UNIVERSITÀ DEGLI STUDI DI TRIESTE

Dipartimento di Scienze Economiche, Aziendali, Matematiche e Statistiche "Bruno de Finetti"

Non parametric statistics

Spline as mixed model

Francesco Pauli

A.A. 2015/2016

Penalized spline as mixed effect model

We see that the penalized spline model can be written as a mixed effect model, consider again

$$f(x) = \beta_1 + \beta_2 x + \ldots + \beta_{p+1} x^p + \sum_{k=1}^{K} b_k (x - \kappa_k)_+^p$$

$$\boldsymbol{\beta} = (\beta_1, \ldots, \beta_{p+1})^T \in \mathbb{R}^{p+1} \qquad \boldsymbol{b} = (b_1, \ldots, b_K)^T \in \mathbb{R}^K$$

$$X = \begin{bmatrix} 1 & x_1 & \ldots & x_1^p \\ \vdots & \vdots & & \vdots \\ 1 & x_n & \ldots & x_n^p \end{bmatrix} \qquad Z = \begin{bmatrix} (x_1 - \kappa_1)_+^p & \ldots & (x_1 - \kappa_K)_+^p \\ \vdots & & \vdots \\ (x_n - \kappa_1)_+^p & \ldots & (x_n - \kappa_K)_+^p \end{bmatrix}$$

then

$$\boldsymbol{f} = X\boldsymbol{\beta} + Z\boldsymbol{b}$$

Penalized spline as mixed effect model (continua)

Consider now the penalized fitting criterion:

$$||\boldsymbol{y} - X\boldsymbol{\beta} - Z\boldsymbol{b}||^2 + \lambda||\boldsymbol{b}||^2$$

this is proportional to

$$\frac{1}{\sigma^2}||\boldsymbol{y} - X\boldsymbol{\beta} - Z\boldsymbol{b}||^2 + \frac{\lambda}{\sigma^2}||\boldsymbol{b}||^2$$

which is minus the log-likelihood of the model

$$\boldsymbol{y} = X\boldsymbol{\beta} + Z\boldsymbol{b} + \varepsilon$$

$$\begin{bmatrix}\boldsymbol{b}\\ \varepsilon\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}\sigma_b^2 I & 0\\ 0 & \sigma^2 I\end{bmatrix}\right)$$

where $\sigma_b^2 = \sigma^2/\lambda$.
This means that all the tools and methods developed for mixed effect model can be used to estimate splines.

Classical inference

We do not consider classical inference, suffices to say that all methods and software for mixed models can be used.

Bayesian inference

The model is defined by

$$\boldsymbol{y}|\boldsymbol{\beta}, \boldsymbol{b}, \sigma^2 \sim \mathcal{N}(X\boldsymbol{\beta} + Z\boldsymbol{b}, \sigma^2 I)$$

$$\boldsymbol{b}|\sigma_b^2 \sim \mathcal{N}(0, \sigma_b^2 I)$$

and the prior distributions

$$\boldsymbol{\beta} \sim \mathcal{N}(0, .)$$

$$\sigma^2 \sim IG(., .)$$

$$\sigma_b^2 \sim IG(., .)$$

The prior distributions above are conjugate and allow to write some full conditionals, however, we are not interested in doing this, also because we want to be able to use alternative priors.

Indice

In practice

We will estimate a spline model for the LIDAR data using Stan, in particular we will

- learn how to call Stan from within R using *rstan*
- estimate a spline through the implementation above
- represent the spline in R
- Stan was developed by Andrew Gelman et al
- (see `http://https://mc-stan.org/`).
- Can be used from within R using the package rstan or rstanarm

```
library(rstan)
```

Stan

Stan is a state-of-the-art platform for statistical modeling and high-performance statistical computation. Thousands of users rely on Stan for statistical modeling, data analysis, and prediction in the social, biological, and physical sciences, engineering, and business.

- Users specify log density functions in Stan's probabilistic programming language and get:
- full Bayesian statistical inference with MCMC sampling (NUTS, HMC)
- approximate Bayesian inference with variational inference (ADVI)
- penalized maximum likelihood estimation with optimization (L-BFGS)

Stan's math library provides differentiable probability functions & linear algebra (C++ autodiff). Additional R packages provide expression-based linear modeling, posterior visualization, and leave-one-out cross-validation.

Stan from R: 1st step, define quantities

We need to prepare the sample, that is, the vector y and the matrices X and Z, so, using the data of the motivating example

```
lidar=read.table("lidar.dat",header=TRUE)
x=(lidar$range-mean(lidar$range))/sd(lidar$range)
y=lidar$logratio
```

Let $p = 3$, the matrix X is then

```
X=cbind(1,x,x^2,x^3)
dim(X)
```

Stan from R: 1st step, define quantities (continua)

Define the knots as

```r
knots=seq(min(x),max(x),length=12)[2:11]
```

(We do not want to define knots equal to the minimum or the maximum, why?)
We can then define the matrix Z by

```r
Z=matrix(NA,221,20)
for (i in 1:221) for (j in 1:20)
  Z[i,j]=ifelse(x[i]-knots[j]>0,(x[i]-knots[j])^3,0)
```

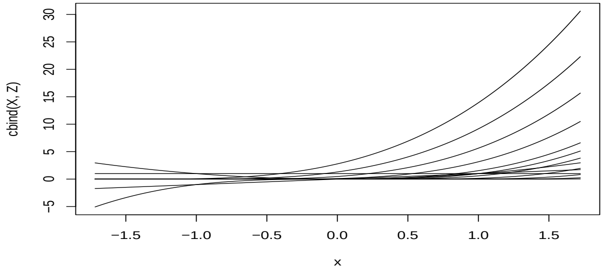
or, much more efficiently

```r
Z=outer(x,knots,
        FUN=function(x,y) ifelse(x-y>0,(x-y)^3,0))
```

Stan from R: 1st step, define quantities (continua)

We can have a look at the basis

```
matplot(x,cbind(X,Z),type="l",col="black",lty=1)
```



Stan from R: 1st step, define quantities (continua)

Data must now be ‘passed’ to Stan by creating a named list

```
dati1=list(N=length(y),
           p=3,
           K=length(knots),
           y=y,
           X=X,
           Z=Z)
```

2nd step define the model

```
data {
  int<lower=0> N;
  int<lower=0> p;
  int<lower=0> K;
  vector[N] y;
  matrix[N, 4] X;
  matrix[N, K] Z;
}
parameters {
  vector[4] beta;
  vector[K] b;
  real<lower=0> sigma2;
  real<lower=0> sigmab2;
}
model {
  for (i in 1:N) {
   y[i] ~ normal( X[i,]*beta[1:(p+1)] + Z[i,]*b[1:K],sqrt(sigma2));
  }
  for (k in 1:K){
    b[k] ~ normal(0,sqrt(sigmab2));
  }
  for (j in 1:(p+1)) {
    beta[j] ~ normal( 0  ,1000);
  }
  sigma2 ~ inv_gamma(0.001,0.001);
  sigmab2 ~ inv_gamma(0.001, 0.001);
}
```

Stan from R: 3rd step, do the MCMC!

Do it!

```r
fit_spline1 <- stan(file = 'spline1.stan', data = dati1, cores=parallel::detectCores())
```

Stan from R: 5th step, what's in there?

