Lecture 7. Scalar-on-Function Regression

Functional Data Analysis

Jun Song

Department of Statistics
Korea University

Table of contents

Introduction

Functional regression

We again restrict ourselves to $L^2[0,1]$, though, as always, this can be relaxed. Over the next two chapters, we will explore three different regression models.

$$Y \sim X$$

- Scalar-on-function: scalar(or vector) response v.s. functional predictor
- Function-on-scalar: functional response v.s. scalar(or vector) predictor
- Function-on-function: functional response v.s. functional predictor

In this chapter, we focus on the first.

Multiple regression

Let's first give a quick review of multiple regression:

$$y_i = \sum_{j=1}^{p} \beta_j x_{ij} + \varepsilon_i.$$

Typically one assumes that the x_{ij} are deterministic and the ε_i are iid mean zero random variables (either normal or with a second moment), with variance σ^2. It is convenient to write the model using vector notation:

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}.$$

Estimation

Estimating $\boldsymbol{\beta}$ is typically done with least squares:

$$S(\boldsymbol{\beta}) = \sum_{i=1}^{N} |y_i - \mathbf{x}_i^\top \boldsymbol{\beta}|^2 = (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})^\top (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}).$$

Multivariate calculus shows that minimizing the above is equivalent to solving the *normal equations*:

$$(\mathbf{X}^\top \mathbf{X})\widehat{\boldsymbol{\beta}} = \mathbf{X}^\top \mathbf{Y}.$$

As long as $\mathbf{X}$ has full rank then the solution is unique and given by

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}.$$

One can show that the LS estimator is actually consistent and asymptotically normal under a wide array of assumptions.

F-Tests

One can test the hypothesis $H_0 : \beta_{m+1} = \cdots = \beta_p = 0$ for some $m < p$ using F-tests. In particular one fits two models, the full model and the reduced one (using first m predictors), and compares the fit:

$$R_p = \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{p} x_{ij} \hat{\beta}_j \right)^2 \qquad R_m = \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{m} x_{ij} \hat{\beta}_j \right)^2 ,$$

then compares them in the following way

$$F = \frac{(R_m - R_p)/(p-m)}{R_p/(N-p)}.$$

Under H_0 we have $F \sim F_{p-m,N-p}$.

Scalar-on-function regression

When the predictor is a function, the model looks a bit different. In particular, we assume that

$$Y_i = \int_0^1 \beta(t) X_i(t)\ dt + \varepsilon_i.$$

Here ε_i are iid, but now $X_i(t)$ are viewed as random as well; iid and indeptnent of the errors. One can think about this model as a sort of limiting case of the multiple regression model:

$$\int_0^1 \beta(t) X_i(t)\ dt \approx \sum_{j=1}^{J} \frac{1}{J} X_i(t_j)\beta(t_j).$$

Moment perspective

To help in the estimation of $\beta(t)$ it helps to consider a second perspective on estimation that focuses on moments. In particular, consider the covariance between $X(t)$ and Y:

$$\mathrm{E}[X(t)Y] = \mathrm{E}\left[X(t)\int X(s)\beta(s)\ ds\right] = \int C_X(t,s)\beta(s).$$

So one has a relationship

$$C_{XY} = C_X(\beta).$$

If C_X were invertible, then

$$\beta = C_X^{-1}(C_{XY}).$$

Moment perspective: Operators

Model: Population-level We assume that $E(Y) = 0$, $E(X) = 0$.

$$Y = \langle X, \beta \rangle + \epsilon$$

$\langle X, \beta \rangle$ is the predictor of Y given X. It's loss can be defined as

$$\ell(Y, \langle X, \beta \rangle) = (Y - \langle X, \beta \rangle)^2$$

Its expected value is called the risk of β.

$$R(\beta) = E(Y - \langle X, \beta \rangle)^2$$

$$\hat{\beta} = \text{argmin}_{\beta \in \mathcal{H}} E(Y - \langle X, \beta \rangle)^2 = \text{argmin}_{\beta \in \mathcal{H}} - 2E(Y \langle X, \beta \rangle) + E(\langle X, \beta \rangle^2)$$

