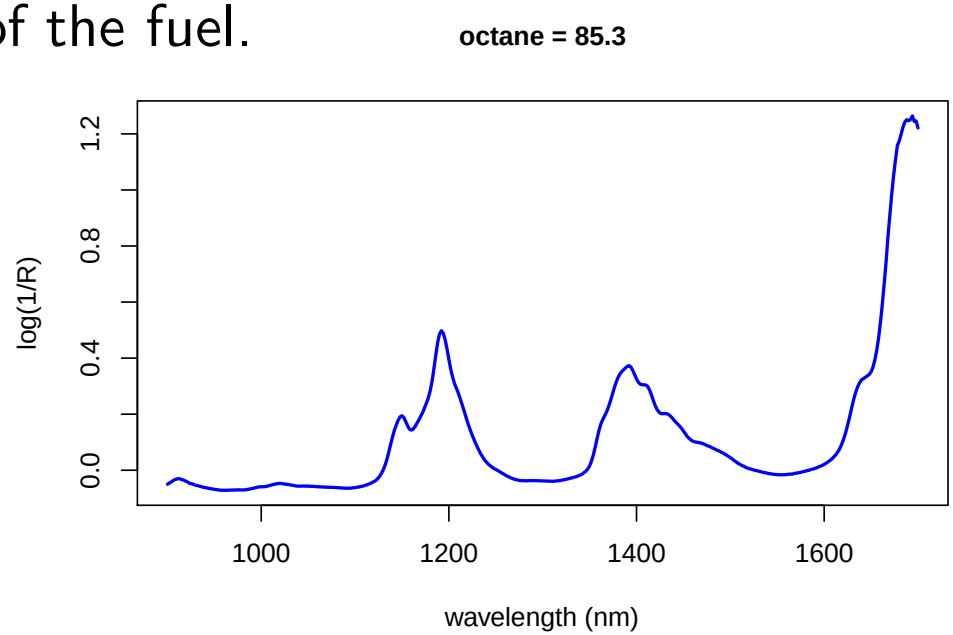
$$\log\{\mathbb{E}(y_i)\} = \int k_i(x)f(x)dx \simeq \frac{1}{h}\sum_{k=1}^{p} k_i(x_k)f(x_k)$$

  (the x_k are evenly spaced points).
- Let $X_{ik} = x_k \ \forall \ i$ and $L_{ik} = k_i(x_k)/h$. The model is fit by

  ```
  gam(y ~ s(X,by=L),poisson)
  ```

Summation example: predicting octane

- Consider predicting octane rating of fuel from near infrared spectrum of the fuel.



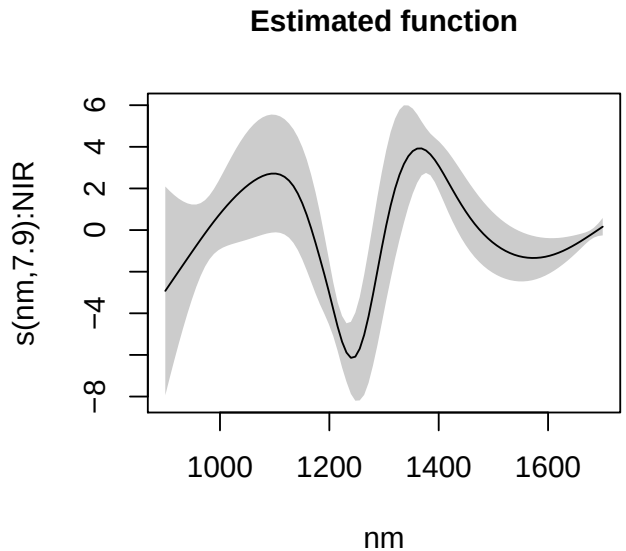
- There are 60 such spectrum ($k_i(x)$) - octane (y_i) pairs (x is wavelength), and a model might be

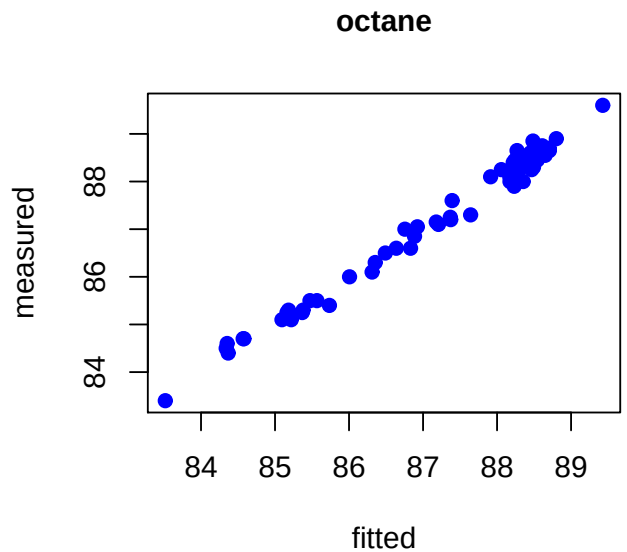
$$y_i = \alpha + \int f(x)k_i(x)dx + \epsilon_i$$

where $f(x)$ is a smooth function of wavelength.

