kokoszka FDA CH4

Noa Jeong

4. Scalar-on-function regression

Scalar-on-function regression :

$$Y_i = \int \beta(s) X_i(s) ds + \epsilon_i$$

Y_i : response are scalars $X_i(s)$: regressors are curves

perfect fit -> erratic, noisy

4.1 Examples

`gasoline` in `refund` package

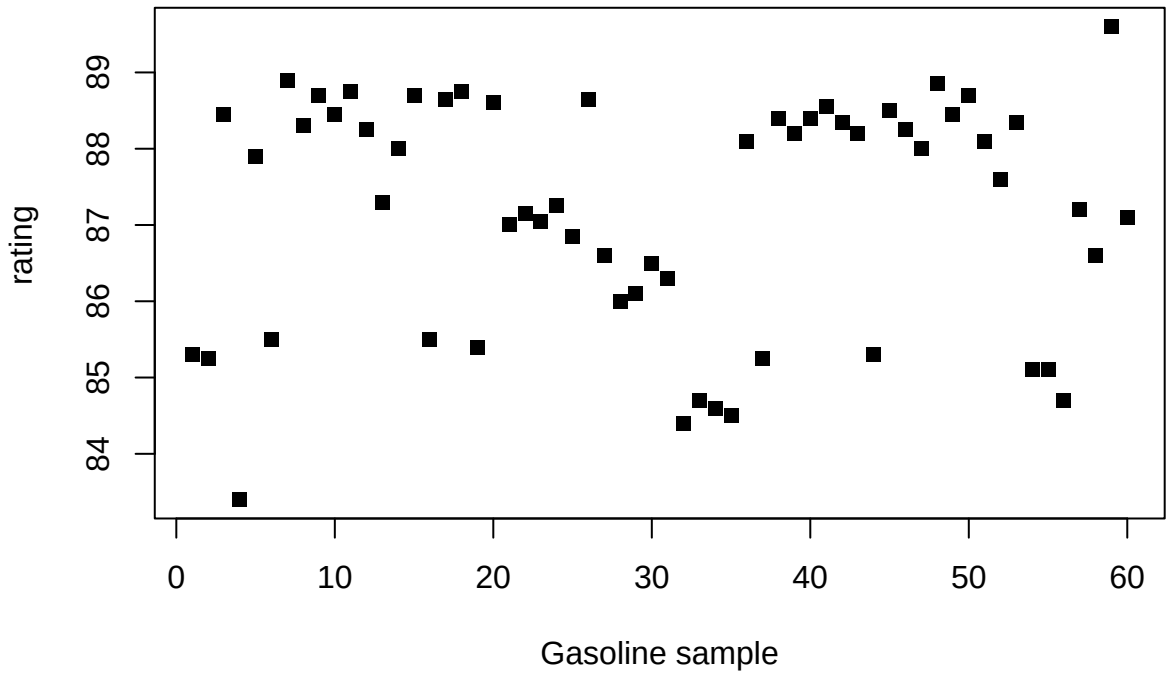
octane rating Y, spectral curve X
e.g. $g : L^2 \to R$ such that $Y \approx g(X)$

Problems 4.8

```
rm(list=ls())
library(fda)
library(refund); library(ggplot2)
#install.packages('dplyr')
#install.packages('reshape2')
library(dplyr); library(reshape2)
set.seed(9000)
```

