A toolbox of smooths

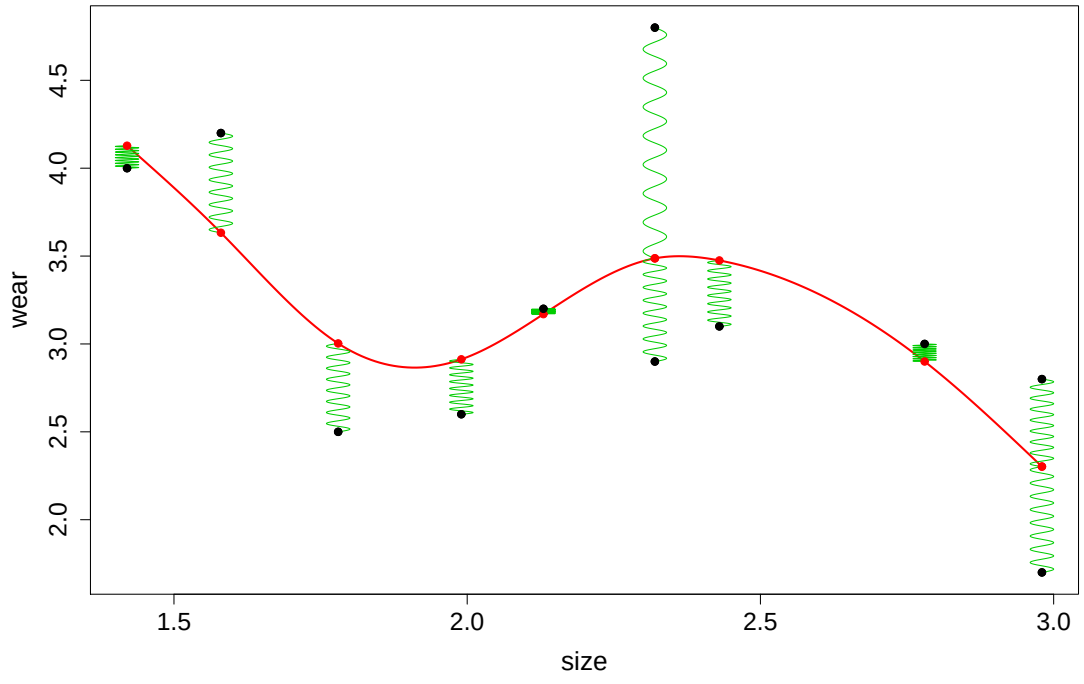
Simon Wood
Mathematical Sciences, University of Bath, U.K.

Smooths for semi-parametric GLMs

- To build adequate semi-parametric GLMs requires that we use functions with appropriate properties.
- 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.

Splines

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

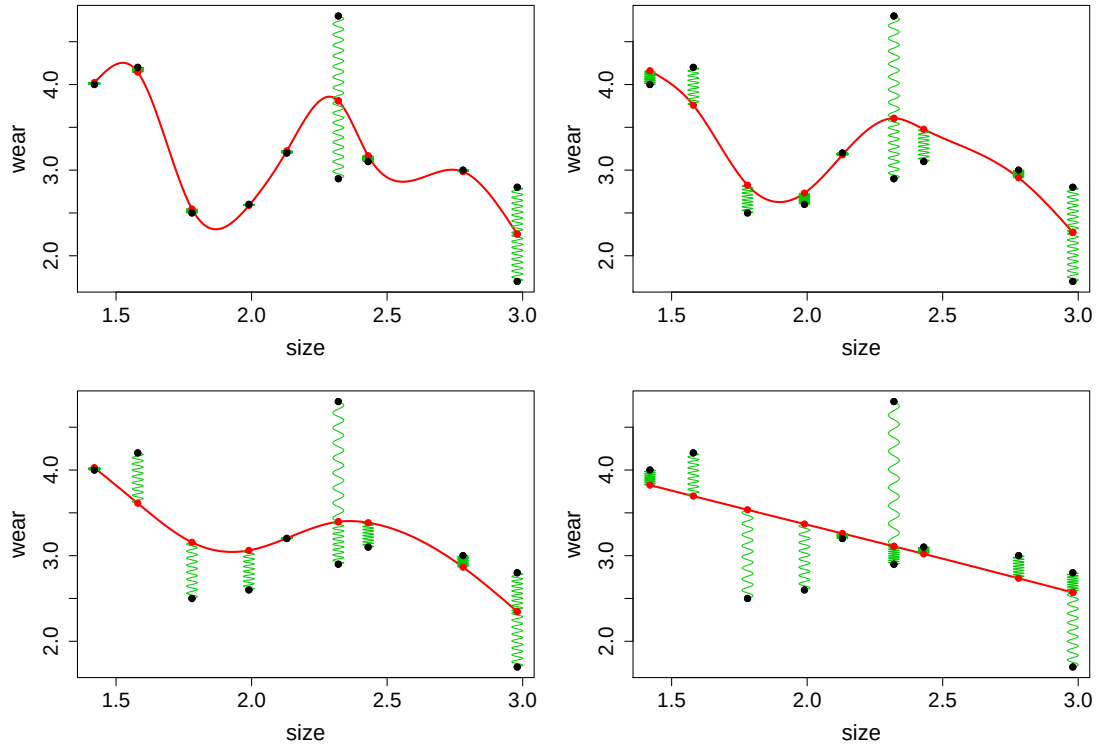


- Mathematically the red curve is the *function* minimizing

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

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.

