

# Lecture 2a

block 2

Splines

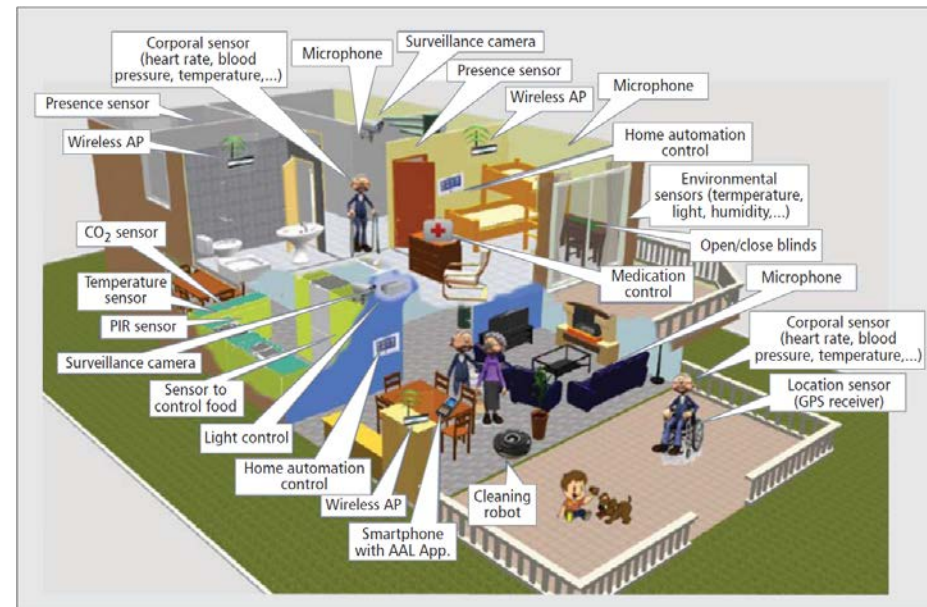
Generalized additive models

# Moving beyond simple models

- Sometimes using simple models (linear regression) is not enough
  - Too simple → *more flexible models are needed*

## Example: Ambient Assisted Living

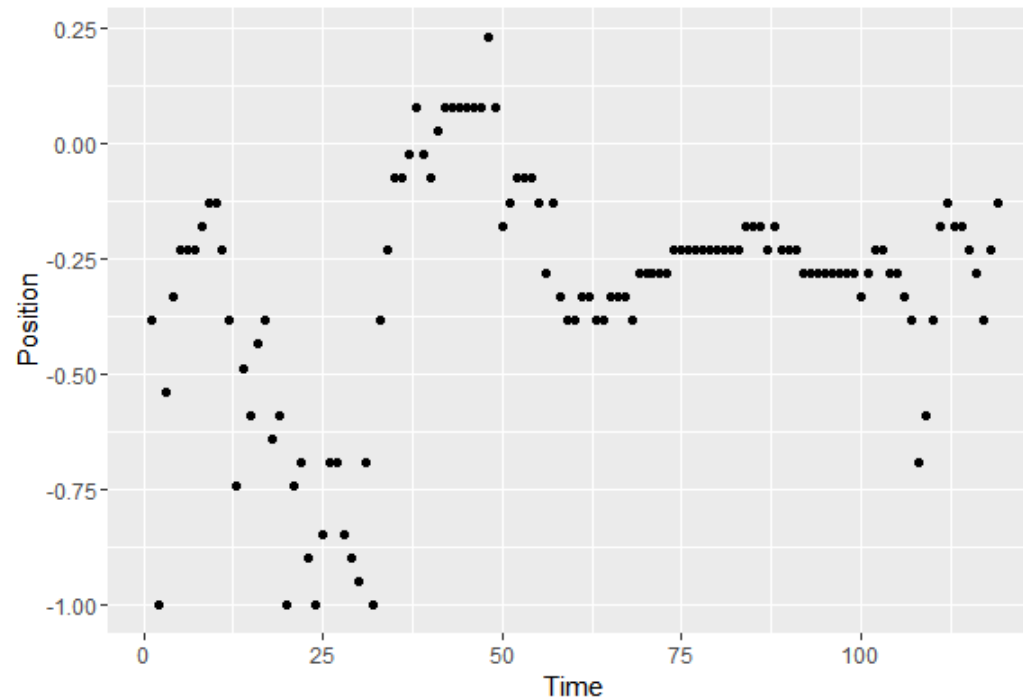
- digitally connected and controlled devices for support of people with special needs
- emergency buttons, pressure emergency services with connection to a broader smart home