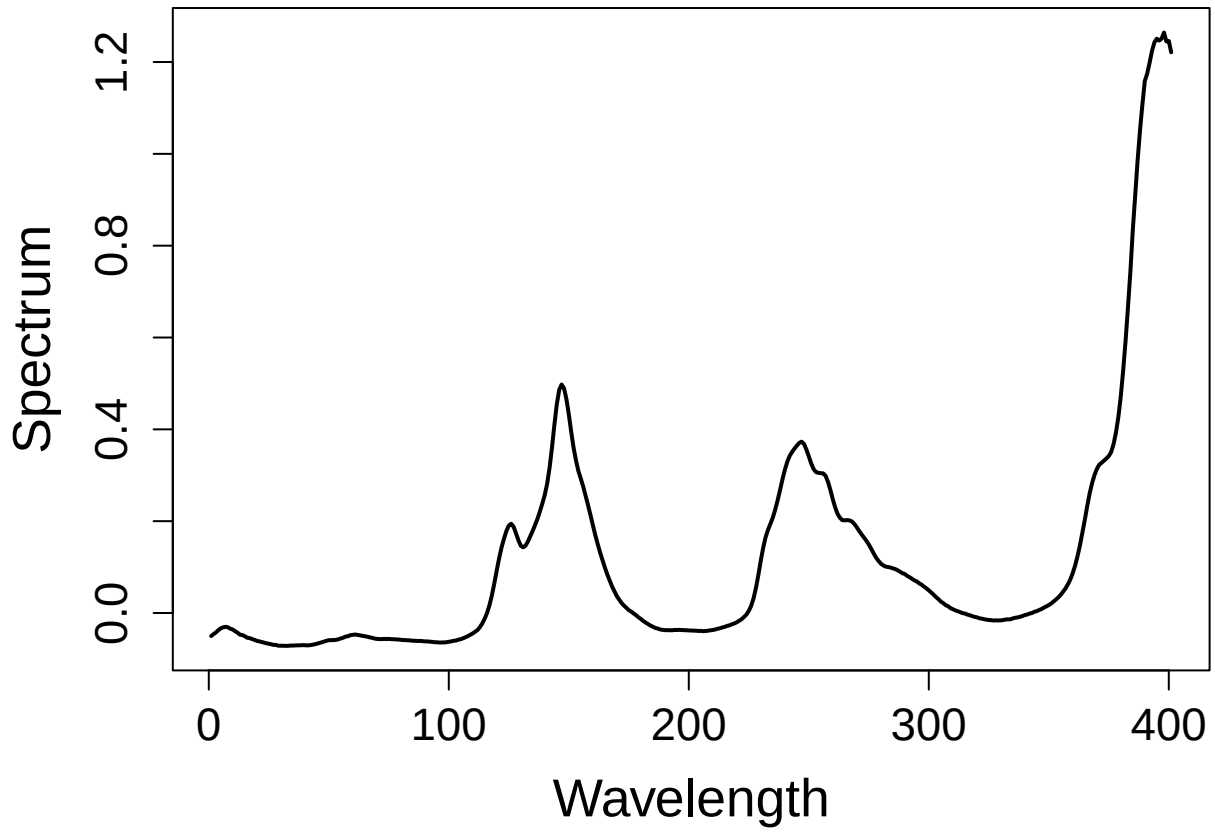
- Octane rating of 60 gasoline samples

```
plot(gasoline$octane, xlab="Gasoline sample",ylab="Octane
    rating", pch=15)
```



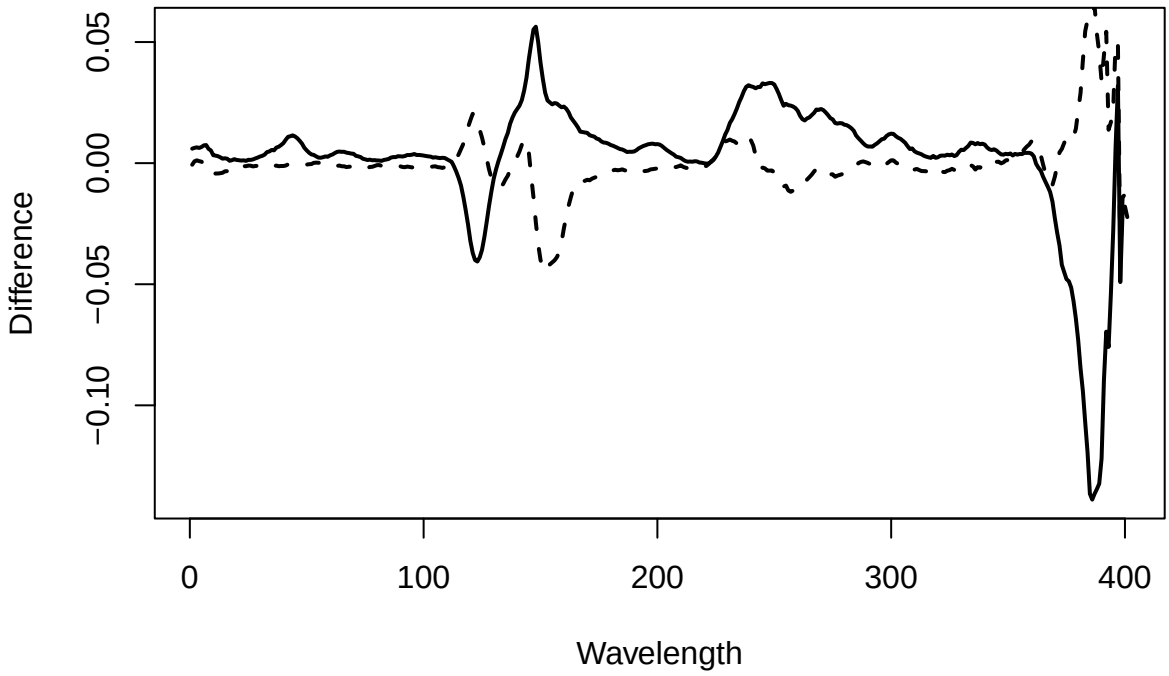
- Near infrared spectrum of gasoline sample with index 1

```
par(ps = 12, cex = 1, cex.lab=1.7, cex.axis=1.4, cex.main=1.7,
     cex.sub=1,mar=c(4.25,4.5,1,1))
plot.ts(gasoline$NIR[1,], lw=2, xlab="Wavelength", ylab="
Spectrum")
```