Moment perspective: Operators

Goal: Find

$$\hat{\beta} = \text{argmin}_{\beta\in\mathcal{H}} E(Y - \langle X, \beta\rangle)^2 = \text{argmin}_{\beta\in\mathcal{H}} - 2E(Y\langle X, \beta\rangle) + E(\langle X, \beta\rangle^2).$$

Define the covariance operator (remark: they are linear operators)

$$C_{XY} = E[X \otimes Y], \quad C_{YX} = E[Y \otimes X], \quad C_X = E[X \otimes X]$$

$$\hat{\beta} = \text{argmin}_{\beta\in\mathcal{H}} - 2C_{YX}(\beta) + \langle \beta, C_X\beta\rangle.$$

Thus, the estimation equation is

$$C_X\beta = C_{XY} = E(XY)$$

Recap

Recap

Recall that we aim to fit the model

$$Y_i = \int_0^1 \beta(t) X_i(t) \ dt + \varepsilon_i.$$

However, we ran into issues with X_i being infinite-dimensional. We will now go through several options.

Basis Expansion

Our first approach fits in quite naturally with how we constructed functional units. Let $\{B_k(t)\}$ be a basis of $L^2[0,1]$, then we assume that

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

It is important to remember that this is just an approximation. In this case, we have

$$Y_i \approx \int_0^1 \sum_{k=1}^{K} c_k B_k(t) X_i(t)\ dt + \varepsilon_i = \sum_{k=1}^{K} c_k x_{ik} + \varepsilon_i.$$

Where $x_{ik} = \langle X_i, B_k \rangle$.

Basis Expansion

After estimating the c_k we have

$$\hat{\beta}(t) = \sum_{k=1}^{K} \hat{c}_k B_k(t).$$

However, there are two things to keep in mind:

1. The parameter K is a sort of smoothing parameter, but usually must be fairly small.
2. The estimate one gets is only an **approximation** of $\beta(t)$.

In particular, to show $\hat{\beta}(t)$ converges to $\beta(t)$ one must let $K \to \infty$ with the sample size. One can choose K using cross validation, GCV, AIC, or BIC.

Bias-variance trade-off

If we compare $\hat{\beta}(t)$ to $\beta(t)$ then we have

$$\hat{\beta}(t) - \beta(t) = \sum_{k=1}^{K} (\hat{c}_k - c_k) B_k(t) - \sum_{k=K+1}^{\infty} c_k B_k(t).$$

As we increase K

- The second term decreases ("bias" decreases)
- This increases the number of c_k to estimate ("variance" increases)

This sort of trade-off comes up very often in FDA and nonparametric statistics.

Penalized Regression

- Cons of basis expansion approach:
  - K acts a role as a smoothing parameter, but hard to tune
  - The shape of $\hat{\beta}(t)$ depends heavily on $B_k(t)$ (restrict the possible shapes one can estimate)
- An alternative: use the model with a very large K and then use a penalty to control the smoothing.

Penalized regression

Let L be a linear differentiable operator (such as the second derivative), then define

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

Penalized regression

We can again convert this into a multivariate regression problem. First, as before

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

However, now K is taken to be "large" so that the approximation above is nearly exact. Turning to the penalty we have

$$(L\beta)(t) \approx \sum_{k=1}^{K} c_k (LB_k)(t).$$

Penalized regression

The norm of $L\beta$ can then be expressed as

$$\lambda \int [(L\beta)(t)]^2 \, dt = \lambda \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} c_{k_1} c_{k_2} \langle LB_{k_1}, LB_{k_2} \rangle$$
$$= \lambda \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} c_{k_1} c_{k_2} R_{k_1,k_2}.$$