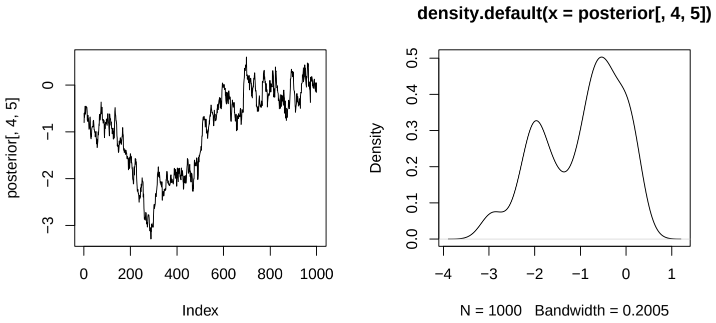
```
print(fit_spline1)

Inference for Stan model: spline1.
4 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=4000.

          mean se_mean   sd   2.5%    25%    50%
beta[1]  -0.26    2.01 3.29  -7.58  -1.82   0.18
beta[2]  -0.30    4.24 6.92 -15.70  -3.79   0.78
beta[3]  -0.12    2.97 4.84 -10.77  -2.61   0.72
beta[4]  -0.01    0.69 1.13  -2.50  -0.58   0.21
b[1]      0.13    0.77 1.31  -2.18  -0.76   0.08
b[2]      0.05    0.12 0.87  -1.56  -0.53   0.01
b[3]     -0.53    0.08 0.67  -1.77  -0.99  -0.54
b[4]     -0.29    0.06 0.60  -1.64  -0.64  -0.23
b[5]      0.77    0.12 0.73  -0.98   0.37   0.81
b[6]      1.26    0.38 1.06  -0.18   0.53   0.97
b[7]     -1.58    0.37 1.09  -4.31  -2.24  -1.33
b[8]     -0.37    0.12 0.70  -1.61  -0.84  -0.39
b[9]      0.74    0.10 0.80  -0.74   0.15   0.71
b[10]    -0.10    0.11 1.14  -2.49  -0.73  -0.07
sigma2    0.01    0.00 0.00   0.01   0.01   0.01
sigmab2   1.87    0.55 2.49   0.19   0.61   1.12
lp__    441.27    0.72 3.58 433.85 438.91 441.50
           75%  97.5% n_eff Rhat
beta[1]   2.14   5.29     3 2.52
beta[2]   4.74  10.93     3 2.48
beta[3]   3.34   7.65     3 2.42
beta[4]   0.77   1.83     3 2.32
b[1]      0.70   3.04     3 2.38
b[2]      0.61   1.83    49 1.14
```

A look inside

```
posterior=as.array(fit_spline1)
str(posterior)

 num [1:1000, 1:4, 1:17] -1.239 -1.039 -0.807 -0.844 -0.285 ...
 - attr(*, "dimnames")=List of 3
  ..$ iterations: NULL
  ..$ chains    : chr [1:4] "chain:1" "chain:2" "chain:3" "chain:4"
  ..$ parameters: chr [1:17] "beta[1]" "beta[2]" "beta[3]" "beta[4]" ...
```

Convergence of chains

```
par(mfrow=c(1,2))
plot(posterior[,4,5],type="l")
plot(density(posterior[,4,5]))
```



Unfortunately, the chains are not satisfying, this is due to the correlation among the b_i's.

Other basis: radial basis

The basis is

$$f(x) = \beta_1 + \beta_2 x + \sum_{k=1}^{K} b_k |x - \kappa_k|^3$$

the penalization is given by

$$\sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \theta^T D \theta$$

where

$$D = \begin{bmatrix} 0_{2\times 2} & 0_{2\times K} \\ 0_{K\times 2} & \Omega \end{bmatrix}$$

and

$$\Omega_{ij} = |\kappa_i - \kappa_j|^3, \quad i, j = 1, \ldots, K$$

Other basis: radial basis (continua)

As the objective function is written now, we do not recognize the likelihood of a mixed model

$$\sum_{i=1}^{n}(y_i - f(x_i))^2 + \lambda\theta^T D\theta$$

basically because $\theta^T D\theta$ is not the kernel of a multivariate uncorrelated normal distribution.
In order to write the model as a mixed effect model we need a transformation of the b parameters.

Other basis: radial basis (continua)

Let then

$$y = X\beta + Z\Omega^{-1/2}\Omega^{1/2}\boldsymbol{b} + \varepsilon$$

and

$$Z_u = Z\Omega^{-1/2}, \quad \boldsymbol{u} = \Omega^{1/2}\boldsymbol{b}$$

Then the model is

$$y = X\beta + Z_u\boldsymbol{u} + \varepsilon$$

and the penalization is

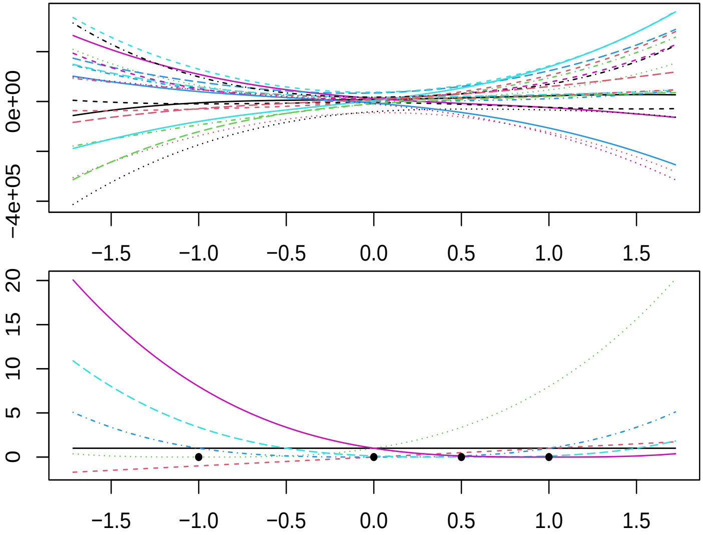
$$\boldsymbol{b}^T\Omega\boldsymbol{b} = \boldsymbol{u}^T I_K \boldsymbol{u}$$

Other basis: radial basis (continua)

Note that the explanation above, although pretty standard in the literature, overlooks some details, which can be read in French et al (2001), in particular

- the penalization matrix Ω is an approximation to the appropriate penalization matrix (and is not even positive semidefinite);
- as a consequence, its square root is not well defined (and will be computed using singular value decomposition rather than spectral value decomposition);
- French et al 2001 show that the estimates obtained using the proper matrix are equivalent to those obtained using the above defined matrix.

Other basis: radial basis (continua)



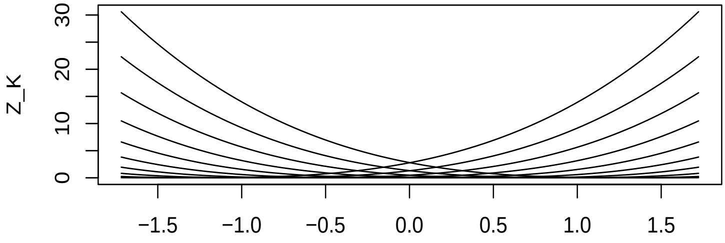
Estimate with TPS

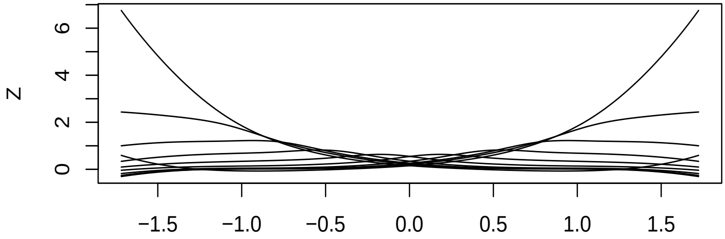
Matrices X and Z change:

```
X=cbind(1,x)
```

```
Z_K<-(abs(outer(x,knots,"-")))^3
OMEGA_all<-(abs(outer(knots,knots,"-")))^3
svd.OMEGA_all<-svd(OMEGA_all)
sqrt.OMEGA_all<-t(svd.OMEGA_all$v %*%
                    (t(svd.OMEGA_all$u)*sqrt(svd.OMEGA_all$d)))
Z<-t(solve(sqrt.OMEGA_all,t(Z_K)))
```