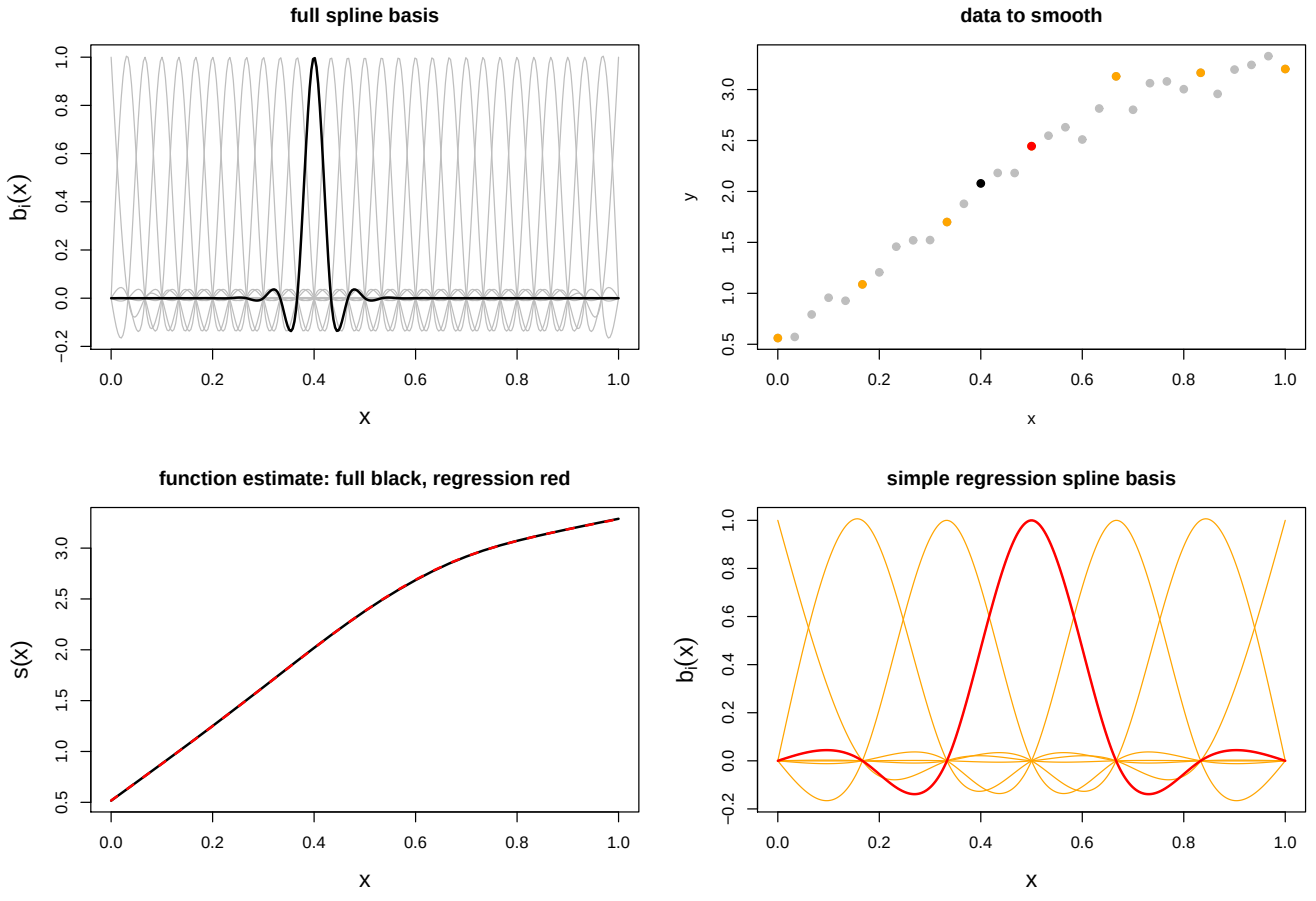
Penalized regression splines

- 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*.