<https://bstassen.files.wordpress.com/2015/01/aal.png>

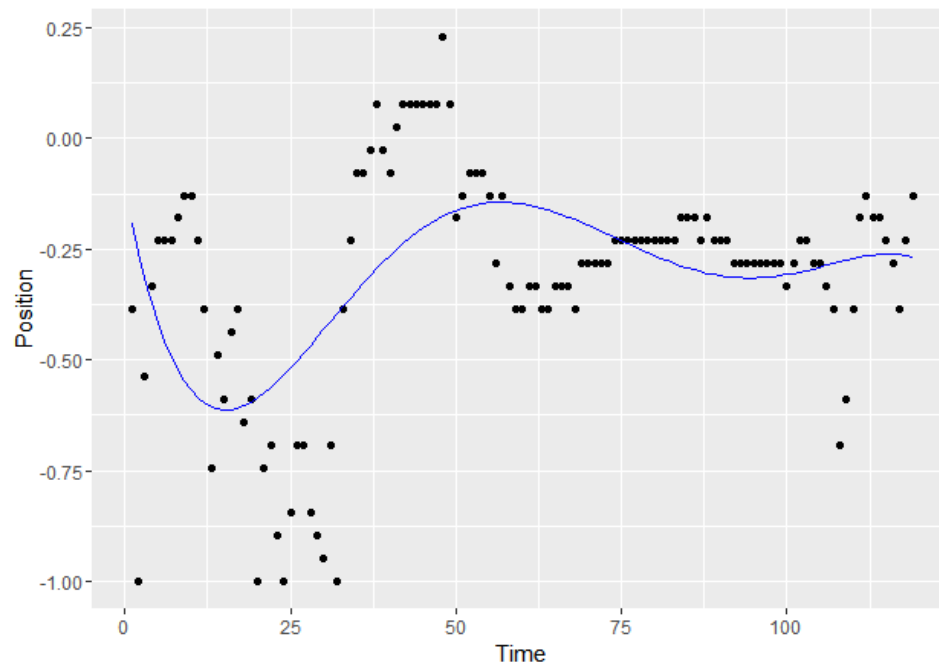
# Moving beyond simple models

- Ambient Assisted Living
  - Person's movement is detected by Radio Signal Strength (RSS) measurements → how to remove noise?