Estimate with TPS (continua)





Estimate with TPS (continua)

```
model {
  for (i in 1:N) {
   y[i] ~ normal( X[i,]*beta[1:2] + Z[i,]*b[1:K],sigma);
  }
  for (k in 1:K){
    b[k] ~ normal(0,sigmab);
  }
  for (j in 1:2) {
    beta[j] ~ normal( 0  ,1000);
  }
}
```

Estimate with TPS (continua)

'Pass' the data to Stan

```
dati2 <- list(N=length(y),p=3,
                         K=length(knots),
                         y=y,X=X,Z=Z)
```

Estimate with TPS (continua)

Do it!

```
fit_spline2 <- stan(file = 'spline2.stan', data = dati2, cores=
posterior=as.array(fit_spline2)
```

Check convergence

```
launch_shinystan(fit_spline2)
```

From MCMC chains to splines

The chains stored in *posterior* are now regarded as a sample from the posterior distribution, in particular

```
str(posterior)
```

is an array matrix where each column is a sample from the posterior of a parameter

```
dimnames(posterior)
```

From MCMC chains to splines (continua)

In order to merge the 4 chains we can use `extract`

```
posterior=extract(fit_spline2)
```

is a named list where each element contains the samples for a paramter (a vector or possibily a matrix)

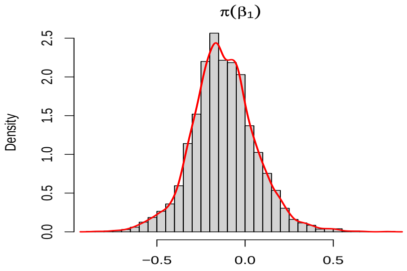
```
str(posterior)
```

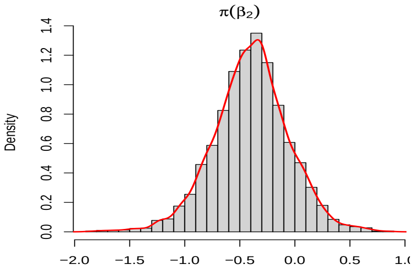
From MCMC chains to splines (continua)

`posterior$ beta` contains a sample from the posterior of β_1 and β_2, let

```
beta=posterior$beta
dim(beta)
```

We denote as $\beta_{i,j}$, $i = 1, \ldots, B$, $j = 1, 2$ the MCMC-sampled values.



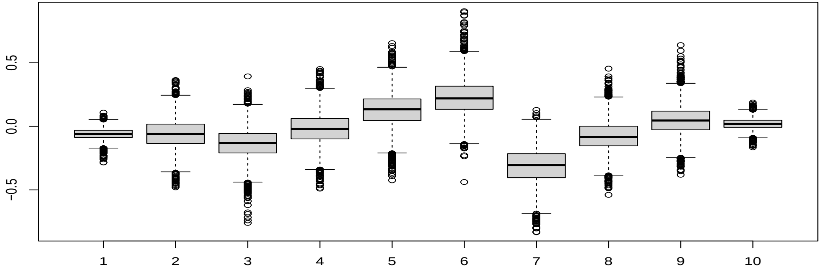


From MCMC chains to splines (continua)

Analogously

```
b=posterior$b
dim(b)
```

We denote as $b_{i,j}$, $i = 1, \ldots, B$, $j = 1, \ldots, 10$ the MCMC-sampled values.



From MCMC chains to splines (continua)

However, we are not directly interested in the distributions of these parameters, but rather in the distribution of the spline values.
How do we obtain the distribution of $f(x_h)$?

- From the definition of the spline

$$f_h = f(x_h) = \beta_1 + \beta_2 x_i + \sum_{k=1}^{K} b_k |x_h - \kappa_k|^3$$

- this is a straightforward transformation of the r.v. $\beta_1,\ \beta_2,\ b_1, \dots, b_K$
- hence, we can obtain a sample of values for $f(x_h)$

$$f_{i,h} = [f(x_h)]_i = \beta_{i,1} + \beta_{i,2} x_i + \sum_{k=1}^{K} b_{i,k} |x_h - \kappa_k|^3$$

From MCMC chains to splines (continua)

The computation is straightforward

```
f=X%*% t(beta) + Z %*% t(b)
dim(f)
```

Each column is a spline function, sampled from the 'posterior distribution of the splines'
The h-th row of the matrix `f` is now a sample from the posterior distribution of the spline value in x_h

$$\pi(f(x_h)|\boldsymbol{y})$$

From MCMC chains to splines (continua)

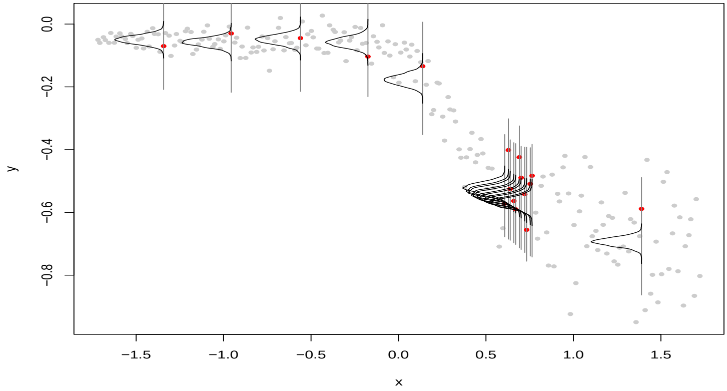
Each column is a spline function, sampled from the ‘posterior distribution of the splines’

```
Error: <text>:3:29: ')' inatteso
2: plot(x,y,pch=20)
3: lines(x,f[,1],col=gray(0.8)))
                               ^
```

From MCMC chains to splines (continua)

The h-th row of the matrix `f` is now a sample from the posterior distribution of the spline value in x_h

$$\pi(f(x_h)|\boldsymbol{y})$$



From MCMC chains to splines (continua)

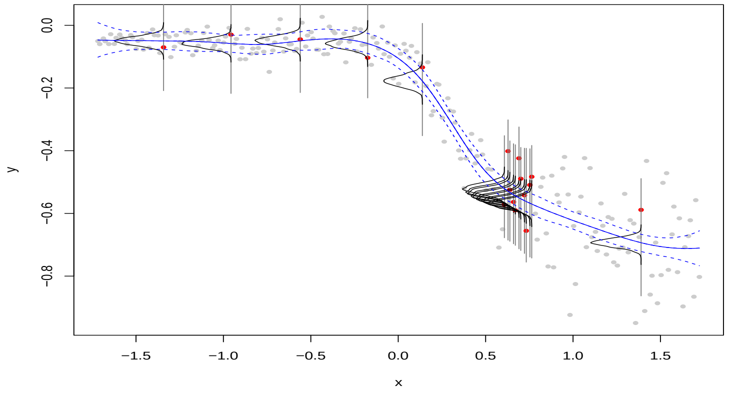
From this, we can compute point estimates, such as the medians

```
pe=apply(f,1,median)
```

as well as bounds of credibility intervals

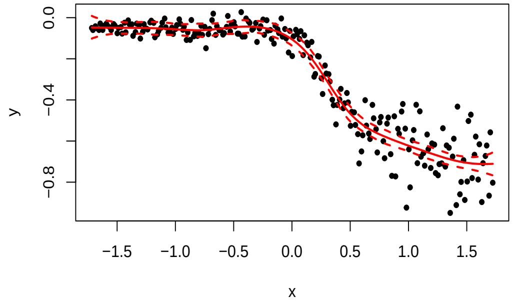
```
ci=apply(f,1,quantile,probs=c(0.025,0.5,0.975))
```

From MCMC chains to splines (continua)



From MCMC chains to splines (continua)

```
plot(x,y,pch=20)
lines(x,pe,col="red")
matlines(x,t(ci),lwd=2,col="red",lty=c(2,1,2))
```



Exercises

1. Plot the posterior distribution for σ_b^2, also compute a point and interval estimate.
2. Do the same for σ_b.
3. Obtain the estimate and c.i. for $x = 2$ (a point different from (all) the x_h).