- differences between the spectrum of the samples with indexes 2 and 1(continuous) and 5 and 1 (dashed)

```
plot.ts(gasoline$NIR[2,]- gasoline$NIR[1,], lw=2, lty=1, xlab=
    "Wavelength", ylab="Difference")
lines(gasoline$NIR[5,]- gasoline$NIR[1,], lw=2, lty=2, xlab="
    Wavelength", ylab="Difference")
```



Problems 4.9

$N = 215$ meat samples.

`tecator` in the R package `fda.usc`

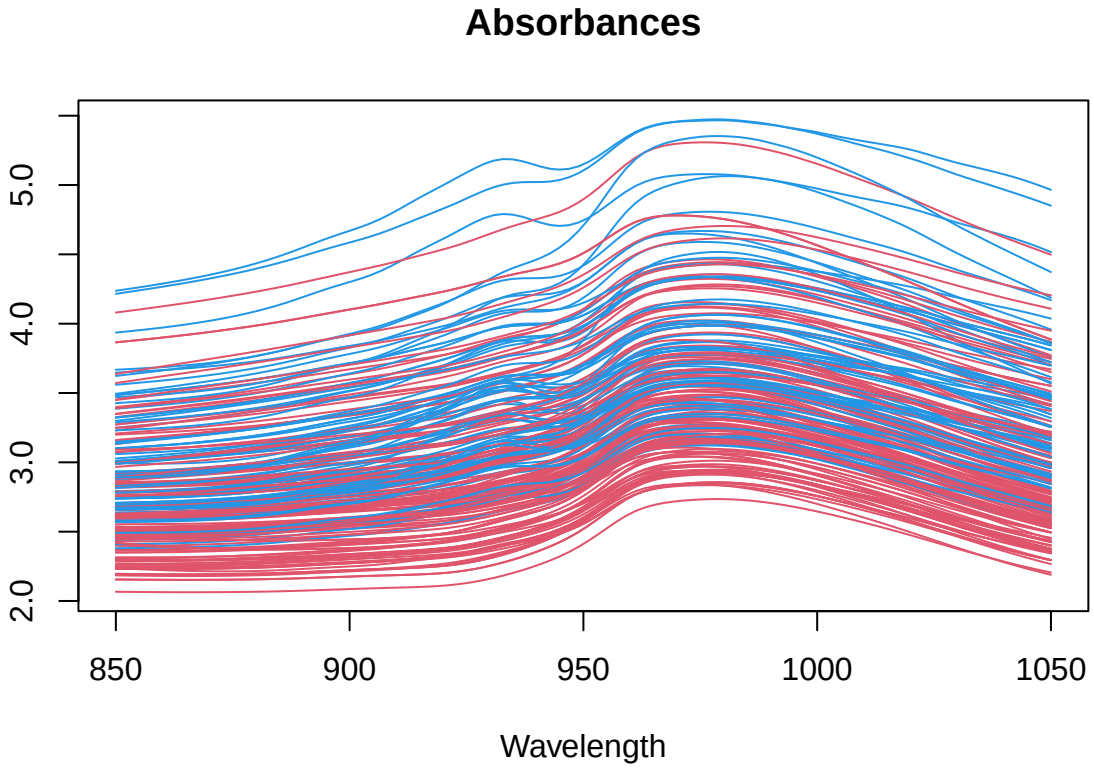
3 scalar responses for each meat sample : fat, water, protein content in percent

```
#install.packages('fda.usc')
library(fda.usc);  data("tecator");  names(tecator)
```

```
## [1] "absorp.fdata" "y"
```