# Moving beyond simple models

- Attempt 1: 5th degree polynomial

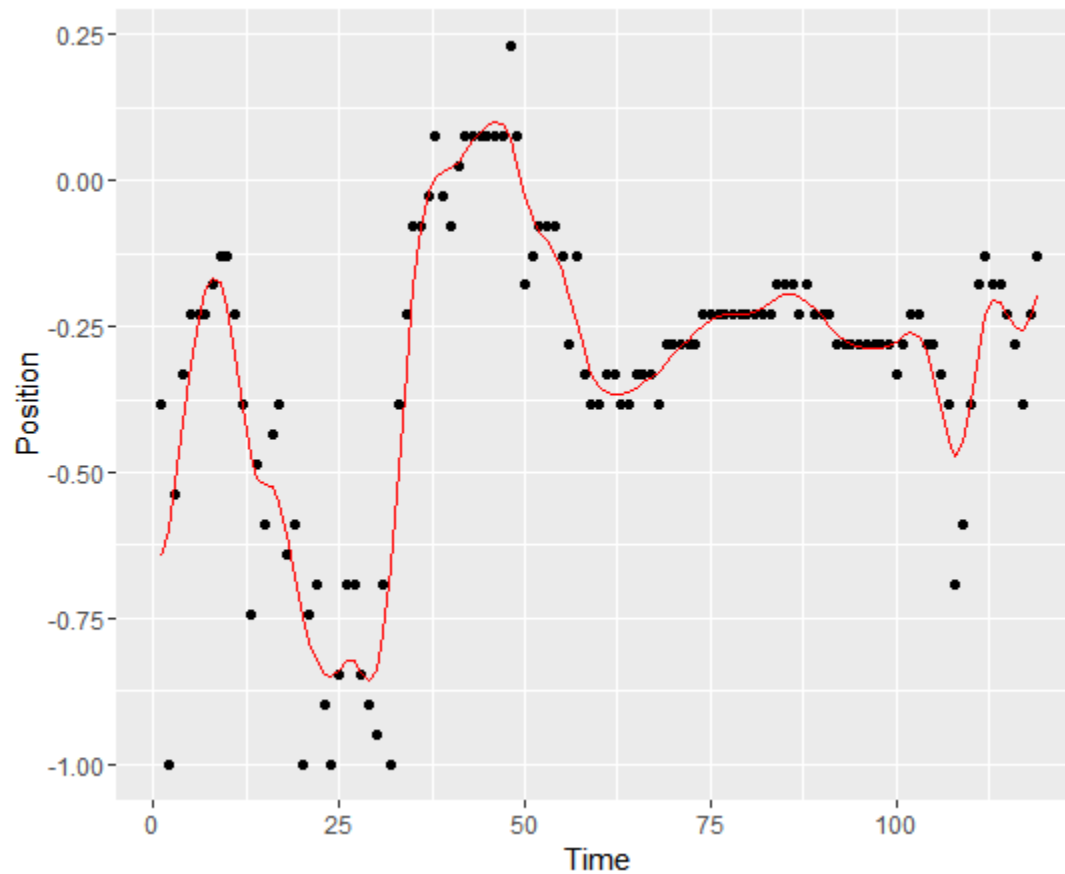


Model underfits data → need more flexible models



# Moving beyond simple models

- Using smoothing splines



# Basis function expansion

If  $y = w_0 + w_1x_1 + w_2x_1^2 + w_3e^{-x_2} + \epsilon$ ,

Model becomes linear if to recompute:

$$\begin{aligned}\phi_1(x_1) &= x_1 \\ \phi_2(x_1) &= x_1^2 \\ \phi_3(x_1) &= e^{-x_2}\end{aligned}$$

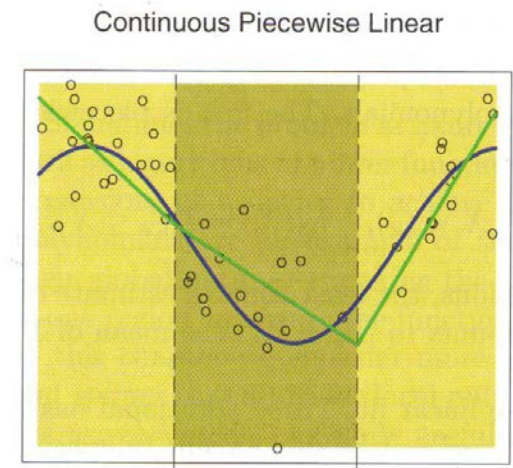
- Any model of the type  $Ey = \sum_i w_i \phi_i(\mathbf{x})$  can be fit by linear regression!

# Constructing a piecewise linear function

**Method A.** Introduce linear functions on each interval and a set of constraints

(4 free parameters)

$$\begin{cases} y_1 = \alpha_1 x + \beta_1 \\ y_2 = \alpha_2 x + \beta_2 \\ y_3 = \alpha_3 x + \beta_3 \end{cases}$$
$$\begin{cases} y_1(\xi_1) = y_2(\xi_1) \\ y_2(\xi_2) = y_3(\xi_2) \end{cases}$$



**Method B.** Use a basis expansion (4 free parameters)

$$h_1(X) = 1, h_2(X) = X, h_3(X) = (X - \xi_1)_+, h_4(X) = (X - \xi_2)_+$$

**Theorem.** The two methods are equivalent.



# Splines

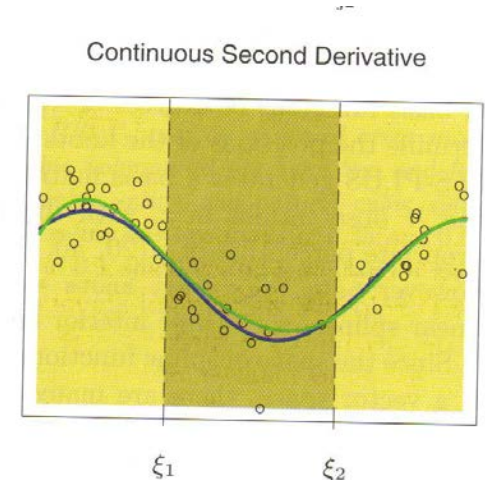
- A piecewise polynomial is called an **order- $M$**  (or degree  $M-1$ ) **spline** if it is continuous and has continuous derivatives up to order  $M-2$  at the knots.
- **Equivalent:** An order- $M$  spline with  $K$  knots:

$$h_j(X) = X^{j-1}, j = 1, \dots, M$$

$$h_{M+l}(X) = (X - \xi_l)_+^{M-1}, l = 1, \dots, K$$

- An order-4 (degree-3) spline is called a **cubic spline**

In cubic splines, knot discontinuity is not visible





# Natural cubic spline

- A cubic spline  $f$  is called **natural cubic spline** if its 2<sup>nd</sup> and 3<sup>rd</sup> derivatives are zero at  $a$  and  $b$