Indice

Mixing

A good mixing of the chains is essential to guarantee that the parameter space is fully explored (and so credibility intervals are reliable).

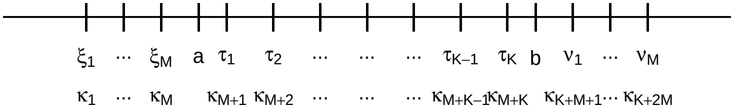
some advice to improve mixing

- standardize covariates
- (hierarchical standardization if it is the case)
- choose a good spline basis (thin plate are good, truncated polynomials are not),
- other good basis: B-splines, natural cubic splines
- a good basis reduce posterior correlation of the coefficients

B-spline: basis construction

- let $\tau_1 < \ldots < \tau_K$ be the internal nodes;
- let $[a, b]$ be the range on which we are interested ($a < \tau_1$, $b > \tau_K$);
- fix, arbitrarily, $\xi_1 \leq \ldots \xi_M \leq a$ and $\nu_M \geq \ldots \geq \nu_1 \geq b$ (one can set $\xi_i = a$ and $\nu_i = b$);
- we then have a sequence $\kappa_1, \ldots, \kappa_{K+2M}$.

B-spline: basis construction (continua)


ξ_1 ⋯ ξ_M a τ_1 τ_2 ⋯ ⋯ ⋯ τ_{K-1} τ_K b ν_1 ⋯ ν_M
κ_1 ⋯ κ_M κ_{M+1} κ_{M+2} ⋯ ⋯ ⋯ κ_{M+K-1} κ_{M+K} κ_{K+M+1} ⋯ κ_{K+2M}

B-spline: basis construction (continua)

Let then $B_{i,m}$ denote the i-th basis function of order $m < M$,
$i = 1, \ldots, K + 2M - m$, this is defined recursively by

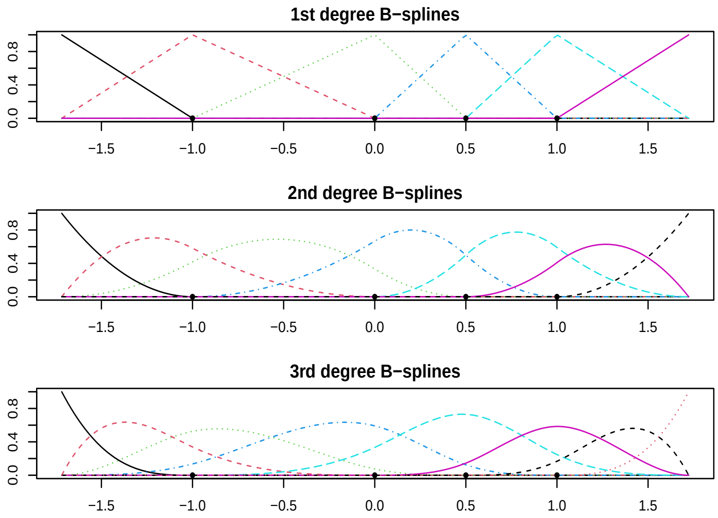
$$B_{i,1}(x) = \begin{cases} 1 & \text{if } \kappa_i \leq x < \kappa_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

for $i = 1, \ldots, K + 2M - 1$ and

$$B_{i,m}(x) = \frac{x - \kappa_i}{\kappa_{i+m-1} - \kappa_i} B_{i,m-1}(x) + \frac{\kappa_{i+m} - x}{\kappa_{i+m} - \kappa_{i+1}} B_{i+1,m-1}(x)$$

$i = 1, \ldots, K + 2M - m$.
For $M = 4$ one obtains $K + 4$ cubic splines.

Other basis: B-splines



B-spline: penalty matrix

Setting $M = 4$, the penalty matrix is defined as

$$\Omega_{ij} = \int_a^b B_i''(x) B_j''(x) dx$$

Wand and Ormerod (2009) obtained formulas for calculating Ω in practice

$$\Omega = (\tilde{B}'')^T \mathrm{diag}(\mathrm{w}) \tilde{B}''$$

where

$$[\tilde{B}'']_{ij} = \tilde{B}_j(\tilde{x}_i) \in \mathcal{M}_{3(K+7)\times(K+4)}$$

$$\tilde{x} = \left(\kappa_1, \frac{\kappa_1 + \kappa_2}{2}, \kappa_2, \ldots, \kappa_{K+7}, \frac{\kappa_{K+7} + \kappa_{K+8}}{2}, \kappa_{k+8}\right)$$

$$\mathrm{w} = \left(\frac{1}{6}(\Delta\kappa)_1, \frac{4}{6}(\Delta\kappa)_1, \frac{1}{6}(\Delta\kappa)_1, \ldots, \frac{1}{6}(\Delta\kappa)_{K+7}, \frac{4}{6}(\Delta\kappa)_{K+7}, \frac{1}{6}(\Delta\kappa)_{K+7}\right)$$

where $(\Delta\kappa)_h = \kappa_{h+1} - \kappa_h$.

B-spline: mixed model formulation

It is assumed that a matrix $L \in \mathcal{M}_{(K+4)\times(K+4)}$ exists such that

$$L^T \Omega L = \begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix}$$

This in fact can be obtained through a spectral decomposition, let

$$\Omega = U \mathrm{diag}(d) U^T$$

since $\mathrm{rank}(\Omega) = K + 2$ there are $K + 2$ positive eigenvalues, let d_Z be the vector and let U_Z be the corresponding eigenvectors, then let

$$L = [U_X | U_Z \mathrm{diag}(d_Z^{-1/2})]$$

the mixed model formulation is then obtained by setting

$$X = BU_X, \quad Z = BU_Z \mathrm{diag}(d_Z^{-1/2})$$

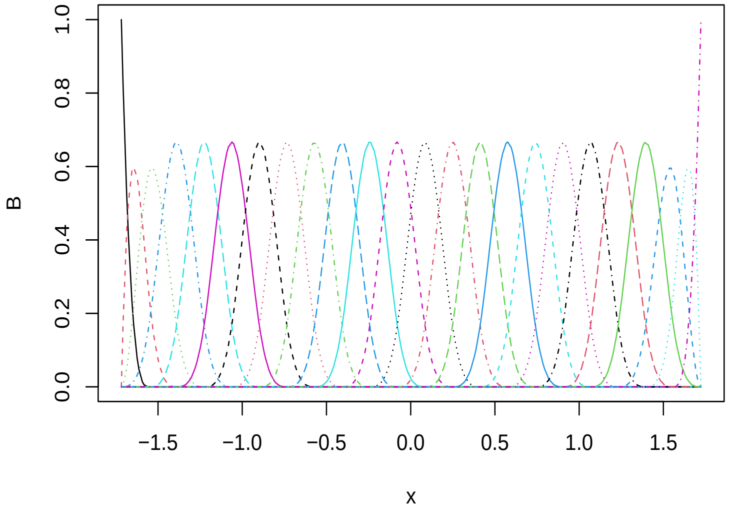
(but note that one can set $X = [1\ \boldsymbol{x}]$ since BU_X is a basis for straight lines)