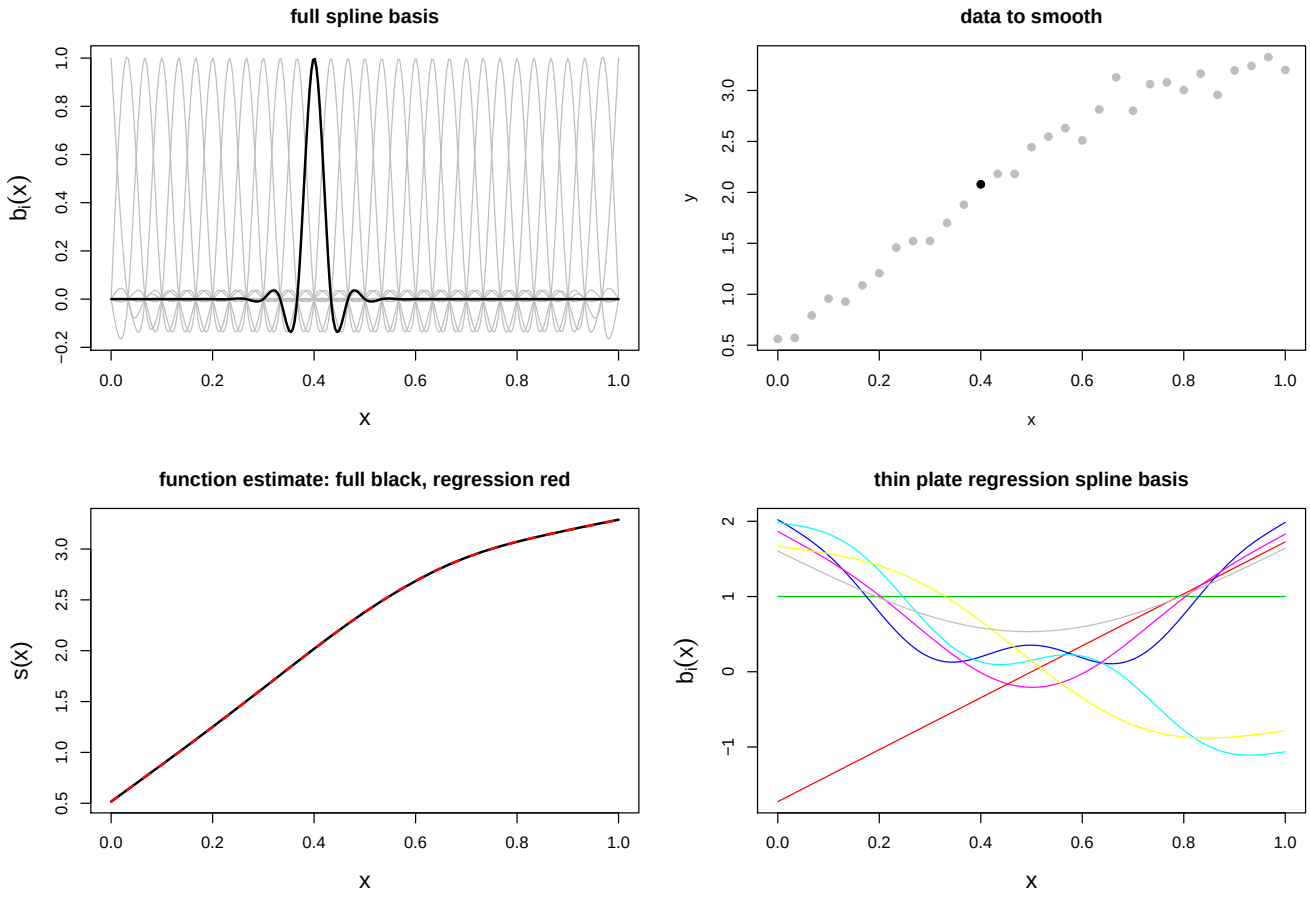
Knot based example: `"cr"`

- 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).



1 dimensional smoothing in `mgcv`

- Smooth functions are specified by terms like `s(x,bs="ps")`, on the rhs of the model formula.
- The `bs` argument of `s` specifies the class of basis...

  `"cr"` knot based cubic regression spline.

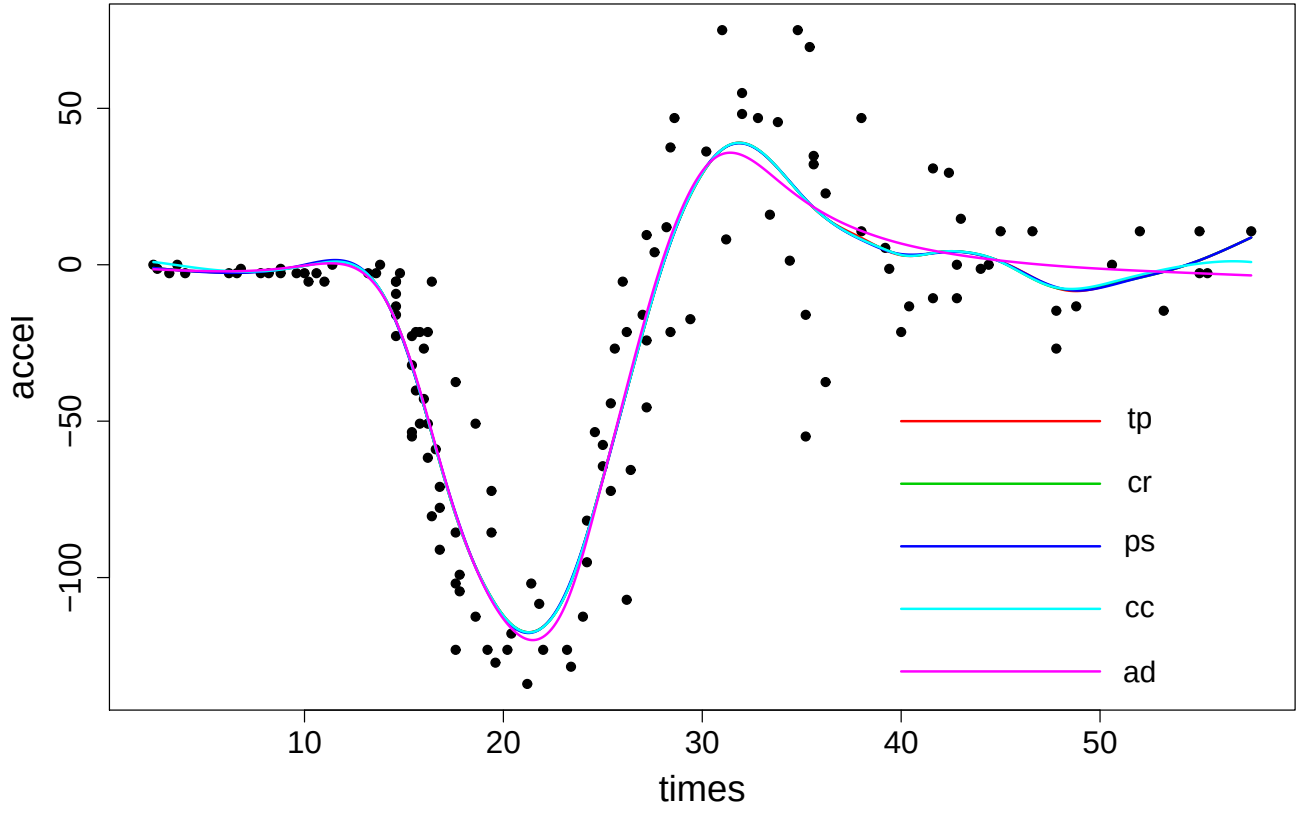
  `"cc"` cyclic version of above.

  `"ps"` Eilers and Marx style p-splines, with flexibility as to order of penalties and basis functions.

  `"ad"` adaptive smoother in which strength of penalty varies with covariate.