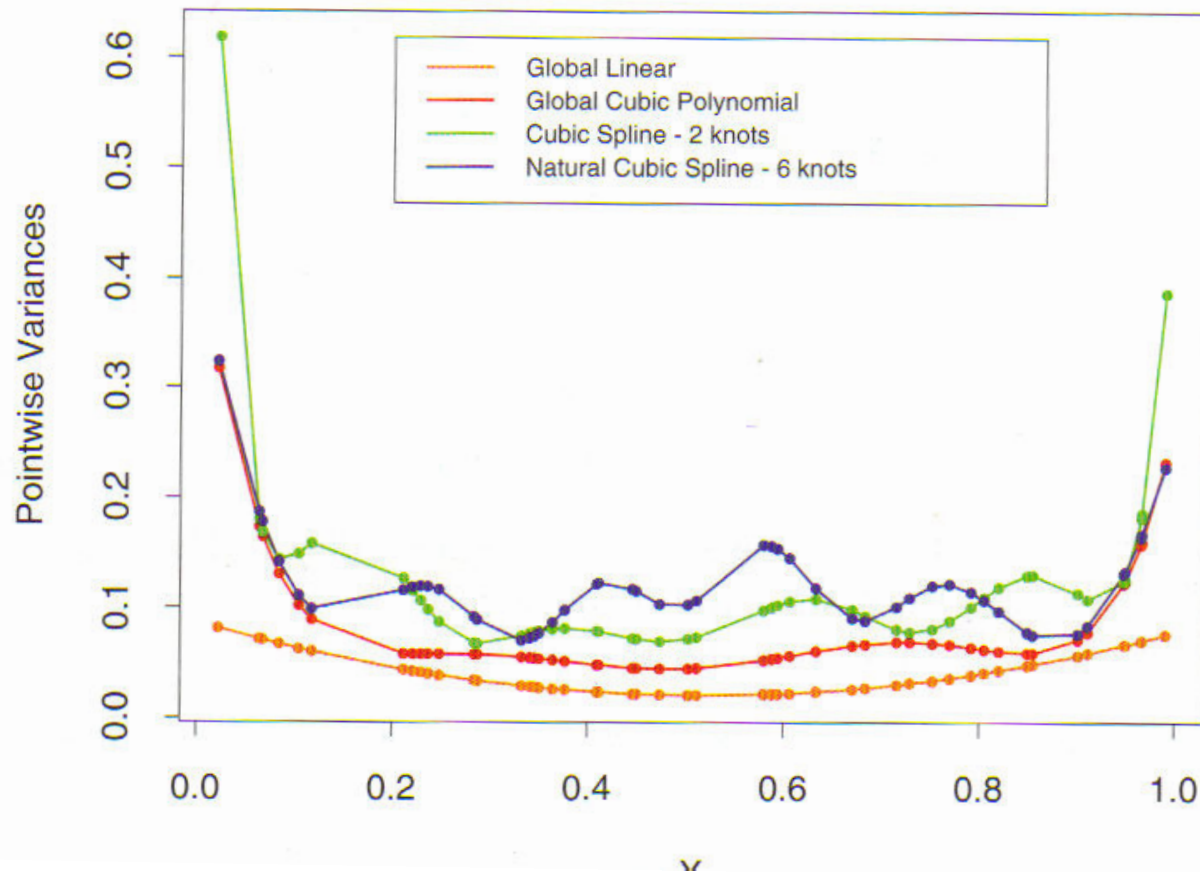
**Note** that  $f$  is linear on extreme intervals

Basis functions of natural cubic splines

$$N_1(X) = 1, N_2(X) = X, N_{k+2} = d_k(X) - d_{K-1}(X), \quad k = 1, \dots, K-2$$

$$\text{where } d_k(X) = \frac{(X - \xi_k)_+^3 - (X - \xi_K)_+^3}{\xi_K - \xi_k}$$

# Variance of spline estimators – boundary effects



# Fitting smooth functions to data

- Minimize

$$RSS(f, \lambda) = \sum_{i=1}^N (y_i - f(x_i))^2 + \lambda \int \{f''(t)\}^2 dt$$

where  $\lambda$  is **smoothing parameter**.

$\lambda=0$  : any function interpolating data

$\lambda=+\infty$  : least squares line fit

# Optimality of smoothing splines

- The function  $f$  minimizing  $RSS$  for a given  $\lambda$  is a natural cubic spline with knots at all unique values of  $x_i$  (NOTE:  $N$  knots!)
- Minimizing sum of squares:

$$f(x) = \sum_{j=1}^N N_j(x) \theta_j = \mathbf{N}(x)^T \boldsymbol{\Theta}$$

$$RSS(\boldsymbol{\Theta}, \lambda) = (\mathbf{y} - \mathbf{N}\boldsymbol{\Theta})^T (\mathbf{y} - \mathbf{N}\boldsymbol{\Theta}) + \lambda \boldsymbol{\Theta}^T \boldsymbol{\Omega}_N \boldsymbol{\Theta}$$

$$\{\mathbf{N}\}_{ij} = N_j(x_i) \quad \{\boldsymbol{\Omega}_N\}_{ij} = \int N_i''(t) N_j''(t) dt$$

$$\hat{\boldsymbol{\Theta}} = (\mathbf{N}^T \mathbf{N} + \lambda \boldsymbol{\Omega}_N)^{-1} \mathbf{N}^T \mathbf{y}$$



# A smoothing spline is a linear smoother

- Smoothing spline

$$\hat{f} = \mathbf{N}(\mathbf{N}^T \mathbf{N} + \lambda \Omega_N)^{-1} \mathbf{N}^T \mathbf{y} = \mathbf{S}_\lambda \mathbf{y}$$

is a **linear smoother**.