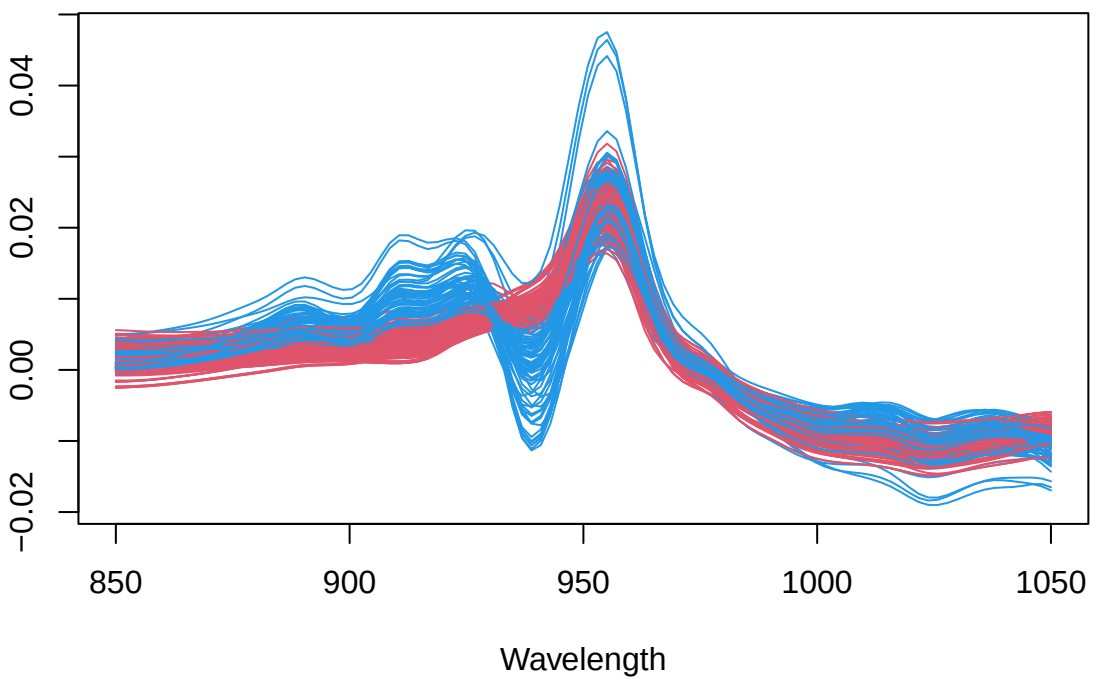
- Absorbance curves of 100 meat samples

```
absorp <- tecator$absorp.fdata
Fat20 <- ifelse(tecator$y$Fat < 20, 0, 1) * 2 + 2 # fat content of less than 20% are red
plot(tecator$absorp.fdata,
     col = Fat20,
     ylab=" ", xlab="Wavelength", main="Absorbances")
```



- Derivatives of these curves

```
absorp.d1 <- fdata.deriv(absorp, nderiv = 1)
plot(absorp.d1, col = Fat20, ylab=" ",
xlab="Wavelength", main="Derivatives")
```



4.2 Standard regression theory review

standard linear model

$$Y = X\beta + \epsilon$$

L_x : subspace of $\mathbb{R}^N$ spanned by columns of X

$X\hat{\beta}$ is projection $\hat{\theta}$ of Y onto L_x . -> $\hat{\theta}$ is the unique vector minimizing the length of $Y - \theta$ over $\theta \in L_x$

$\hat{\beta}$ can uniquely determine only if the **columns of X are linearly independent**

$Y - \hat{\theta}$ is orthogonal to L_x <=> $X^T(Y - \hat{\theta}) = 0$

Thus, least square estimator (unbiased)

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

$$\hat{\beta} = (X^T X)^{-1} X^T (X\beta + \epsilon) = \beta + (X^T X)^{-1} X^T \epsilon$$

To ensure estimator is consistent, $\hat{\beta} \xrightarrow{P} \beta$ as $N \to \infty$ if

$$N^{-1} X^T X \to \Sigma_X \quad \& \quad N^{-1} X^T \epsilon \xrightarrow{P} 0$$

4.3 Difficulties specific to functional regression

functional regression

$$Y = \int \beta(t)X(t)dt + \epsilon$$

cannot compute partial derivatives to find minimum

$$c_x(t,s) = E[X(t)X(s)], \quad c_{XY}(t) = E[X(t)Y]$$

assuming X and ϵ are independent then

$$c_{XY}(t) = \int c_x(t,s)\beta(s)ds$$

(proof in Problem 4.6)

Assuming $E(X(t)) = 0$, kernel $c_x(t,s)$ is the covariance function of random function X, and also

$$c_{XY}(t) = \int c_x(t,s)\beta(s)ds = C_x(\beta)$$

where C_x is the integral operator with kernel $c_x(t,s) = \sum_{j=1}^{\infty} \lambda_j v_j(t) v_j(s)$

approximation

$$Y_i = \int \beta(s)X_i(s)ds + \epsilon_i = \sum_{j=1}^{J} \beta(t_j)X_i(t_j) + \epsilon_i$$

with large number of points t_j

In general, $X(t_j) = [X_1(t_j), X_2(t_j), .., X_N(t_j)]^T$ is strongly correlated (linearly dependent).
colinearity / multicolinearity , so the variance of some components $\hat{\beta}$ become large.