So the penalized regression can be phrased as

$$P_\lambda(\mathbf{c}) = (\mathbf{Y} - \mathbf{X}\mathbf{c})^\top (\mathbf{Y} - \mathbf{X}\mathbf{c}) + \lambda \mathbf{c}^\top \mathbf{R}\mathbf{c}.$$

where

$$\mathbf{c} = \begin{pmatrix} \alpha \\ c_1 \\ \vdots \\ c_K \end{pmatrix} \quad \mathbf{X} = \begin{pmatrix} 1 & x_{11} & \dots & x_{1K} \\ \vdots & & & \\ 1 & x_{N1} & \dots & x_{NK} \end{pmatrix} \quad \mathbf{R} = \begin{pmatrix} 0 & 0 & \dots & 0 \\ 0 & R_{11} & \dots & R_{1K} \\ \vdots & & & \\ 0 & R_{K1} & \dots & R_{LK} \end{pmatrix}.$$

Review: Matrix Calculus: $\mathbb{R}^n \to \mathbb{R}$

When $f : \mathbb{R}^n \to \mathbb{R}$,

$$\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \cdots & \frac{\partial f}{\partial x_n} \end{bmatrix}, \qquad \nabla f = \left(\frac{\partial f}{\partial x}\right)^\top \in \mathbb{R}^n,$$

where ∇f is the gradient of f.

Review: Matrix Calculus: $\mathbb{R}^n \to \mathbb{R}^m$

When $f : \mathbb{R}^n \to \mathbb{R}^m$, then $f(x) = (f_1(x), \ldots, f_m(x))^\top$.

$$\frac{\partial f}{\partial x} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix} \in \mathbb{R}^{m \times n}$$

One can use an alternative expression. Let $Df : \mathbb{R}^n \to \mathbb{R}^m$ such that $x_0 \mapsto \frac{\partial f}{\partial x}(x_0)$, which is a linear mapping such that

$$\lim_{x \to x_0} \frac{f(x) - f(x_0) - Df(x_0)(x - x_0)}{\|x - x_0\|} = 0$$

Remark: Some literature use the transpose for the definition of derivatives. See matrix calculus.

Penalized regression

We are minimizing the following loss function with respect to $\mathbf{c} \in \mathbb{R}^K$

$$P_\lambda(\mathbf{c}) = \mathbf{Y}^\top\mathbf{Y} - 2\mathbf{c}^\top\mathbf{X}^\top\mathbf{Y} + \mathbf{c}^\top(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{R})\mathbf{c}.$$

Then the first derivative is

$$\frac{\partial P_\lambda(\mathbf{c})}{\partial \mathbf{c}} = -2\mathbf{Y}^\top\mathbf{X} + 2\mathbf{c}^\top(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{R})$$

The second derivative is

$$\frac{\partial^2 P_\lambda(\mathbf{c})}{\partial \mathbf{c}^2} = 2(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{R}),$$

which is non-negative definite. Thus, the minimizer is the solution to

$$\frac{\partial P_\lambda(\mathbf{c})}{\partial \mathbf{c}} = 0.$$

Penalized regression

This type of regression is commonly called "ridge regression" in multivariate statistics. The penalized least squares estimate is then given by

$$\hat{\mathbf{c}} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{R})^{-1} \mathbf{X}^\top \mathbf{Y}.$$

Again, we have that

$$\hat{\beta}(t) = \sum_{k=1}^{K} \hat{c}_k B_k(t).$$

Penalized regression (choosing λ)

In practice, it can be difficult to find a reasonable range of λ to consider. As with generalized cross validation, one can use the *hat* matrix to compute the effective degrees of freedom. In particular

$$\mathbf{H} = \mathbf{X}(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{R})^{-1}\mathbf{X}^\top \qquad df = N - \mathrm{trace}(\mathbf{H}) \qquad \hat{\mathbf{Y}} = \mathbf{H}\mathbf{Y}.$$

To choose λ, one can use the degrees of freedom above to compute the GCV, AIC, or BIC.

$$GCV = \frac{N|\mathbf{Y} - \hat{\mathbf{Y}}|^2}{(N - \mathrm{trace}(\mathbf{H}))^2}$$

Basis expansion

Scalar-on-function regression

- $Y \in \mathbb{R}$, $X \in \mathcal{H}$. Assume Y and X are associated as

  $$Y = \alpha + \langle \beta, X \rangle + \epsilon,$$

  where β is an unknown function in $\mathcal{H}$.