  `"tp"` thin plate regression spline. Optimal low rank eigen approx. to a full spline: flexible order penalty derivative.
- Smooth classes can be added (`?smooth.construct`).

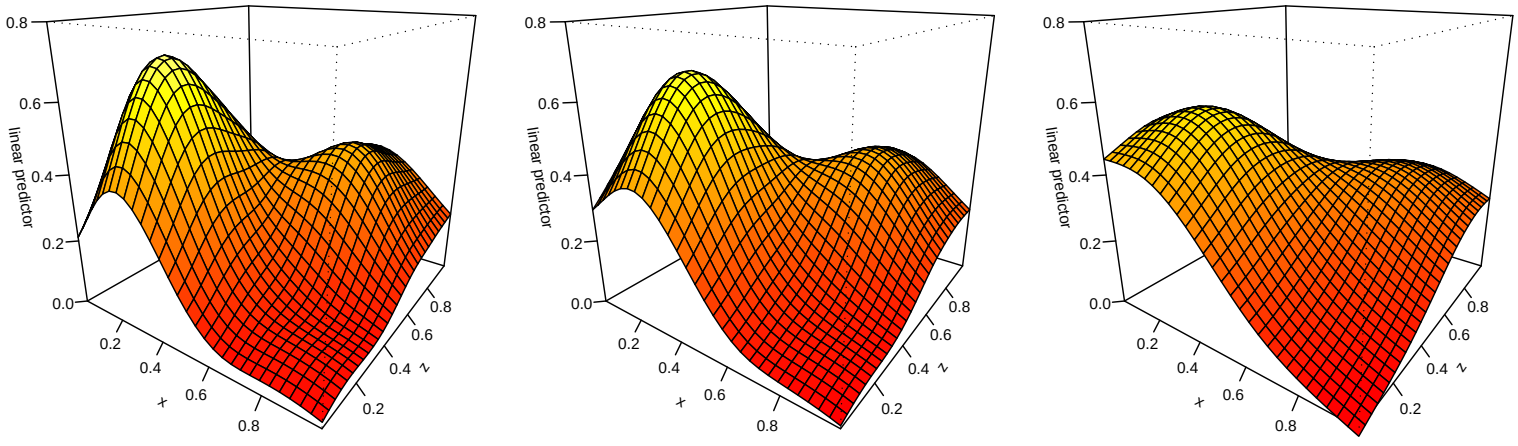
1D smooths compared



- 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.

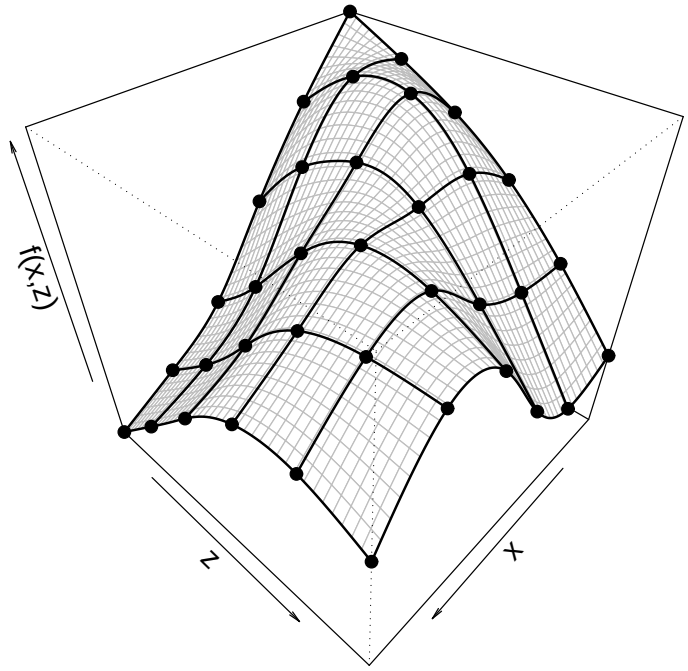
Isotropic smooths

- 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.