We can use ridge regression and principal component regression.

4.4 Estimation through a basis expansion

with intercept term

$$Y_i = \alpha + \int \beta(s) X_i(s) ds + \epsilon_i$$

basis expansion

$$\beta(t) = \sum_{k=1}^{K} c_k B_k(t)$$

basis function B_k influence the shape of the estimate.
K is smaller than number of points t_j

disadvantage is that the estimate $\hat{\beta}(t)$ depends on the shape of the basis function and number K.

$$\int \beta(s) X_i(t) dt = \sum_{k=1}^{K} c_k \int B_k(t) X_i(t) dt =: \sum_{k=1}^{K} x_{ik} c_k$$

then $\hat{c} = (X^T X)^{-1} X^T Y$

- residuals

$$\epsilon_i = y_i - \hat{\alpha} - \int \hat{\beta}(t) X_i(t) dt = y_i - \hat{\alpha} - \sum_{k=1}^{K} x_{ik} \hat{c}_k$$

The variance of $\hat{c}_k$ is estimated by diagonal entry of matrix $\hat{\sigma}_k^2 = \hat{\sigma}_\epsilon^2 (X^T X)^{-1}$

- approximate 95 % CI of β

$$\sum_{k=1}^{K} = \hat{c}_k B_k(t) + -1.96 \sum_{k=1}^{K} \hat{\sigma}_k B_k(t)$$

β can be expanded without any approx error, i.e. $\beta(t) = \sum_{k=1}^{\infty} c_k B_k(t)$

$$\beta(t) = \sum_{k=1}^{K} c_k B_k(t) + \sum_{k=K+1}^{\infty} c_k B_k(t) = \sum_{k=1}^{K} c_k B_k(t) + \delta(t)$$

$\delta(t)$: truncation error

- model

$$\int \beta(s) X_i(t) dt = \sum_{k=1}^{K} x_{ik} c_k + \int \delta(t) X_i(t) dt := \sum_{k=1}^{K} x_{ik} c_k + \delta_i$$

$$\mathbf{Y} = \mathbf{X}\mathbf{c} + \delta + \epsilon$$

- least square estimator (biased)

$$\hat{\mathbf{c}} = \mathbf{c} + (\mathbf{X^T X})^{-1} \mathbf{X^T} \delta + (\mathbf{X^T X})^{-1} \mathbf{X^T} \epsilon$$

bias.. so necessary to assume that K increases to infinity with the sample size N

- proof

$$N^{-1}(X^T\delta)(k) = N^{-1}\sum_{i=1}^{N} x_{ik}\delta_i$$
$$= \sum_{K+1}^{\infty} c_j \int\int B_k(t)\hat{c}_x(t,x)B_j(s)dtds$$

non-zero number, to make limit vanish assume $K = K(N) \to \infty$

4.5 Estimation with a roughness penalty

K : tuning parameter, adjust the smoothness of resulting estimator $\hat{\beta}(t)$

$$P_\lambda(\alpha,\beta)=\sum_{i=1}^{N}\{Y_i-\alpha\int\beta(t)X_i(t)dt\}^2+\lambda\int[L\beta(t)]^2dt$$

common choice of differential operator : $(L\beta)(t)=\beta''(t)$

if λ is too large β is too smooth.
if λ is too small β reflect random errors

$$\begin{aligned}P_\lambda(\alpha,\beta)&=\sum_{i=1}^{N}\{Y_i-\alpha-\sum_{k=1}^{K}x_{ik}c_k\}^2+\lambda\int[\sum_{k=1}^{K}c_k(LB_k)(t)]^2dt\\&=\sum_{i=1}^{N}\{Y_i-\alpha-\sum_{k=1}^{K}x_{ik}c_k\}^2+\lambda\sum_{k,k'=1}^{K}c_kc_{k'}R_{kk'}\end{aligned}$$

where $R_{kk'}=\int(LB_k)(t)(LB_{k'})(t)dt$

$$\hat{c}=(X^TX+\lambda R)^{-1}X^TY$$

For fixed λ , compute estimates $\alpha_\lambda^{(-i)}$ and $\beta_\lambda^{(-i)}$ using the sample without (Y_i,X_i)

- estimates

$$\hat{Y}_\lambda^{(-i)}=\alpha_\lambda^{(-i)}+\int\hat{\beta}_\lambda^{(-i)}(s)X_i(s)ds$$