Inference with B-splines

We compute the B matrix

```
nodi <- seq(min(x),max(x),length=22)[2:21]
B <- bs(x,knots=nodi,degree=3,
        Boundary.knots=c(min(x),max(x)),
        intercept=TRUE)
```

Inference with B-splines (continua)



Inference with B-splines (continua)

```
formOmega <- function(a,b,intKnots) {
  allKnots <- c(rep(a,4),intKnots,rep(b,4))
  K <- length(intKnots) ; L <- 3*(K+8)
  xtilde <- (rep(allKnots,each=3)[-c(1,(L-1),L)]+
    rep(allKnots,each=3)[-c(1,2,L)])/2
  wts <- rep(diff(allKnots),each=3)*rep(c(1,4,1)/6,K+7)
  Bdd <- spline.des(allKnots,xtilde,
                    derivs=rep(2,length(xtilde)),
                    outer.ok=TRUE)$design
  Omega <- t(Bdd*wts) %*% Bdd
  return(Omega)
}
```

Inference with B-splines (continua)

```
Omega <- formOmega(min(x),max(x),nodi)
eigOmega <- eigen(Omega)
indsZ <- 1:(20+2)
UZ <- eigOmega$vectors[,indsZ]
LZ <- t(t(UZ)/sqrt(eigOmega$values[indsZ]))
```

Perform a stability check:

```
indsX <- (20+3):(20+4)
UX <- eigOmega$vectors[,indsX]
L <- cbind(UX,LZ)
stabCheck <- t(crossprod(L,t(crossprod(L,Omega))))
if (sum(stabCheck^2) > 1.0001*(20+2)) print("WARNING: NUMERICAL
```

Inference with B-splines (continua)

Obtain the matrices

```
X <- cbind(rep(1,length(x)),x)
Z <- B%*%LZ
```

Inference with B-splines (continua)

```
model<- 'model {
  for (i in 1:N) {
   y[i] ~ dnorm(mu[i],tau)
   mu[i] <- X[i,] %*% beta[1:2] + Z[i,] %*% b[1:K]
  }
  for (k in 1:K){
    b[k] ~ dnorm(0,taub)
  }
  taub ~ dgamma(1.0E-3,1.0E-3)
  tau ~ dgamma(1.0E-3,1.0E-3)
  for (j in 1:2) {
    beta[j] ~ dnorm( 0  ,1.0E-3)
  }
}'
```

Inference with B-splines (continua)

'Pass' the data to Stan

```
dati3 <- list(N=length(y),
                         K=ncol(Z),
                          y=y,X=X,Z=Z)
```

Do it!

```
fit_spline3 <- stan(file = 'spline3.stan', data = dati3,
                    cores=parallel::detectCores())
```

```
posterior=extract(fit_spline3)
```

Inference with B-splines (continua)

The first two columns are then a sample from the distributions of β_1 and β_2, while the following 20 are a sample from the distribution of $\boldsymbol{b}$, we extract them

```
beta=posterior$beta
dim(beta)
b=posterior$b
dim(b)
```

```
f=X%*% t(beta) + Z %*% t(b)
dim(f)
```

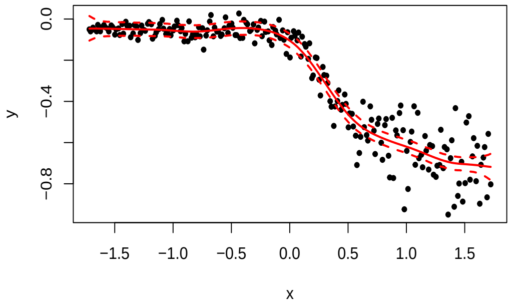
Inference with B-splines (continua)

From this, we can compute point estimates, such as the medians

```
pe=apply(f,1,median)
ci=apply(f,1,quantile,probs=c(0.025,0.5,0.975))
```

```
plot(x,y,pch=20)
lines(x,pe,col="red")
matlines(x,t(ci),lwd=2,col="red",lty=c(2,1,2))
```

Inference with B-splines (continua)



Indice

Prior specification

We specified the following prior distributions, regarding them as scarcely informative

$$\beta \sim \mathcal{N}(0, 10^3)$$

$$\tau = 1/\sigma^2 \sim G(10^{-3}, 10^{-3})$$

$$\tau_b = 1/\sigma_b^2 \sim G(10^{-3}, 10^{-3})$$

The chosen prior for β is generally not disputed (as long as the variance is big enough, also 10^6 is suggested).

Prior specification for variance (precision) parameters

The traditional choice is

$$\tau_\bullet = 1/\sigma^2_\bullet \sim G(A_\bullet, B_\bullet)$$

- A and B are chosen so that the variance is high, usually they are equal so that the mean is 1 (the aim is to have a prior which does not influence the results).
- this choice is advocated by Crainiceanu, Ruppert and Wand (2005) provided some rules of thumb concerning the values of A and B are satisfied (see paper).
- according to Gelman (2006) the IG prior is informative whenever the variance is very low.

Prior specification for variance (precision) parameters (continua)

Gelman (2006) suggests the following alternative choices for a prior on σ

- uniform (improper)
- uniform on a wide range
- half-normal centered at 0 with standard deviation set to a high value such as 100
- half-Cauchy
- half-t

Indice

Generalization to non gaussian data

The extension of the methods we outlined to non gaussian data is very simple, also because we work with the Bayesian paradigm.

- For gaussian data, the spline is estimated as a Linear Mixed Model.
- For non gaussian data, the spline is estimated as a Generalized Linear Mixed Model.
- The spline is represented as seen for gaussian data.
- Closed form solutions seen above can not be found, but this is not very important in the bayesian paradigm.

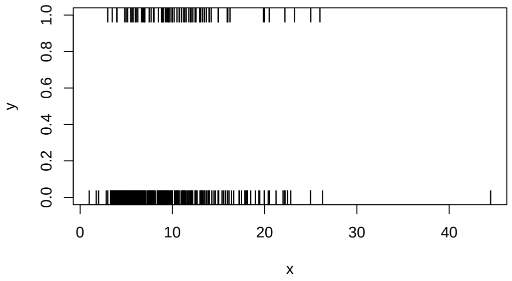
Model for binary data

We consider a sample of workers for which we observed

- their wage (dollars per hour)
- whether they belong to a union

The aim is to estimate the probability that a worker belongs to a union for different wages.

Model for binary data (continua)



Model for binary data (continua)

With respect to the model written above we only change the first part

$$Y_i \sim \text{Bernoulli}(p(x_i))$$

$$\log\left(\frac{p(x_i)}{1-p(x_i)}\right) = X\boldsymbol{\beta} + Z\boldsymbol{b}$$

then we assume

$$\boldsymbol{b} \sim \mathcal{N}\left(0, \sigma_b^2\right)$$

$$\sigma_b \sim h\mathcal{N}(0, 10^3)$$

$$\beta_j \sim N(0, 10^6), \quad j = 1, 2$$

Note that we changed the prior for the variance of the coefficient of the basis functions.