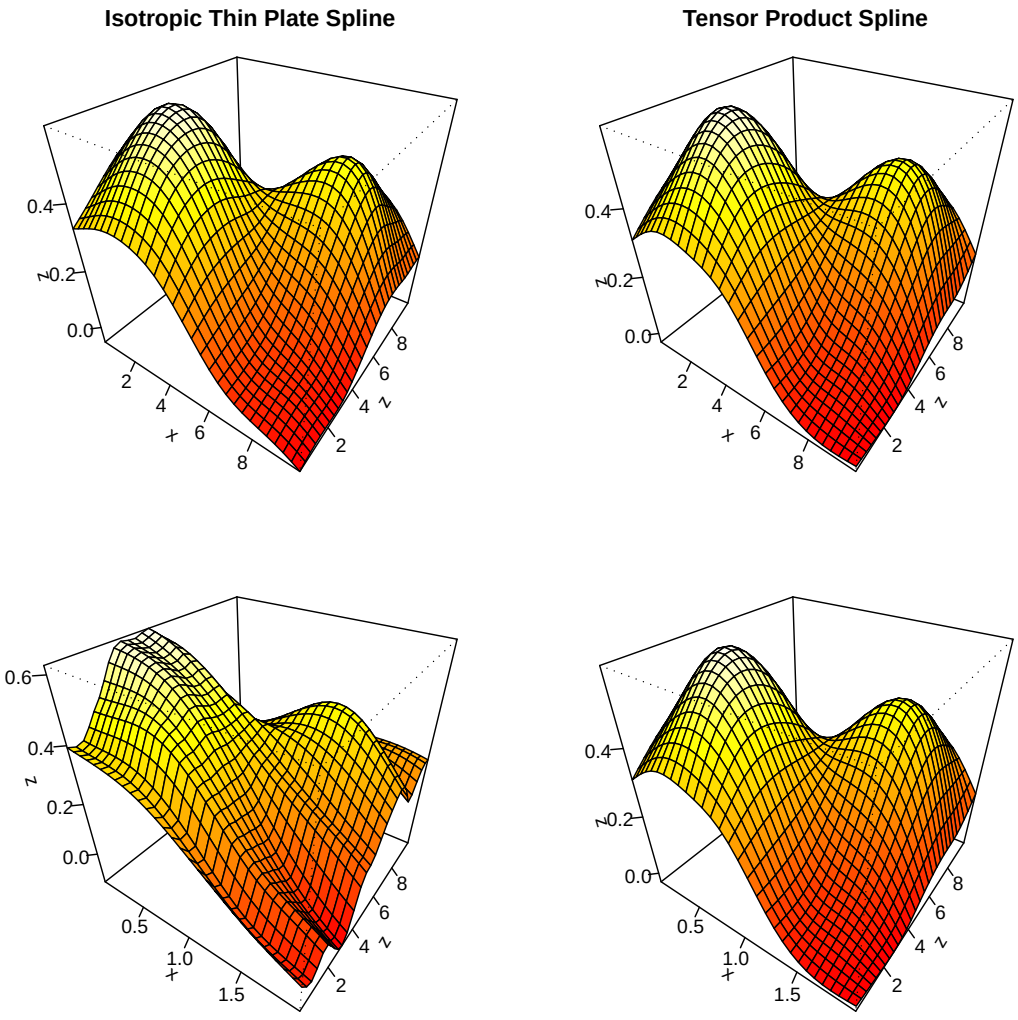


Scale invariant smoothing: tensor product smooths

- 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*.



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.

Tensor product smoothing in `mgcv`

- Tensor product smooths are constructed automatically from *marginal* smooths of lower dimension. The resulting smooth has a penalty for each marginal basis.
- `mgcv` can construct tensor product smooths from any *single penalty* smooths useable with `s` terms.
- `te` terms within the model formula invoke this construction. For example:
  - `te(x,z,v,bs="ps",k=5)` creates a tensor product smooth of x, z and v using rank 5 P-spline marginals: the resulting smooth has 3 penalties and basis dimension 125.
  - `te(x,z,t,bs=c("tp","cr"),d=c(2,1),k=(20,5))` creates a tensor product of an isotropic 2-D TPS with a 1-D smooth in time. The result is isotropic in x,z, has 2 penalties and a basis dimension of 100. This sort of smooth would be appropriate for a location-time interaction.
- `te` terms are invariant to linear rescaling of covariates.

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`.
- The 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.
- Suppose that you want to check if the `s(x,k=15)` term in a model has too small a basis. Here's a trick ...

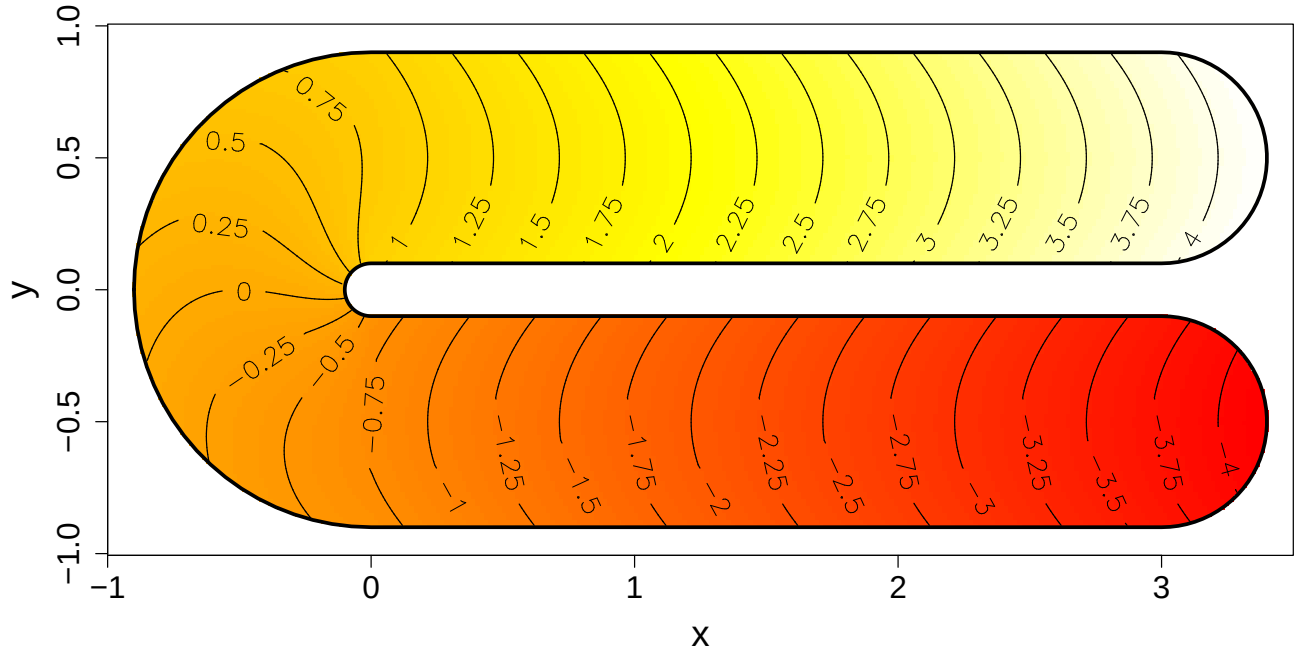
```
b <- gam(y~s(x,k=15)+s(v,w),Gamma(log))
rsd <- residuals(b)
b1 <- gam(rsd ~ s(x,k=30),method="ML")
b1 ## any pattern?
```

Miscellanea

- Most smooths will require an identifiability condition to avoid confounding with the model intercept: `gam` handles this by automatic reparameterization.
- `gam` will also handle the side conditions required for nested smooths. e.g. `gam(y~s(x)+s(z)+s(x,z))` will work.
- However, nested models make most sense if the bases are strictly nested. To ensure this, smooth interactions should be constructed using marginal bases identical to those used for the main effects.
  `gam(y~te(x)+te(z)+te(x,z))`
  would achieve this, for example.
- `te` and `s(...,bs="tp")` can, in principle, handle any number of covariates.
- The `"ad"` basis can handle 1 or 2 covariates, but no more.