- CV method, selects λ which minimizes

$$S_N(\lambda)=\frac{1}{N}\sum_{i=1}^{N}\{Y_i-\hat{Y}_\lambda^{(-i)}\}^2$$

4.6 Regression on functional principal components

function X in L^2

$$X(t) = \mu(t) + \sum_{j=1}^{\infty} \xi_j v_j(t)$$

$$\begin{aligned} Y_i &= \alpha + \int \beta(t)(\hat{\mu}(t) + \sum_{j=1}^{p} \hat{\xi}_{ij}\hat{v}_j(t))dt + \epsilon_i \\ &= \beta_0 + \sum_{j=1}^{p} \hat{\xi}_{ij}\beta_j + \epsilon_i \end{aligned}$$

$$\hat{\beta}(t) = \sum_{j=1}^{p} \hat{\beta}_j \hat{v}_j(t), \quad \hat{\alpha} = \hat{\beta}_0 - \sum_{j=1}^{p} \hat{\beta}_j \int \hat{v}_j(t)\hat{\mu}(t)dt$$

$$Y = \Xi\beta + \varepsilon$$

selection of number of EFPC **p**

4.7 Implementation in `refund` package

```
rm(list=ls())
library(fda)
library(refund); library(ggplot2)
#install.packages('dplyr')
#install.packages('reshape2')
library(dplyr); library(reshape2)
set.seed(9000)
```

$$Y_i = \int \beta(s)X_i(s)ds + \epsilon_i \quad i = 1, 2, .., N$$

- 2 regression functions

$$\beta_1(t) = sin(2\pi t), \quad \beta_2(t) = -f_1(t) + 3f_2(t) + f_3(t), \quad t \in [0, 1]$$

f_1, f_2, f_3 are normal densities

```
n = 1000
grid = seq(0,1,length=101)
beta1 = sin(grid*2*pi) # beta1
beta2 = -dnorm(grid, mean=.2, sd=.03) +
  3*dnorm(grid,mean=.5,sd=.04) +
  dnorm(grid,mean=.75, sd=.05)
```

- regressor curves are generated. $N = 1000$ independent pairs (X_i, ϵ_i) of curves

$$X(t_j) = Zt_j + U + \eta(t_j) + \epsilon(t_j), \quad Z \sim N(1, 0.2^2), \quad U \sim UNIF[0, 5], \eta(t_j) \sim N(0, 1)$$

$$\epsilon(t) = \sum_{k=1}^{10} \frac{1}{k}\{Z_{1k}sin(2\pi tk) + Z_{2k}cos(2\pi tk)\}$$

```
set.seed(9000)
X <- matrix(0, nrow=n, ncol=length(grid))
for(i2 in 1:n){
  X[i2,] <- X[i2,]+rnorm(length(grid),0,1) # Z
  X[i2,] <- X[i2,]+runif(1,0,5) # U
  X[i2,] <- X[i2,]+rnorm(1,1,0.2)*grid # eta
  # epsilon
  for(j2 in 1:10){
    e=rnorm(2, 0, 1/j2^ (2))
    X[i2,] <- X[i2,]+e[1]*sin((2*pi)*grid*j2)
    X[i2,] <- X[i2,]+e[2]*cos((2*pi)*grid*j2)
  }
}
```

- generate artificial data

```
Y <- X %*% beta1 * .01 + rnorm(n, 0, .4)

# FPCR
fit.fpcr = pfr(Y~fpc(X)) # fpc() : construct FPC regression term

# Basis : lf() Construct a FLM regression term
```