Coordinate representation

- Let $\mathcal{H}_1$ be a vector space with basis $\mathcal{B}=\{b_1,\ldots,b_n\}$.
- For each $f \in \mathcal{H}_1$, there is a vector $\alpha = (\alpha_1,\ldots,\alpha_n)^\top$ such that $f = \sum_{i=1}^n \alpha_i b_i$.
- $[f]_{\mathcal{B}} = \alpha$ is a coordinate representation of f w.r.t. $\mathcal{B}$
- Let $\mathcal{H}_2$ be another Hilbert spaces, spanned by $\mathcal{C} = \{c_1,\ldots,c_m\}$
- If $A : \mathcal{H}_1 \to \mathcal{H}_2$ is a linear operator. Then, it can be shown that, for any $f \in \mathcal{H}_1$,

$$[Af]_{\mathcal{C}} = ({}_{\mathcal{C}}[A]_{\mathcal{B}})[f]_{\mathcal{B}},$$

where

$${}_{\mathcal{C}}[A]_{\mathcal{B}} = ([Ab_1]_{\mathcal{C}},\ldots,[Ab_n]_{\mathcal{C}})$$

Coordinate representation

- Let $\mathcal{H}_1$ be a vector space with basis $\mathcal{B}=\{b_1, \ldots, b_n\}$.
- Let $\mathcal{H}_2$ be another Hilbert spaces, spanned by $\mathcal{C} = \{c_1, \ldots, c_m\}$
- Let $\mathcal{H}_3$ be another Hilbert spaces, spanned by $\mathcal{D} = \{d_1, \ldots, d_\ell\}$
- If $A : \mathcal{H}_1 \to \mathcal{H}_2$, $B : \mathcal{H}_2 \to \mathcal{H}_3$ are linear operators. Then, it can be shown that, for any $f \in \mathcal{H}_1$,

$$[BAf]_{\mathcal{C}} = ({}_{\mathcal{D}}[B]_{\mathcal{C}})({}_{\mathcal{C}}[A]_{\mathcal{B}})[f]_{\mathcal{B}},$$

- Subscripts omitted if it is trivial.

Coordinate Representation Toolkit

Let G be a gram matrix for $\mathcal{H}_1$ w.r.t. $\mathcal{B}$. i.e., $G \in \mathbb{R}^{n \times n}$ such that $(G)_{ij} = \langle b_i, b_j \rangle_{\mathcal{H}_1}$.
Suppose that $x_1, \ldots, x_N \in \mathcal{H}_1$

1. $\langle x_i, x_j \rangle = [x_i]^\intercal G [x_j]$
2. For $g \in \mathcal{H}_1$, $f \in \mathcal{H}_2$, ${}_{\mathcal{C}}[g \otimes f]_{\mathcal{B}} = [g]_{\mathcal{C}} [f]_{\mathcal{B}}^\intercal G$

   Let $Q_N = I_N - N^{-1} 1_N 1_N^\intercal$ and $[x_{1:N}] \in \mathbb{R}^{n \times N}$ whose columns are $[x_i]$.
   $\hat{C} = \frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x}) \otimes (x_i - \bar{x})$
3. $[\hat{C}] = N^{-1} [x_{1:N}] Q_N [x_{1:N}]^\intercal G$
4. $[\hat{C}^\alpha] = G^{-1/2} (N^{-1} G^{1/2} ([x_{1:N}] Q_N [x_{1:N}] [x_{1:N}]^\intercal G^{1/2})^\alpha G^{1/2}$

FPC-Regression

Scalar-on-function regression

- $Y \in \mathbb{R}$, $X \in \mathcal{H}$. Assume Y and X are associated as

$$Y = \alpha + \langle \beta, X \rangle + \epsilon,$$

where β is an unknown function in $\mathcal{H}$.