- Compare with other smoothers, such as linear regression.

# Degrees of freedom

- It can be shown that

$$\mathbf{S}_\lambda = (\mathbf{I} + \lambda \mathbf{K})^{-1}$$

where  $\mathbf{K}$  is **penalty matrix**

- Eigenvalue decomposition of  $\mathbf{K}$  :

$$\mathbf{S}_\lambda = \sum_{k=1}^N \rho_k(\lambda) \mathbf{u}_k \mathbf{u}_k^T$$

$$\rho_k(\lambda) = \frac{1}{1 + \lambda d_k}$$

- $d_k$  and  $\mathbf{u}_k$  are eigenvalues and eigenvectors

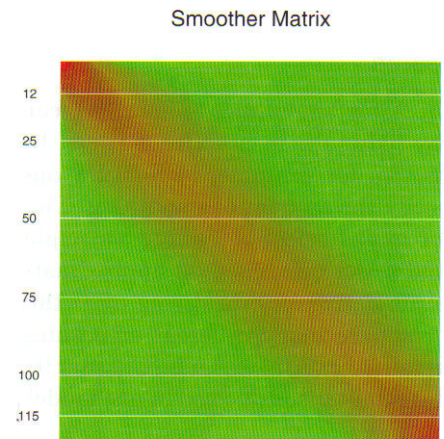
# Smoothing splines and shrinkage

$$\mathbf{S}_\lambda \mathbf{y} = \sum_{k=1}^N \mathbf{u}_k \rho_k(\lambda) \mathbf{u}_k^T \mathbf{y}$$

- Smoothing spline decomposes vector  $\mathbf{y}$  with respect to basis of eigenvectors and shrinks respective contributions
- The eigenvectors ordered by  $\rho$  increase in complexity. The higher the complexity, the more the contribution is shrunk.

# Penalty and degrees of freedom

- $df_\lambda = \text{trace}(\mathbf{S}_\lambda) \rightarrow df_\lambda = \sum_{k=1}^N \frac{1}{1 + \lambda d_k}$
- $\lambda$  increase  $\rightarrow df_\lambda$  decrease
- higher  $\lambda \rightarrow$  higher penalization.
- Smoother matrix is has banded nature  
 $\rightarrow$  local fitting method







# Automated selection of smoothing parameters

## What can be selected:

### Regression splines

- Degree of spline
- Placement of knots

### Smoothing spline

- Penalization parameter

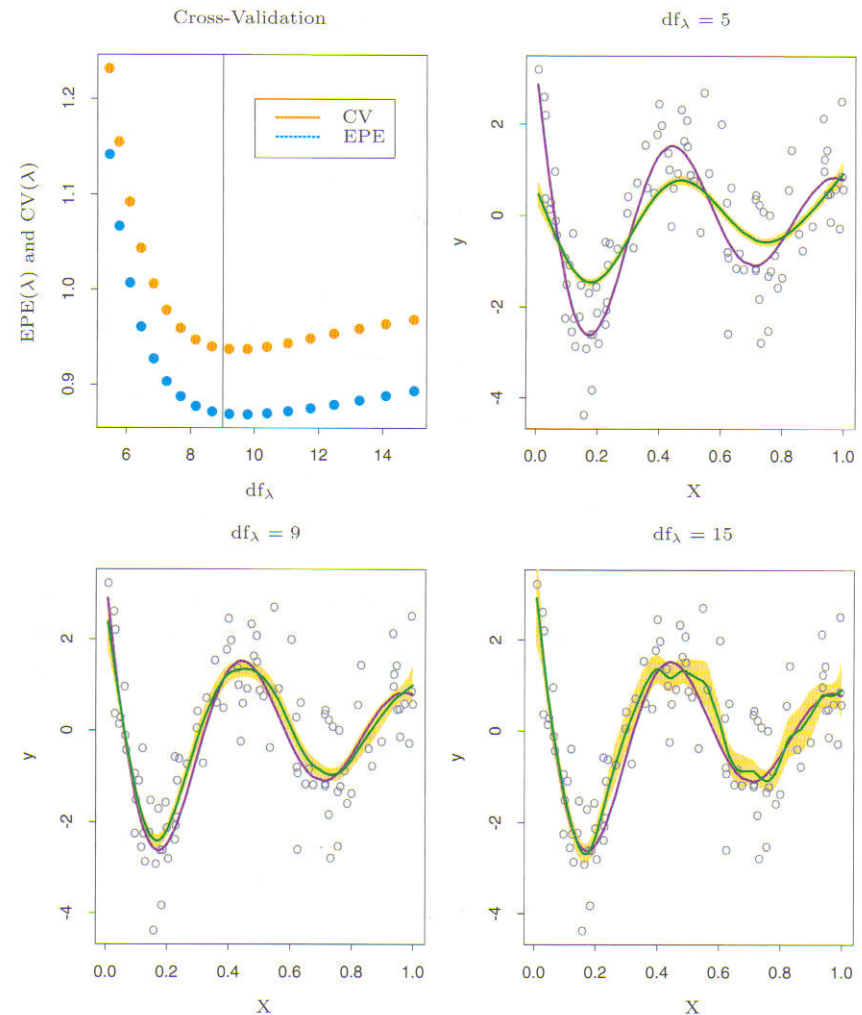
# Automated selection of smoothing parameters