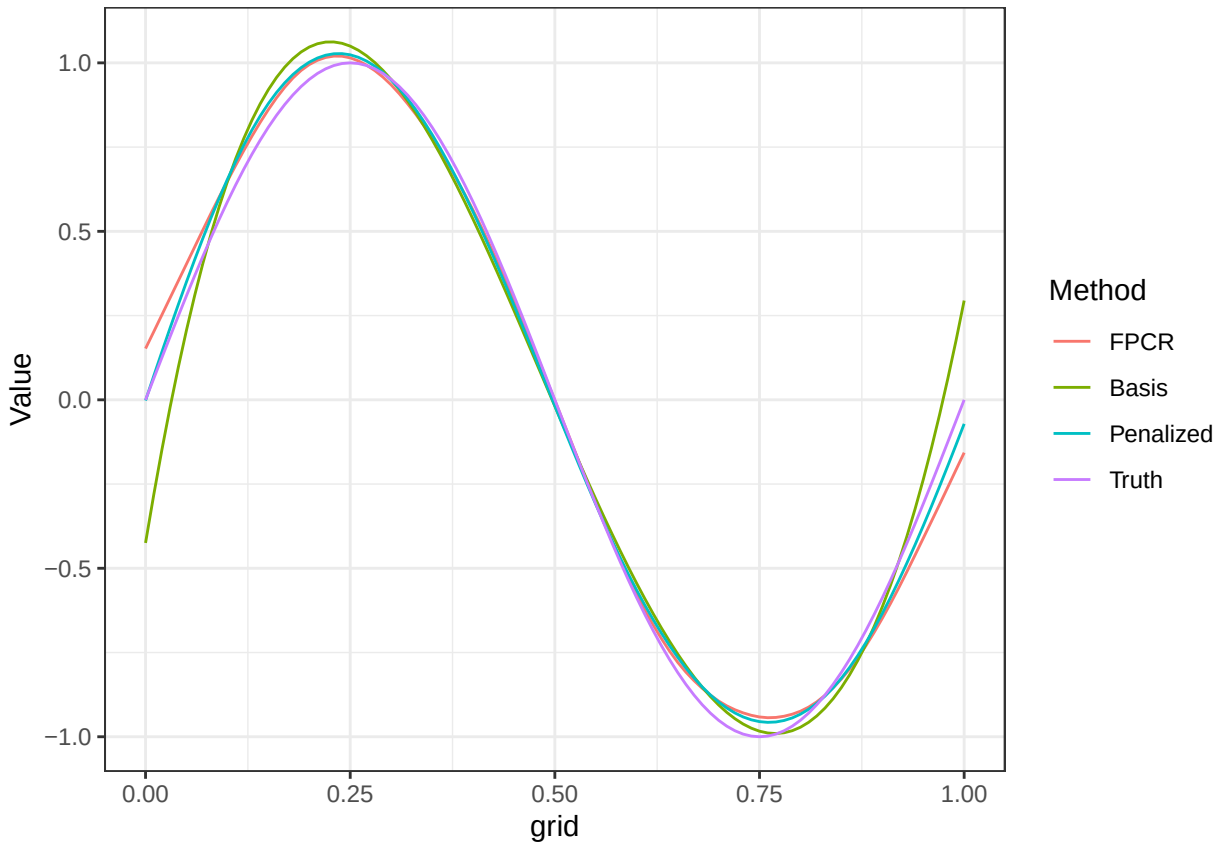
```
fit.lin <- pfr(Y~lf(X,bs='ps',
                    k=5,
                    fx=TRUE)) # no penalty
# Penalized
fit.pfr <- pfr(Y~lf(X,bs='ps',
                    k=10))
```

- plotting

```
coefs <- data.frame(grid=grid,
                    FPCR=coef(fit.fpcr)$value,
                    Basis=coef(fit.lin)$value,
                    Penalized=coef(fit.pfr)$value,
                    Truth=beta1)

coefs.m <- melt(coefs, id='grid') # melt() : convert an object into a molten dataframe
colnames(coefs.m) = c('grid','Method','Value')

ggplot(coefs.m,
       aes(x=grid,y=Value,
           color=Method,group=Method),
       width=12, height=6)+
  geom_path()+
  theme_bw()
```




```
head(coefs)
```