Regression on FPCs

- Fixed basis for $\beta(t)$: basis expansion, penalized regression
- A data-driven basis for $X_i(t)$:
  - Use the FPCs from the $X_i(t)$
  - This might not be optimal for representing $\beta(t)$, but it is optimal for the $X_i(t)$
  - This approach can often result in a nice prediction performance.

Regression on FPCs

This approach takes a form which is basically equivalent to the basis expansion approach. However, one has to be a bit careful as the FPCs are computed using centered $X_i(t)$. In particular, we can expand

$$X_i(t) \approx \hat{\mu}(t) + \sum_{j=1}^{p} \hat{\xi}_{ij}\hat{v}_j(t) \qquad \hat{\xi}_{ij} = \langle X_i - \hat{\mu}, v_j \rangle.$$

Therefore

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

Regresson on FPCs

So, once again, we can carry out a multiple regression to estimate $\beta_0, \ldots, \beta_p$:

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{\Xi}^\top \boldsymbol{\Xi})^{-1} \boldsymbol{\Xi} \mathbf{Y} \qquad \boldsymbol{\Xi} = \begin{pmatrix} 1 & \hat{\xi}_{11} & \ldots & \hat{\xi}_{1p} \\ \vdots & & & \\ 1 & \hat{\xi}_{N1} & \ldots & \hat{\xi}_{Np} \end{pmatrix}.$$

We can then estimate the original parameters as

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

Regression on FPCs

Interestingly, using the FPCs of the X_i causes the design to become orthogonal:

$$\frac{1}{N}\sum_{i=1}^{N}\xi_{ij}\xi_{ik} = \frac{1}{N}\sum_{i=1}^{N}\langle X_i - \hat{\mu}, \hat{v}_j\rangle\langle X_i - \hat{\mu}, \hat{v}_k\rangle = \langle \hat{C}(\hat{v}_j), \hat{v}_k\rangle = \hat{\lambda}_j 1_{j=k}.$$

This means that the estimator for the $\hat{\beta}_j$ can be written down fairly exactly (check by yourself)

$$\hat{\beta}_0 = \frac{1}{N}\sum_{i=1}^{N} Y_i \qquad \hat{\beta}_j = \frac{1}{N\hat{\lambda}_j}\sum_{i=1}^{N}\hat{\xi}_{ij}Y_i.$$

Regression on FPCs

This method closely mimics what we discussed last lecture, in particular, we said

$$\beta = C_X^{-1}(C_{XY}).$$

Using the FPCs we are approximating

$$\hat{C}_X^{-1} \approx \sum_{j=1}^{p} \hat{\lambda}_j^{-1} \hat{v}_j \otimes \hat{v}_j.$$

This is sometimes called a *generalized inverse*, where, instead of inverting the small eigenvalues, you just set them to zero.

Comparing the three

Unfortunately, no single approach is better than the others. Each of them may do better depending on the dataset.

Computation

To compute these different estimates we will use the refund package in R. This package is fairly broad including a lot of different functions.

```
library(refund); library(ggplot2)
library(dplyr); library(reshape2)
set.seed(2016)
```

The package ggplot2 is used for fancy looking plots, while dplyr and reshape2 are used for data manipulation.

(Jeff Goldsmith from Columbia Biostat).

Simulation

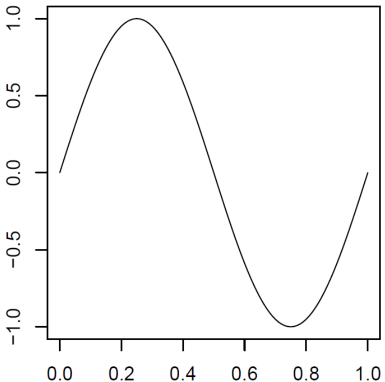
We will simulate data a little differently than before. We will set $\alpha = 0$ and will consider two different slope functions

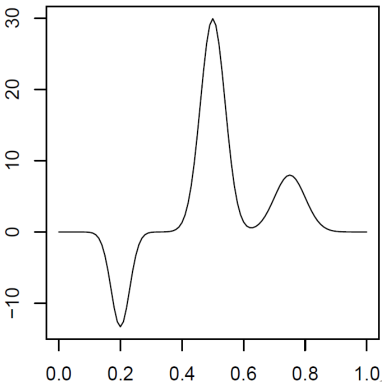
$$\beta(t) = \sin(2\pi t) \quad \text{or} \quad \beta(t) = -f_1(t) + 3f_2(t) + f_3(t),$$

where f_1, f_2, and f_3 are different normal densities means $.2$, $.5$, and $.75$ and standard deviations $.3$, $.4$, and $.5$ respectively.