$$df_{\lambda} = \text{trace}(\mathbf{S}_{\lambda}) = \sum_{k=1}^N \frac{1}{1 + \lambda d_k}$$

- Use either  $df_{\lambda}$  or  $\lambda$ 
  - Given  $df_{\lambda} \rightarrow$  solve equation  $\rightarrow$  find  $\lambda$
- Use holdout principle or cross validation for parameter tuning

# Automated selection of smoothing parameters

- Bias-variance tradeoff



# Multidimensional splines

## How to fit data smoothly in higher dimensions?

- Formulate a new problem

$$\min \sum_i (y_i - f(x_i))^2 + \lambda J[f]$$

- The solution is **thin-plate splines**
- The solution in 2 dimensions is essentially sum of radial basis functions

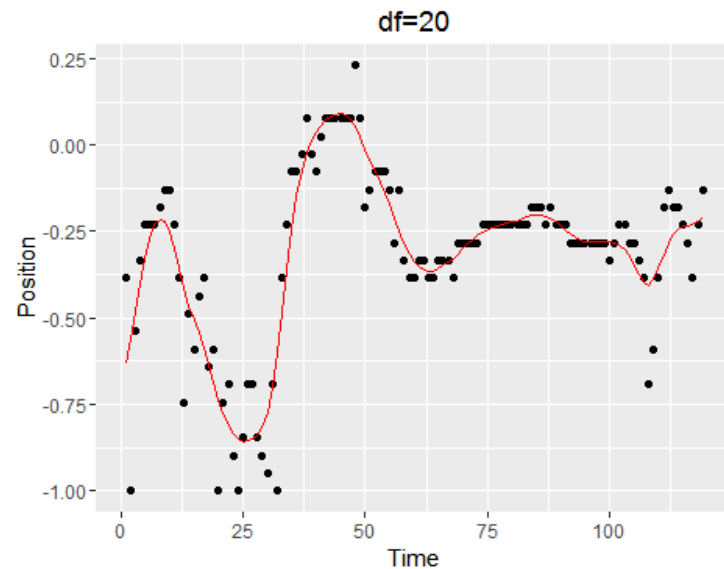
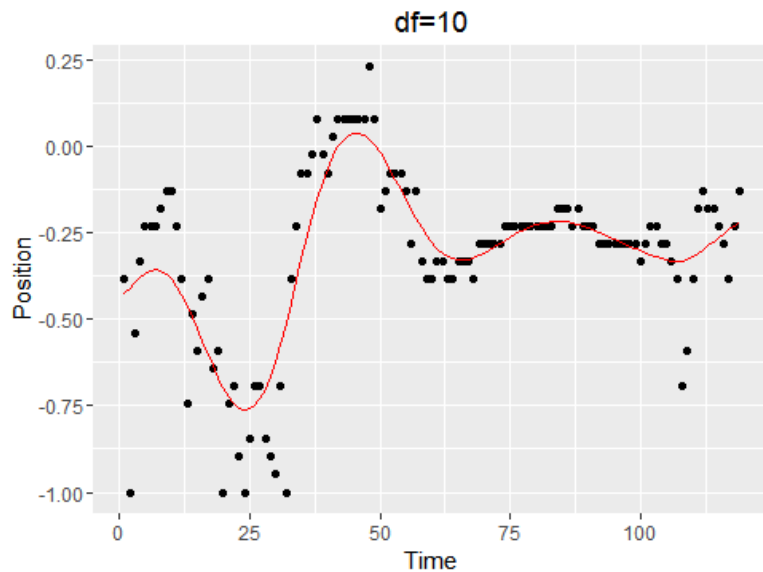
$$f(x) = \beta_0 + \beta^T x + \sum \alpha_j \eta(\|x - x_j\|)$$