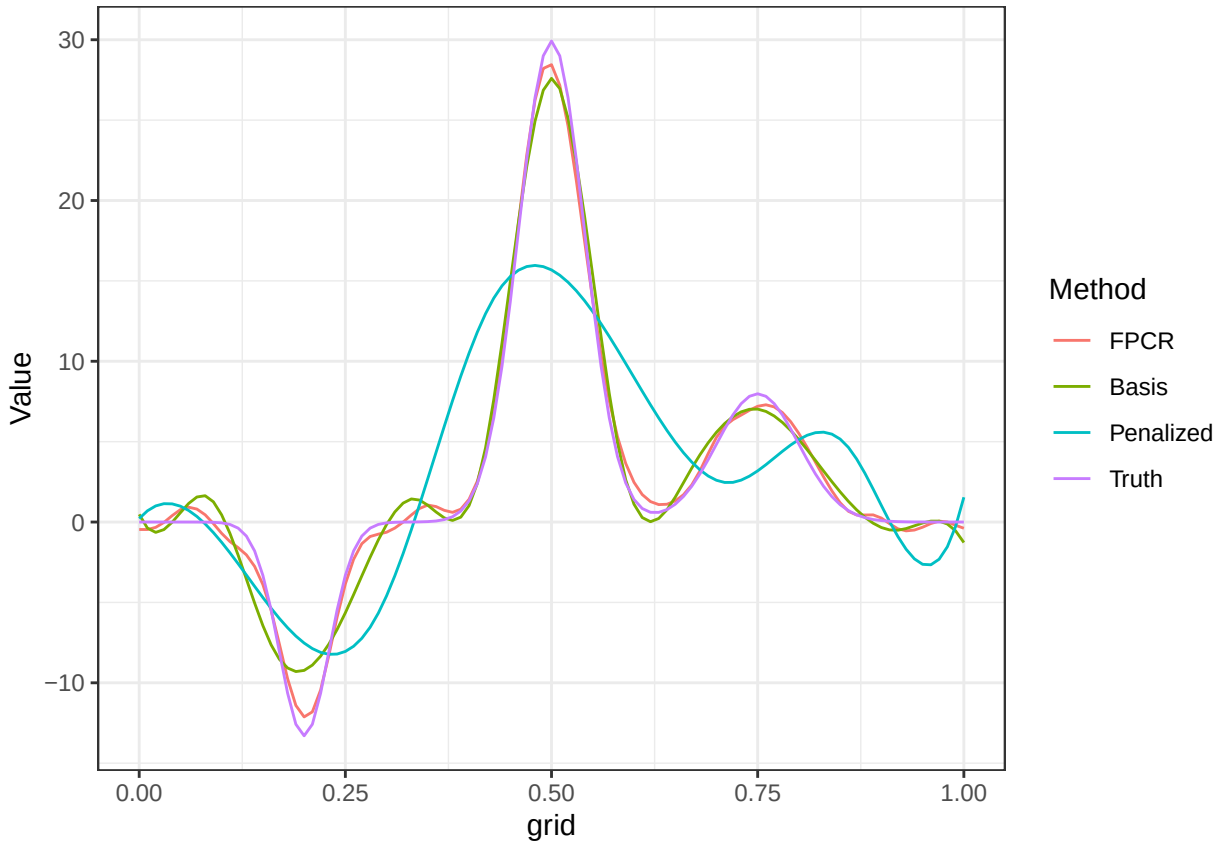
```
##   grid      FPCR       Basis    Penalized      Truth
```

```
## 1 0.00 0.1521190 -0.42477865 -0.002187928 0.00000000
## 2 0.01 0.2019450 -0.28217999  0.070583798 0.06279052
## 3 0.02 0.2522553 -0.14791046  0.142449584 0.12533323
## 4 0.03 0.3026059 -0.02179256  0.213156522 0.18738131
## 5 0.04 0.3528230  0.09635123  0.282451709 0.24868989
## 6 0.05 0.4030920  0.20669842  0.350082237 0.30901699
```

```
X <- matrix(0, nrow=n, ncol=length(grid))
for(i2 in 1:n){
  X[i2,]=X[i2,]+rnorm(length(grid), 0, 1)
  X[i2,]=X[i2,]+runif(1, 0, 5)
  X[i2,]=X[i2,]+rnorm(1, 1, 0.2)*grid
  for(j2 in 1:10){
    e=rnorm(2, 0, 1/j2^(2))
    X[i2,]=X[i2,]+e[1]*sin((2*pi)*grid*j2)
    X[i2,]=X[i2,]+e[2]*cos((2*pi)*grid*j2)
  } }

Y = X %*% beta2 * .01 + rnorm(n, 0, .4)
fit.fpcr = pfr(Y~fpc(X))
fit.lin = pfr(Y~lf(X, bs = "ps", k = 15, fx = TRUE))
fit.pfr = pfr(Y~lf(X, bs = "ps", k = 10))

coefs = data.frame(grid = grid,
                   FPCR = coef(fit.fpcr)$value,
                   Basis = coef(fit.lin)$value,
                   Penalized = coef(fit.pfr)$value,
                   Truth = beta2)
coefs.m = melt(coefs, id = "grid")
colnames(coefs.m) = c("grid", "Method", "Value")
ggplot(coefs.m, aes(x = grid, y = Value, color = Method, group
     = Method),width=12,height=6) + geom_path() + theme_bw()
```


30
20
10
0
−10
Value
0.00
0.25
0.50
0.75
1.00
grid
Method
FPCR
Basis
Penalized
Truth

4.8 Nonlinear scalar-on-function regression

1. functional generalized additive model
2. fully nonparametric representation

$$Y_i = m(X_i) + \varepsilon_i$$

$m : L^2 \to R$

- weighted average

give more weight values Y_i which correspond to regressors close to x

$$\hat{m}(x) = \sum_{i=1}^{N} w_i(x) Y \quad \sum_{i=1}^{N} w_i(x) = 1$$