Simulation

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





Simulation

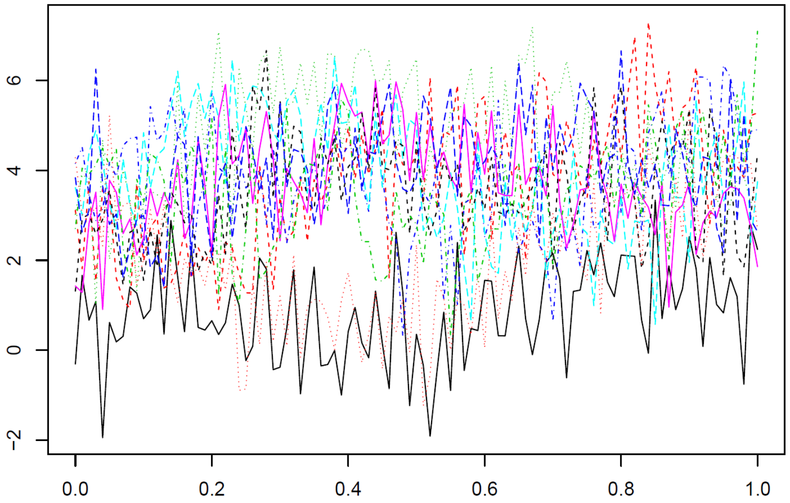
We will generate the predictors as

$$X(t_j) = Zt_j + U + \eta(t_j) + \varepsilon(t_j),\ Z \sim N(1, 0.2^2),$$
$$U \sim \text{Unif}[0,5],\ \eta(t_j) \sim N(0,1).$$

While the last term is given by

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

Simulation



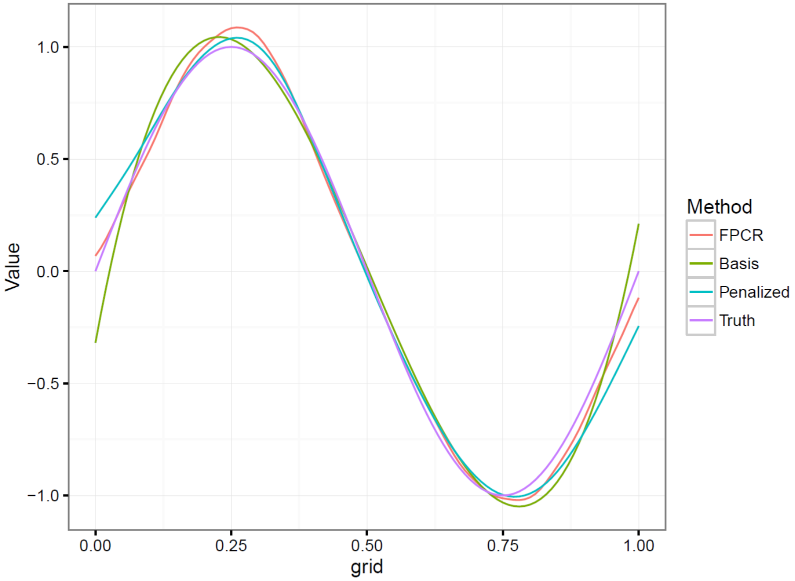
Simulation

```
Y1 = X %*% beta1 * .01 + rnorm(n, 0, .4)
Y2 = X %*% beta2 * .01 + rnorm(n, 0, .4)
fit.fpcr1 = pfr(Y1 ~ fpc(X,pve=.85))
fit.fpcr2 = pfr(Y2 ~ fpc(X,pve = 0.85))
fit.lin1 = pfr(Y1 ~ lf(X, bs = "ps", k=5, fx=TRUE))
fit.lin2 = pfr(Y2 ~ lf(X, bs = "ps", k=20, fx=TRUE))
# "ps" stands for  "penalized splines",
# sp = -1 means no penalty is used
fit.pfr1 = pfr(Y1 ~ lf(X, bs = "ps",k=50))
fit.pfr2 = pfr(Y2 ~ lf(X, bs = "ps",k=50))
# if sp is not specified, data
# driven smoothing is used
```