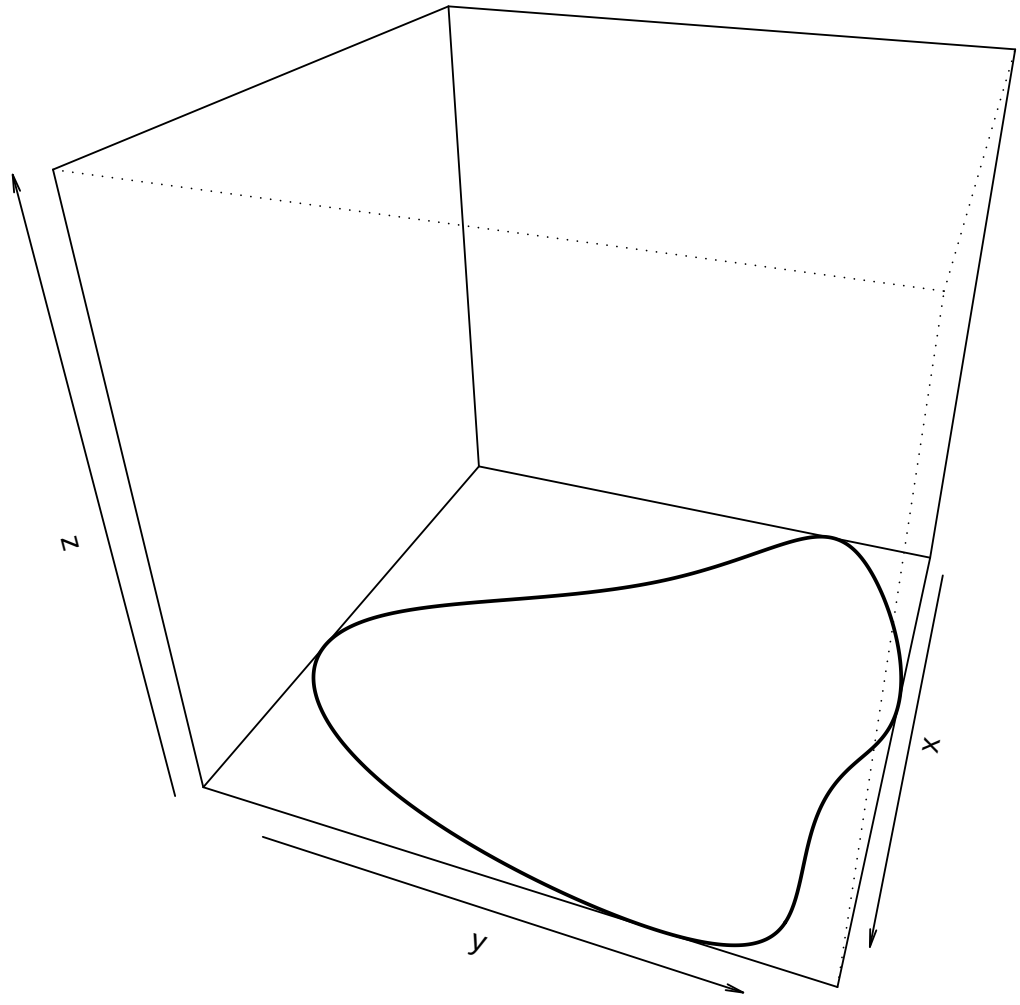
A diversion: finite area smoothing

- Suppose how want to smooth samples from this function

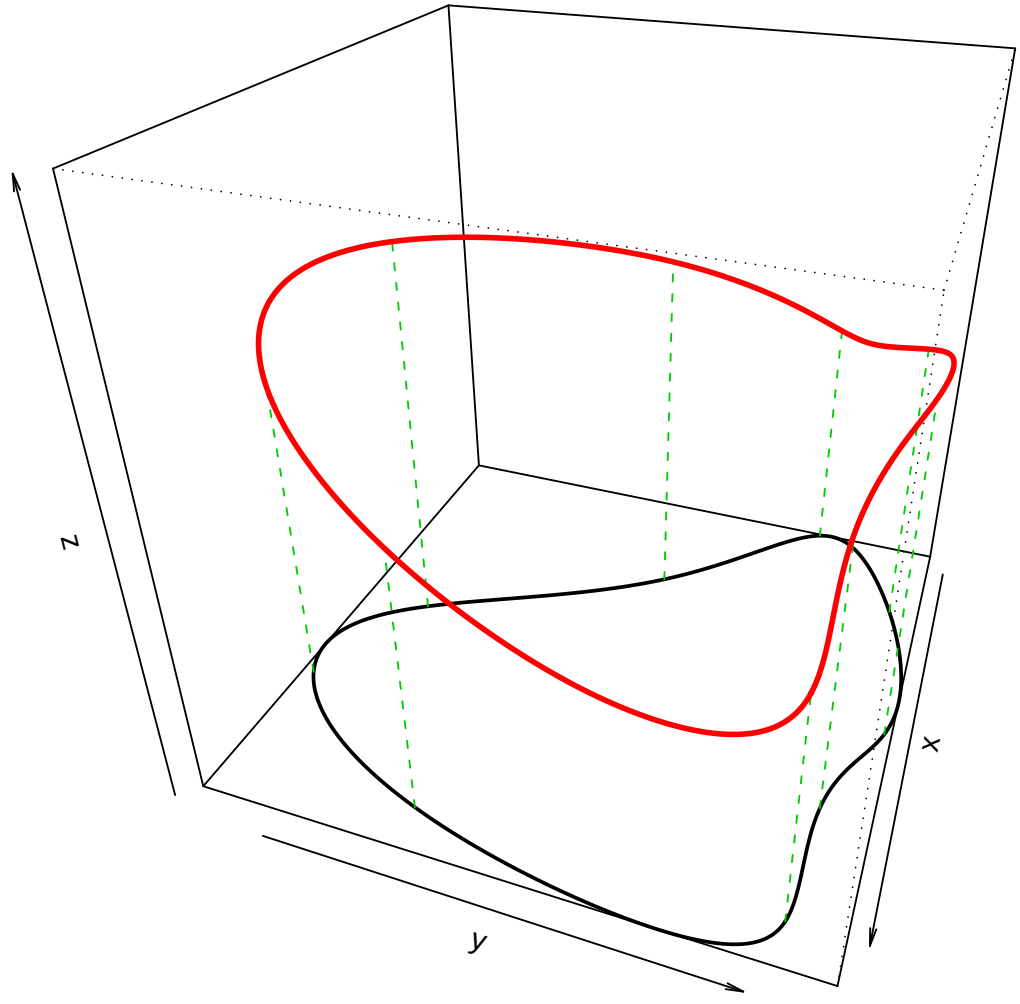


- . . . without ‘smoothing across’ the gap in the middle?
- Let’s use a soap film . . .