0: Read the data

```r
tu=read.table('trade.union.txt',header=TRUE)
x=tu$wage
y=tu$union.member
```

1: Build matrices

```
knots=seq(min(x),max(x),length=22)[2:21]
X=cbind(1,x)
Z_K<-(abs(outer(x,knots,"-")))^3
OMEGA_all<-(abs(outer(knots,knots,"-")))^3
svd.OMEGA_all<-svd(OMEGA_all)
sqrt.OMEGA_all<-t(svd.OMEGA_all$v %*%
                    (t(svd.OMEGA_all$u)*sqrt(svd.OMEGA_all$d)))
Z<-t(solve(sqrt.OMEGA_all,t(Z_K)))
```

```
dati4 <- list(N=length(y),K=length(knots),
                         y=y,X=X,Z=Z)
```

2: write the model

```
model {
  for (i in 1:N) {
   y[i] ~ bernoulli_logit(X[i,] * beta[1:2] + Z[i,] * b[1:K]);
}
for (k in 1:K){
  b[k] ~ normal(0,sigmab);
}
sigmab ~ normal(0,1000);
for (j in 1:2) {
  beta[j] ~ normal( 0  ,1000);
}
}
```

3: Stan

Do the MCMC

```
fit_spline4 <- stan(file = 'spline4.stan', data = dati4,
                    cores=parallel::detectCores())
posterior=extract(fit_spline4)
```

Check convergence

```
launch_shinystan("fit_spline4.Rdata")
```

From MCMC chains to splines

As before, we extract the relevant samples from the posterior
We then obtain the posterior for the spline values

```
logitp=X%*% t(posterior$beta) + Z %*% t(posterior$b)
```

and then we transform back to the scale of the probability

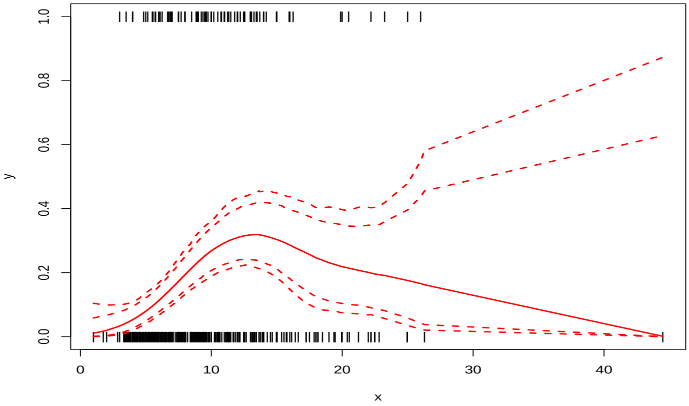
```
p=exp(logitp)/(1+exp(logitp))
```

From MCMC chains to splines (continua)

Finally we plot the spline and the credibility intervals

```
par(mfrow=c(1,1))
plot(x,y,pch="|")
sl=sort.list(x)
x.sorted=x[sl]
p=p[sl,]
ci=apply(p,1,quantile,
         probs=c(0.005,0.025,0.5,0.975,0.995))
matlines(x.sorted,t(ci),lwd=2,
         col="red",lty=c(2,2,1,2,2))
```

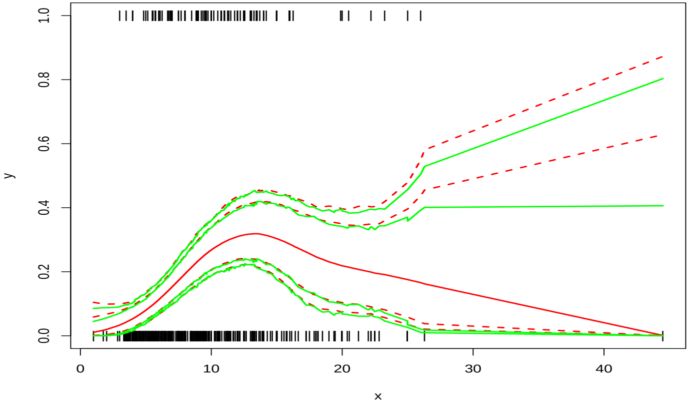
From MCMC chains to splines (continua)



We can also compute the HPD intervals

```
library(coda)
hpd.int=HPDinterval(as.mcmc(t(p)),prob=c(0.95))
matlines(x.sorted,hpd.int,col="green",lwd=2,lty=1)
hpd.int=HPDinterval(as.mcmc(t(p)),prob=c(0.99))
matlines(x.sorted,hpd.int,col="green",lwd=2,lty=1)
```

Intervals



Compute the spline on new x values

Define the new values

```
x.new=seq(0,50,length=50)
```

then compute the relevant matrices

```
X.new=cbind(1,x.new)
Z_K<-(abs(outer(x.new,knots,"-")))^3
OMEGA_all<-(abs(outer(knots,knots,"-")))^3
svd.OMEGA_all<-svd(OMEGA_all)
sqrt.OMEGA_all<-t(svd.OMEGA_all$v %*%
                    (t(svd.OMEGA_all$u)*sqrt(svd.OMEGA_all$d)))
Z.new<-t(solve(sqrt.OMEGA_all,t(Z_K)))
```

Compute the spline on new x values (continua)

obtain the posterior for the spline

```
logitp=X.new%*% t(posterior$beta) + Z.new %*% t(posterior$b)
p=exp(logitp)/(1+exp(logitp))
```

Eventually plot the intervals

```
plot(x,y,pch="|")
matlines(x.new,t(apply(p,1,quantile,
                       probs=c(0.025,0.5,0.975))),
         lwd=2,col="blue",lty=c(2,1,2))
```

Exercises

The data frame contains also other information

- years.educ number of years of education.
- south indicator of living in southern region of U.S.A.
- female gender indicator: 0=male,1=female.
- years.experience number of years of work experience
- age age in years.
- race 1=black, 2=Hispanic, 3=white.
- occupation 1=management, 2=sales, 3=clerical, 4=service, 5=professional, 6=other.
- sector 0=other, 1=manufacturing, 2=construction.
- married indicator of being married: 0=unmarried, 1=married.

Exercises (continua)

Modify (and re-estimate) the model in order to

1. allow for an effect of race (only on the intercept)
   - see variable race 1=black, 2=Hispanic, 3=white.
2. allow for a linear effect of years.experience
   - years.experience number of years of work experience
3. allow for a non-linear effect of years.experience

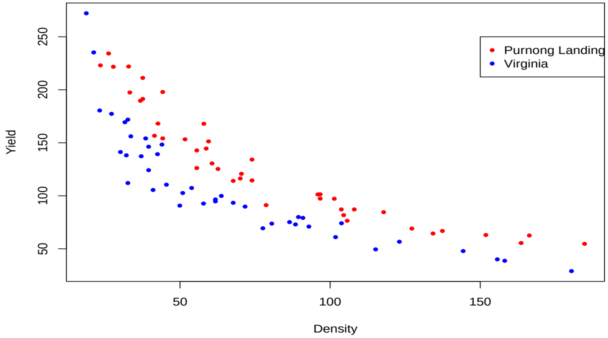
Indice

Model with interaction

- It is relatively easy to extend the model to allow for interactions, specifying different curves for different subpopulations.
- For example, we observe, for 2 locations (Purnong Landing and Virginia), the density and yield of onions plants.
- (Note that this is an example from RWC but unlike them we do not log-transform the response.)
- We denote as y_{ji} the yield of the i-th plant, $i = 1, \ldots, n_j$ in the j-th location, $j = 1, 2$; x_{ij} is the density.
- We specify a model which allows for two different splines

$$y_{ji} = f(x_{ji}) + g_j(x_{ji}) + \varepsilon_{ji}$$

Model with interaction (continua)