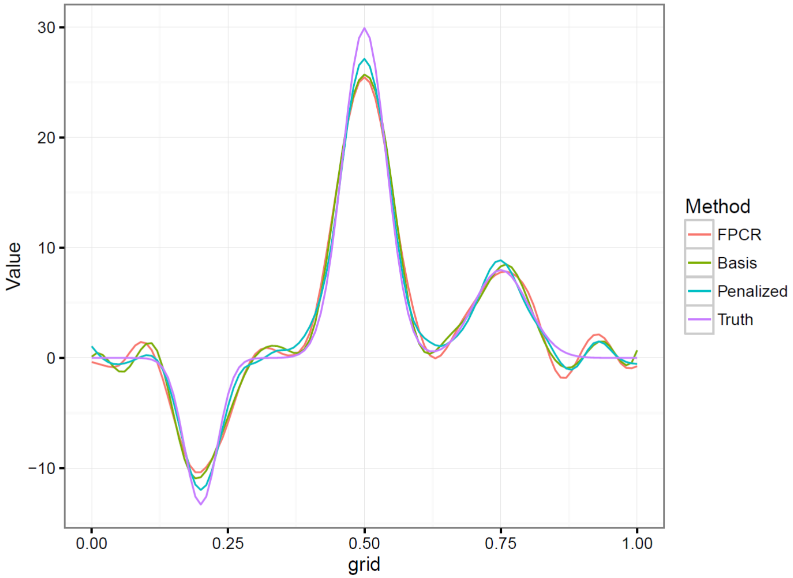
Simulation

```r
par(mar=c(2,2,1,1))
coefs = data.frame(grid = grid,
                   FPCR = coef(fit.fpcr1)$value,
                   Basis = coef(fit.lin1)$value,
                   Penalized = coef(fit.pfr1)$value,
                   Truth = beta1)
# melt stacks the different functions
coefs.m = melt(coefs, id = "grid")
colnames(coefs.m) = c("grid", "Method", "Value")
ggplot(coefs.m, aes(x = grid, y = Value,
  color = Method)) + geom_path() + theme_bw()
```

Simulation


1.0
0.5
0.0
−0.5
−1.0
Value
0.00
0.25
0.50
0.75
1.00
grid
Method
FPCR
Basis
Penalized
Truth

Simulation



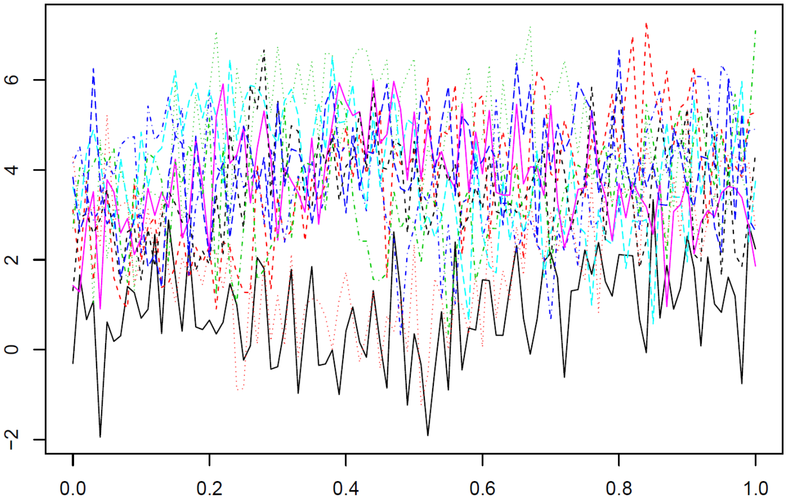
30
20
10
0
−10
Value
0.00
0.25
0.50
0.75
1.00
grid
Method
FPCR
Basis
Penalized
Truth