# Splines: R code

- Smoothing splines : `smooth.spline()`
- Natural cubic splines: `ns()` in **splines**
- Thin plate splines: `Tps()` in **fields**

```
res1=smooth.spline(data$Time,data$RSS_anchor2,df=10)  
predict(res1,x=data$Time)$y
```



# Generalized additive models

- Model

$$Y \sim EF(\mu, \dots)$$

where

- $g(\mu) = \alpha + s_1(X_1) + s_2(X_2) + s_p(X_p)$
  - $s_i(X)$  - smoothers, normally splines
  - EF – distribution from exponential family
  - $g$  – Link function
- Often linear terms are often included separately

$$EY = \alpha + s_1(X_1) + \dots + s_p(X_p) + \sum_{j=1}^q \beta_j X_{p+j}$$

**Example:** EF= normal, EF=Bernoulli (logistic)

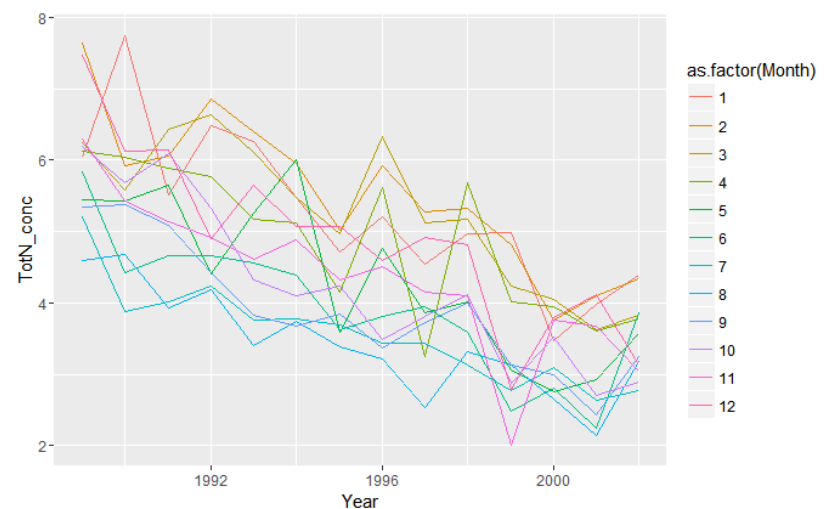
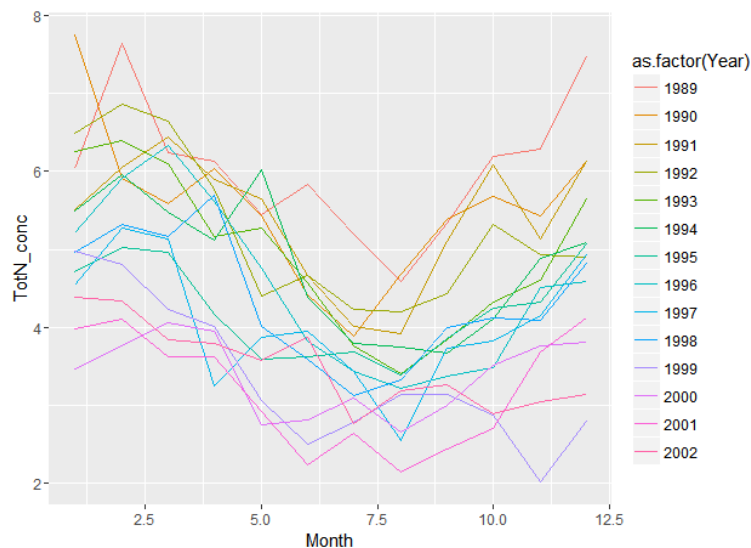
# Generalized additive models

- Sometimes even higher orders are included (thin-plate splines)

$$g(\mu) = \alpha + s_1(X_1) + \dots + s_p(X_p) + \sum_{j=1}^q \beta_j X_{p+j} + s_{12}(X_1, X_2)$$

- Method is reasonable to apply when additivity is observed or admissible

**Example:** Total Nitrogen level in Rhine river



# Estimation of additive models

## Estimation by MLE

$$g(\mu) = \alpha + f_1(x_1) + \dots + f_p(x_p)$$

## The backfitting algorithm for Normal model

1. Initialize:  $\hat{\alpha} = \frac{1}{N} \sum_{i=1}^N y_i, \quad \hat{f}_j \equiv 0, \quad j = 1, \dots, p$

2. Cycle:  $j = 1, \dots, p, 1, \dots, p, \dots, 1, \dots, p$

$$\hat{f}_j \leftarrow s_j \left[ \left\{ y_i - \hat{\alpha} - \sum_{k \neq j} \hat{f}_k(x_{ik}) \right\} \right]$$