Model for onions yield

First, consider a model with a common spline

$$y_{ji} = f(x_{ji}) + \varepsilon_{ji}$$

Let then $(n = n_1 + n_2)$

$$\boldsymbol{y}^T = (y_{11}, \ldots, y_{1n_1}, y_{21}, \ldots, y_{2n_2}) \in \mathbb{R}^n$$

$$X^T = \begin{bmatrix} 1 & \ldots & 1 & 1 & \ldots & 1 \\ x_{11} & \ldots & x_{1n_1} & x_{21} & \ldots & x_{2n_2} \end{bmatrix}$$

let $(\kappa_1, \ldots, \kappa_K)$ be the nodes and so

$$Z = \begin{bmatrix} |x_{11} - \kappa_1|^3 & \ldots & |x_{11} - \kappa_K|^3 \\ \ldots & \ldots & \ldots \\ |x_{1n_1} - \kappa_1|^3 & \ldots & |x_{1n_1} - \kappa_K|^3 \\ |x_{21} - \kappa_1|^3 & \ldots & |x_{21} - \kappa_K|^3 \\ \ldots & \ldots & \ldots \\ |x_{2n_2} - \kappa_1|^3 & \ldots & |x_{2n_2} - \kappa_K|^3 \end{bmatrix} \in \mathcal{M}_{n \times K}$$

Model for onions yield (continua)

the model is then

$$\boldsymbol{y} = X\beta + Z\boldsymbol{b} + \varepsilon$$

$$\begin{bmatrix} \boldsymbol{b} \\ \varepsilon \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \sigma_b^2 I_K & 0 \\ 0 & \sigma^2 I_n \end{bmatrix}\right)$$

Estimate the model

```
y=onions$yield
x=(onions$dens-mean(onions$dens))/sd(onions$dens)
X=cbind(1,x)
knots=seq(min(x),max(x),length=12)[-c(1,12)]
Z_K<-(abs(outer(x,knots,"-")))^3
OMEGA_all<-(abs(outer(knots,knots,"-")))^3
svd.OMEGA_all<-svd(OMEGA_all)
sqrt.OMEGA_all<-t(svd.OMEGA_all$v %*% (t(svd.OMEGA_all$u)*sqrt
Z<-t(solve(sqrt.OMEGA_all,t(Z_K)))
dati5 <- list(N=length(y),
                         K=length(knots),
                         y=y,X=X,Z=Z)
```

Estimate the model (continua)

```
model {
  for (i in 1:N) {
   y[i] ~ normal(X[i,] * beta[1:2] + Z[i,] * mspl[1:K] +
            X[i,] * suspbeta[ind[i],1:2]' +
            ZB[i,] * susp[ind[i],1:KB]',sigma);
}
for (k in 1:K){
  mspl[k] ~ normal(0,sigmaspl);
}
for (h in 1:2){
   suspbeta[h,1] ~ normal( 0  ,sigmau);
   suspbeta[h,2] ~ normal( 0  ,sigmau);
   for (k in 1:KB){
    susp[h,k] ~ normal(0,sigmasusp);
   }}
for (j in 1:2) {
  beta[j] ~ normal( 0  ,1000);
}
}
```

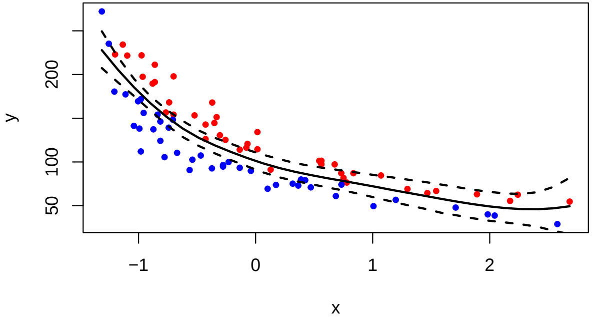
Estimate the model (continua)

```
fit_spline5 <- stan(file = 'spline5.stan', data = dati5,
                    cores=parallel::detectCores())
posterior=extract(fit_spline5)
```

Estimate the model (continua)

```
xnew=seq(min(x),max(x),length=30)
Xnew=cbind(1,xnew)
Z_K<-(abs(outer(xnew,knots,"-")))^3
OMEGA_all<-(abs(outer(knots,knots,"-")))^3
svd.OMEGA_all<-svd(OMEGA_all)
sqrt.OMEGA_all<-t(svd.OMEGA_all$v %*%
                    (t(svd.OMEGA_all$u)*sqrt(svd.OMEGA_all$d)))
Znew<-t(solve(sqrt.OMEGA_all,t(Z_K)))
f=Xnew%*% t(posterior$beta) +  Znew %*% t(posterior$mspl)
plot(x,y,pch=c(20,20)[ind],col=c("red","blue")[ind])
matlines(xnew,t(apply(f,1,quantile,probs=c(0.025,0.5,0.975))),
         col="black",lwd=2,lty=c(2,1,2))
```

Estimate the model (continua)



Alternatives

- Can we model two curves with the same shape but different heights?

Indice

Interaction model

- We want now to specify the model

$$y_{ji} = f(x_{ji}) + g_j(x_{ji}) + \varepsilon_{ji}$$

- The function f is described above, we need to add the functions g_j.
- Note that we may specify a model with subject specific curves without the overall curve f, this would be nothing but a special case of the model above.

Interaction model (continua)

Let $(\nu_1, \ldots, \nu_B)$ be the nodes for the functions g_j and define

$$X_j = \begin{bmatrix} 1 & \ldots & 1 \\ x_{j1} & \ldots & x_{j2} \end{bmatrix}^T$$

$$Z_j = \begin{bmatrix} |x_{j1} - \nu_1|^3 & \ldots & |x_{j1} - \nu_B|^3 \\ \ldots & \ldots & \ldots \\ |x_{jn_j} - \nu_1|^3 & \ldots & |x_{jn_j} - \nu_B|^3 \end{bmatrix} \in \mathcal{M}_{n_j \times B}$$

let then

$$Z' = \begin{bmatrix} Z & X_1 & 0 & Z_1 & 0 \\ & 0 & X_2 & 0 & Z_2 \end{bmatrix} \in \mathcal{M}_{n \times (K+4+2B)}$$

Interaction model (continua)

then the model is

$$\boldsymbol{y} = X\beta + Z'\boldsymbol{u} + \varepsilon$$

where $\boldsymbol{u}$ is multivariate gaussian and

$$\operatorname{cov}(\boldsymbol{u}) = \begin{bmatrix} \sigma_b^2 I & 0 & 0 & 0 & 0 \\ 0 & \Sigma_1 & 0 & 0 & 0 \\ 0 & 0 & \Sigma_2 & 0 & 0 \\ 0 & 0 & 0 & \sigma_b^2 I & 0 \\ 0 & 0 & 0 & 0 & \sigma_b^2 I \end{bmatrix}$$

Estimation

```
y=onions$yield
x=(onions$dens-mean(onions$dens))/sd(onions$dens)
ind=onions$location+1
X=cbind(1,x)
knots=seq(min(x),max(x),length=12)[-c(1,12)]
Z_K<-(abs(outer(x,knots,"-")))^3
OMEGA_all<-(abs(outer(knots,knots,"-")))^3
svd.OMEGA_all<-svd(OMEGA_all)
sqrt.OMEGA_all<-t(svd.OMEGA_all$v %*% (t(svd.OMEGA_all$u)*sqrt
Z<-t(solve(sqrt.OMEGA_all,t(Z_K)))
```

Estimation (continua)

There is actually no need to create the big matrix Z', we will create a matrix like this

$$Z_B = \begin{bmatrix} |x_{11} - \nu_1|^3 & \dots & |x_{11} - \nu_B|^3 \\ \dots & \dots & \dots \\ |x_{1n_1} - \nu_1|^3 & \dots & |x_{1n_1} - \nu_B|^3 \\ |x_{21} - \nu_1|^3 & \dots & |x_{21} - \nu_B|^3 \\ \dots & \dots & \dots \\ |x_{2n_2} - \nu_1|^3 & \dots & |x_{2n_2} - \nu_B|^3 \end{bmatrix} \in \mathcal{M}_{n\times B}$$