Fitting the octane model

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





- . . . can predict octane quite well from NIR spectrum.
  ```

Model selection

- Various model selection strategies are possible. Two stand out

  1. Use backward selection, based on GCV, REML or AIC, possibly guided by termwise approximate p-values and plots.
  2. Let smoothness selection do all the work by adding a penalty on the null space of each smooth. `gam(...,select=TRUE)` does this.

- The second option is nicely consistent with how we select between models of different smoothness, but works the optimizer rather hard.
- If H_0 type selection is desired approximate p-values can be used, or we can increase `gamma` so that single term deletion by AIC is equivalent to using a significance level of e.g. 5% as opposed to the AIC default of 15%. i.e. set `gamma = 3.84/2`.
- It is rare for fully automatic selection to be fully satisfactory.

Mackerel selection example

- As an example, consider the `mack` data again, but this time we'll use an additive structure, with a number of candidate predictors...

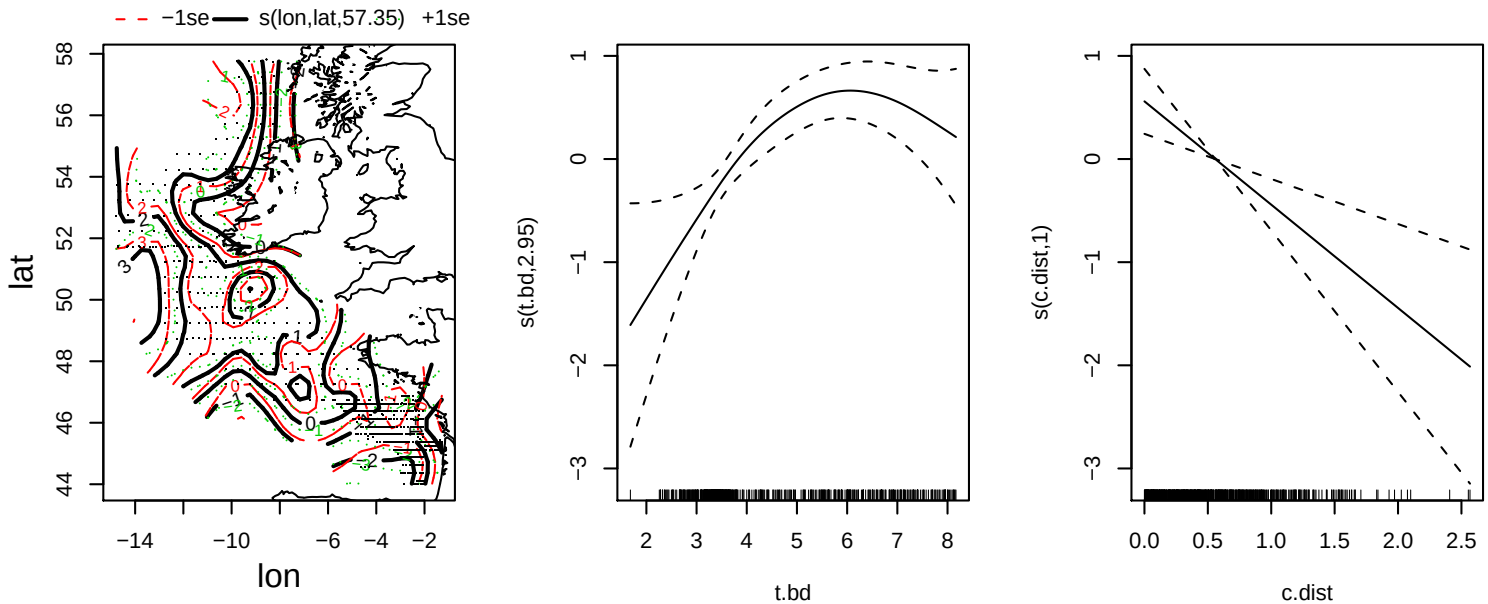
```
> b2 <- gam(egg.count ~ offset(log.na) + s(lon,lat,k=100) + s(t.bd)
+                 + s(temp.20m)+s(c.dist)+s(temp.surf)+s(vessel,bs="re"),
+                 data=mack,family=Tweedie(p=1.3,link=log),
+                 method="ML",select=TRUE)
> b2
...
Estimated degrees of freedom:
5.6647e+01 2.8944e+00 3.1203e-04 1.2619e+00 2.4552e-04 2.8475e+00
total = 64.65149

ML score: 1555.35
```

- So the smooths of `temp.20m` and `temp.surf` have been penalized out of the model.

Mackerel selection example continued

- Refitting we have. . .
- ```
  b3 <- gam(egg.count ~ offset(log.na) + s(lon,lat,k=100) + s(t.bd)
                    + s(c.dist)+s(vessel,bs="re"),
                    data=mack,family=Tweedie(p=1.3,link=log),
                    method="ML")
  data(coast)
  par(mfrow=c(1,3));plot(b3)
  ```


  ```

Time to stop

- Goodbye.