$$\hat{f}_j \leftarrow \hat{f}_j - \frac{1}{N} \sum_{i=1}^N \hat{f}_j(x_{ij})$$

$\lambda$  in each term  
can be  
estimated by  
CV



# Generalized additive models

- **Example:** Modelling the concentration of total nitrogen at Lobith on the Rhine
  - There are seasonal trends (GAM reasonable)
  - Variables
    - Nitrogen level
    - Year
    - Month
- R: package **mgcv** (also package **gam**)
  - `gam(formula, family, data, select, method)`
    - Select allows for term (variable) selection
  - `predict()`, `plot()`, `summary()`...
  - `s(k, sp)`
    - `k` should be the same as the amount of **unique values** of this variable in **smoothing splines**
    - `sp` - smoothing penalty.

# Generalized additive models

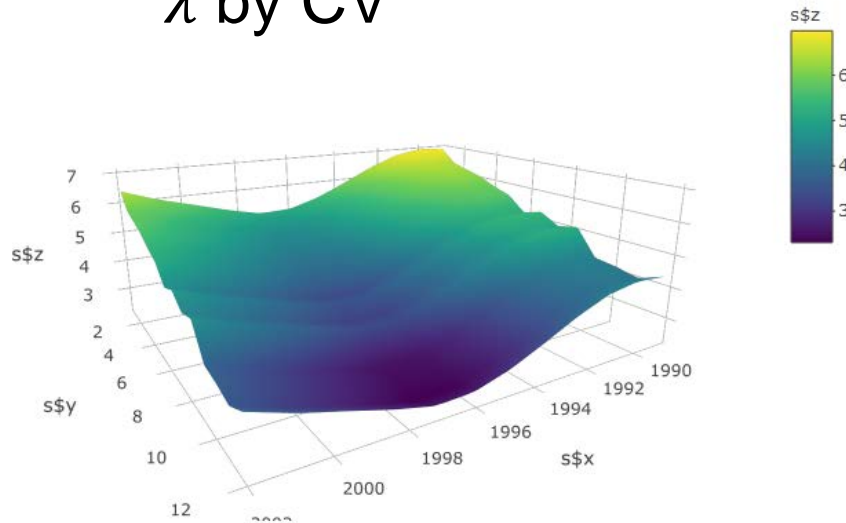
- R code

```
library(mgcv)
library(akima)
library(plotly)
river=read.csv2("Rhine.csv")
res=gam(TotN_conc~Year+Month+s(Year,
k=length(unique(river$Year)))+
      s(Month, k=length(unique(river$Month))),
data=river)

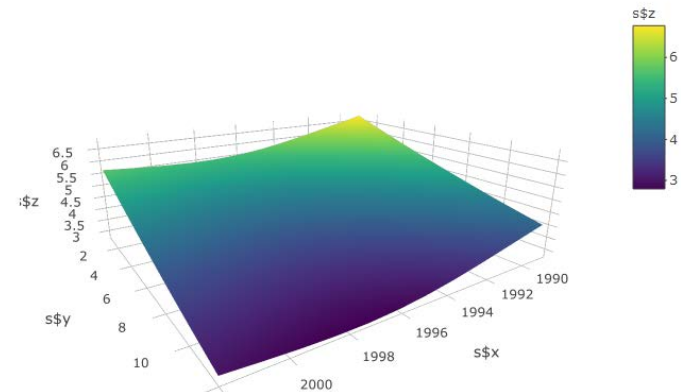
s=interp(river$Year,river$Month, fitted(res))
print(res)
summary(res)
res$sp
plot_ly(x=~s$x, y=~s$y, z=~s$z, type="surface")
```

# Generalized additive models

$\lambda$  by CV



$\lambda = 1$



```
> res$sp
      s(Year)      s(Month)
0.0001853275 0.0060877093
```

# Generalized additive models

```
> summary(res)
```

```
Family: gaussian  
Link function: identity
```

```
Formula:
```

```
TotN_conc ~ Year + Month + s(Year, k = length(unique(river$Year))) +  
  s(Month, k = length(unique(river$Month)))
```

```
Parametric coefficients:
```

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	0.0006039	0.0016063	0.376	0.707
Year	0.0014321	0.0003277	4.370	2.31e-05 ***
Month	0.2471508	0.1004312	2.461	0.015 *

```
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Approximate significance of smooth terms:
```

	edf	Ref.df	F	p-value
s(Year)	11.799	12.751	41.32	<2e-16 ***
s(Month)	4.551	5.819	36.51	<2e-16 ***

```
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Rank: 25/27
```

```
R-sq.(adj) = 0.831   Deviance explained = 84.9%
```

```
GCV = 0.26856   Scale est. = 0.23935   n = 168
```

```
> |
```



# Generalized additive models

- Seeing trend and seasonal pattern

`plot(res)`