Estimation (continua)

```
knotsB=seq(min(x),max(x),length=7)[-c(1,7)]
Z_K<-(abs(outer(x,knotsB,"-")))^3
OMEGA_all<-(abs(outer(knotsB,knotsB,"-")))^3
svd.OMEGA_all<-svd(OMEGA_all)
sqrt.OMEGA_all<-t(svd.OMEGA_all$v %*%
                    (t(svd.OMEGA_all$u)*sqrt(svd.OMEGA_all$d)))
ZB<-t(solve(sqrt.OMEGA_all,t(Z_K)))
```

```
dati6 <- list(N=length(y),K=length(knots),
                         KB=length(knotsB),y=y,X=X,
                         Z=Z,ZB=ZB,ind=ind)
```

Estimation (continua)

```
model {
  for (i in 1:N) {
   y[i] ~ normal(X[i,] * beta[1:2] + Z[i,] * mspl[1:K] +
            X[i,] * suspbeta[ind[i],1:2]' +
            ZB[i,] * susp[ind[i],1:KB]',sigma);
}
for (k in 1:K){
  mspl[k] ~ normal(0,sigmaspl);
}
for (h in 1:2){
   suspbeta[h,1] ~ normal( 0  ,sigmau);
   suspbeta[h,2] ~ normal( 0  ,sigmau);
   for (k in 1:KB){
    susp[h,k] ~ normal(0,sigmasusp);
   }}
for (j in 1:2) {
  beta[j] ~ normal( 0  ,1000);
}
}
```

Estimation (continua)

```
fit_spline6 <- stan(file = 'spline6.stan', data = dati6,
                    cores=parallel::detectCores())
posterior=extract(fit_spline6)
```

As usual, let's have a look at the names of the variables

```
str(posterior)
```

Estimation (continua)

The g_j functions are given by

$$g_1(x) = u_{K+1} + u_{K+2}x + \sum_{h=1}^{B} u_{(K+4)+h}|x - \nu_h|^3$$

$$g_2(x) = u_{K+3} + u_{K+4}x + \sum_{h=1}^{B} u_{(K+4+B)+h}|x - \nu_h|^3$$

The corresponding chains are, for g_1

```
suspA1=posterior$suspbeta[,1,]
susp1=posterior$susp[,1,]
```

and for g_2

Estimation (continua)

```
suspA2=posterior$suspbeta[,2,]
susp2=posterior$susp[,2,]
```

Estimation (continua)

In order to compute the function in

```
xnew=seq(min(x),max(x),length=30)
beta=posterior$beta
```

we define

Estimation (continua)

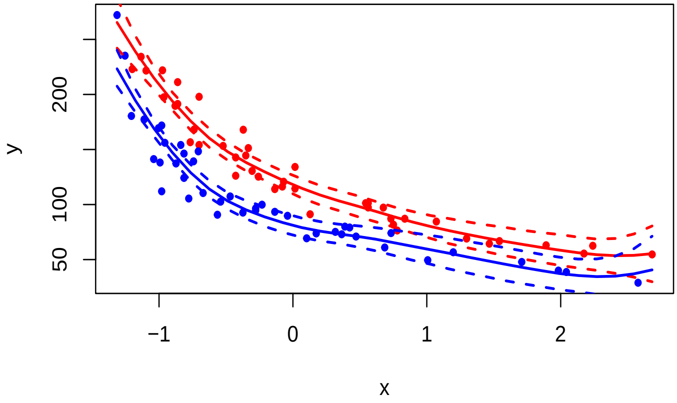
```
Xnew=cbind(1,xnew)
Z_K<-(abs(outer(xnew,knots,"-")))^3
OMEGA_all<-(abs(outer(knots,knots,"-")))^3
svd.OMEGA_all<-svd(OMEGA_all)
sqrt.OMEGA_all<-t(svd.OMEGA_all$v %*% (t(svd.OMEGA_all$u)*sqrt
Znew<-t(solve(sqrt.OMEGA_all,t(Z_K)))

Z_K<-(abs(outer(xnew,knotsB,"-")))^3
OMEGA_all<-(abs(outer(knotsB,knotsB,"-")))^3
svd.OMEGA_all<-svd(OMEGA_all)
sqrt.OMEGA_all<-t(svd.OMEGA_all$v %*% (t(svd.OMEGA_all$u)*sqrt
ZBnew<-t(solve(sqrt.OMEGA_all,t(Z_K)))
```

Estimation (continua)

then

```
spl=posterior$mspl
f1=Xnew%*% t(beta) + Xnew%*%t(suspA1[,1:2])+
    Znew %*% t(spl) + ZBnew %*% t(susp1)
f2=Xnew%*% t(beta) + Xnew%*%t(suspA2[,1:2])+
    Znew %*% t(spl) + ZBnew %*% t(susp2)
plot(x,y,pch=c(20,20)[ind],col=c("red","blue")[ind])
matlines(xnew,t(apply(f1,1,quantile,
                      probs=c(0.025,0.5,0.975))),
         col="red",lwd=2,lty=c(2,1,2))
matlines(xnew,t(apply(f2,1,quantile,
                      probs=c(0.025,0.5,0.975))),
         col="blue",lwd=2,lty=c(2,1,2))
```

Estimation (continua)



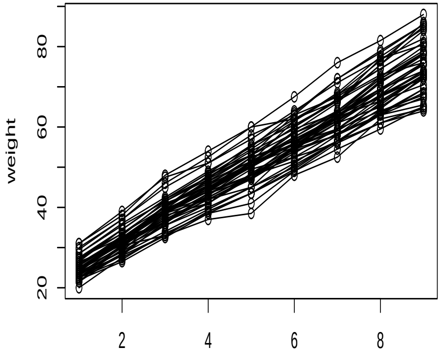
Model for longitudinal data

- One of the advantages of using mixed model representation for splines lies in the fact that we can easily extend it to longitudinal data.
- In particular, we can use what we learnt from the previous exampleto estimate subject specific curves for a number of subjects.

Model for longitudinal data: pigs example

We observe, for 48 pigs their weight for nine successive weeks.

We denote as y_{ij} the weight of the i-th pig in the j-th week.



Bibliography

- Crainiceanu, Ruppert, Wand (2005). Bayesian analysis for penalized spline regression using WinBUGS. Journal of Statistical Software 14(14)
- Eilers and Marx (1996) Flexible Smoothing with B-splines and Penalties. Statistical Science 11(2), 89–121
- Eilers and Marx (2006) Splines, knots and penalties. Computational Statistics 2(6), 637–653
- French, Kammann, Wand (2001) Comment on: Semiparametric Nonlinear Mixed-Effects Models and Their Applications. JASA 96(456), 1285–1288
- Gelman (2006) Prior distributions for variance parameters in hierarchical models. Bayesian Analysis 1(3), 515–533

Bibliography (continua)

- Green and Silverman (1994) Nonparametric Regression and Generalized Linear Models: A Roughness Penalty Approach. London: Chapman and Hall
- Hastie, Tibshirani, Friedman (2009) The elements of statistical learning. (http://www-stat.stanford.edu/ tibs/ElemStatLearn/) Springer
- Marley and Wand (2010). Non-Standard semiparametric regression via BRugs. Journal of Statistical Software 37(5)
- Ruppert, Wand, Carroll (2003) Semiparametric Regression. Cambridge
- Wand and Ormerod (2008) On Semiparametric Regression with O'Sullivan Penalised Splines. Australian and New Zealand Journal of Statistics 50, 179–198

Further issues

- SPR with missing values;
- SPR with measurement error
- robust SPR
- bivariate splines