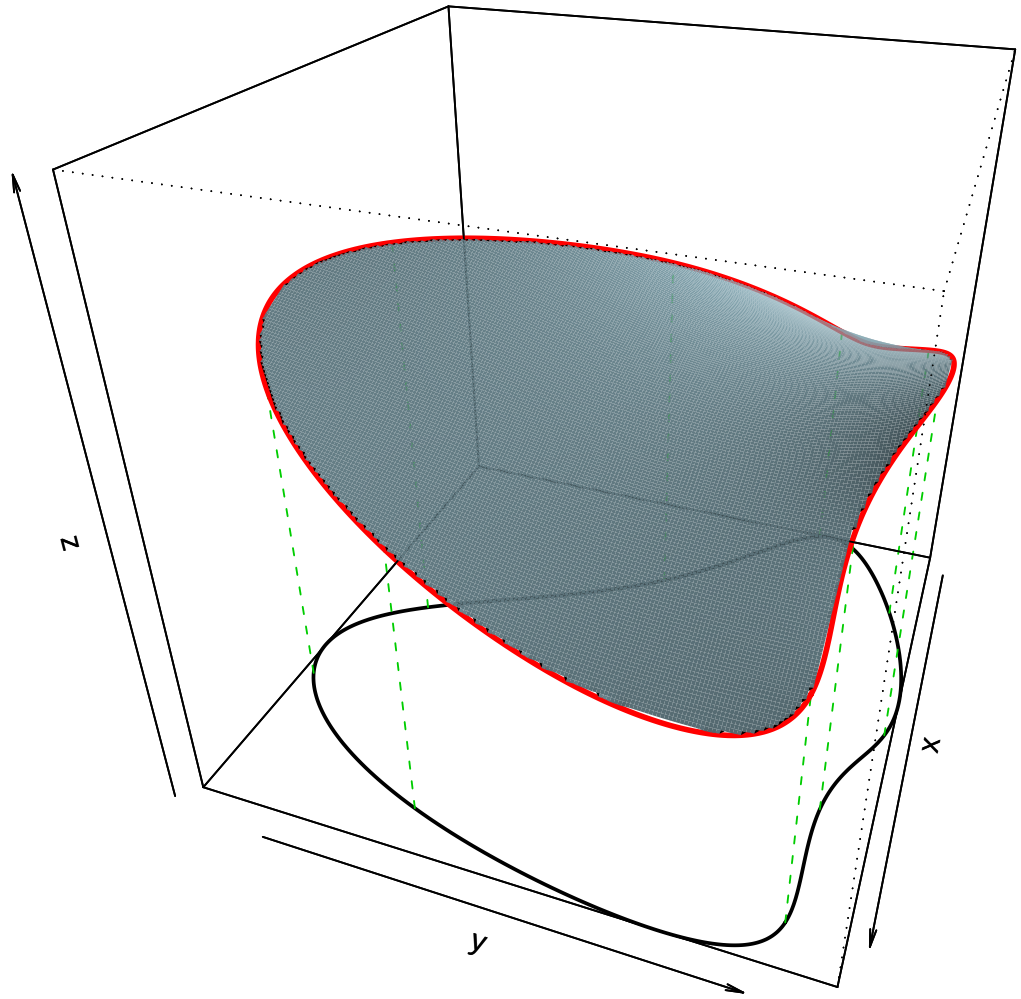
The domain



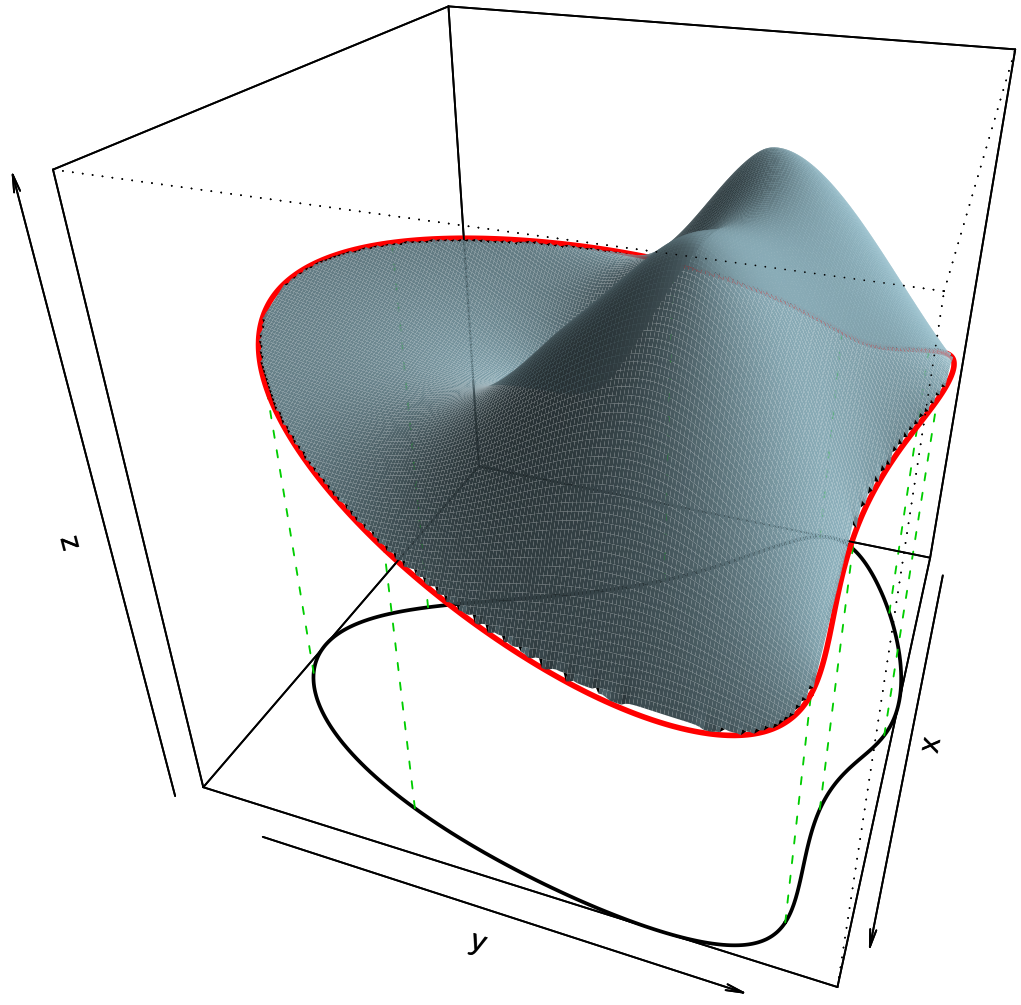
The boundary condition



The boundary interpolating film

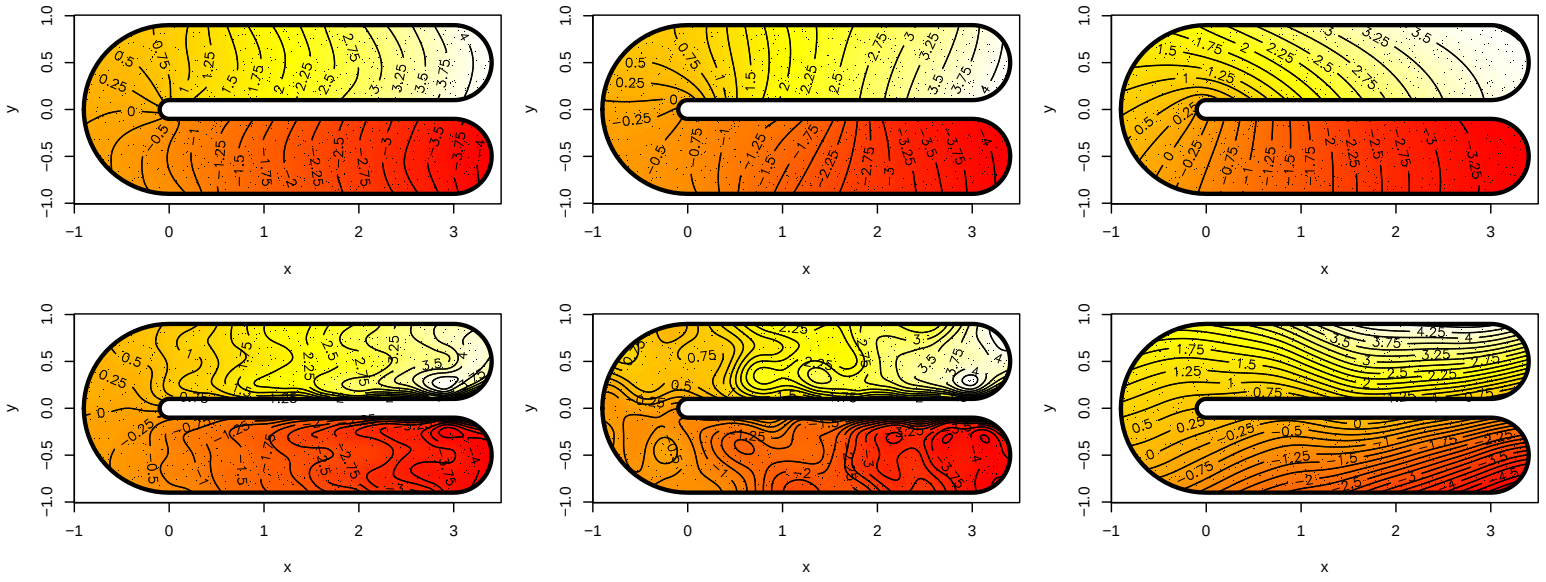


Distorted to approximate data



Soap film smoothers

- Mathematically this smoother turns out to have a basis-penalty representation.
- It also turns out to work. . .



Summary

- 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, `te`, is appropriate. (`te` terms may be isotropic in some marginals.)
- Occasionally in 2D a *finite area* smooth may be needed.
- The basis dimension is a modelling decision that should be checked.