Outline

- Manual regression
- DTI example
- Nonlinear regression

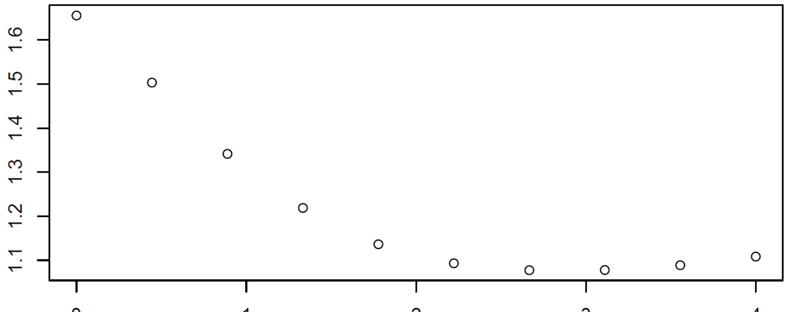
Remind: Simulation

We will manually compute the FPCA method to compare with pfr.

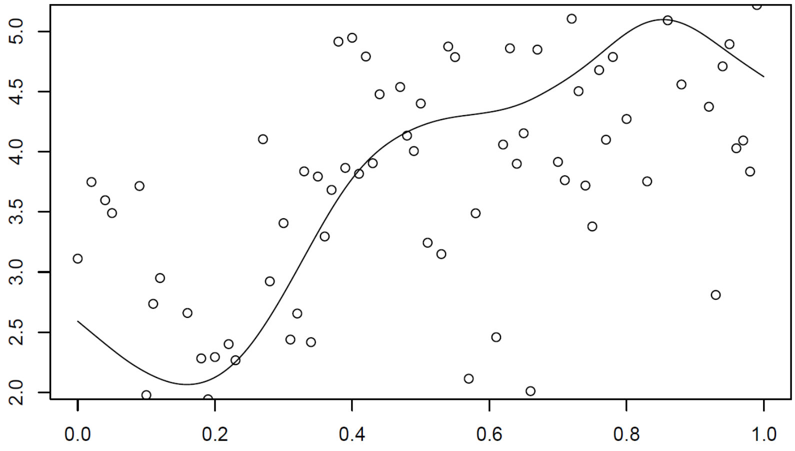


Simulation

```
mybasis<-create.bspline.basis(c(0,1),nbasis=100)
lambda_all<-10^(seq(0,-4,length=10)); gcv<-numeric(0
for(lambda in lambda_all){
  myfdpar<-fdPar(mybasis,2,lambda=lambda)
  X.f<-smooth.basis(grid,t(X),myfdpar)
  gcv<-c(gcv,mean(X.f$gcv))}
plot(-log(lambda_all,base=10),gcv)
```



Simulation



Simulation - FPCA

```
X.pc<-pca.fd(X.f$fd,3)
cumsum(X.pc$varprop)

## [1] 0.6664097 0.8300253 0.9809916

lm_fit1<-lm(Y1~X.pc$scores)
lm_fit2<-lm(Y2~X.pc$scores)
```

Simulation - FPCA

```
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -0.1277     0.0130   -9.80  1.1e-21
## X.pc$scores1    0.0112     0.0092    1.22  2.2e-01
## X.pc$scores2   -0.7478     0.0185  -40.45 2.2e-212
## X.pc$scores3   -0.0098     0.0192   -0.51  6.1e-01
## Analysis of Variance Table
##
## Response: Y1
##               Df Sum Sq Mean Sq F value    Pr(>F)
## X.pc$scores    3 278.40   92.80  545.86 < 2.2e-16
## Residuals    996 169.33    0.17
##
## X.pc$scores ***
## Residuals
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

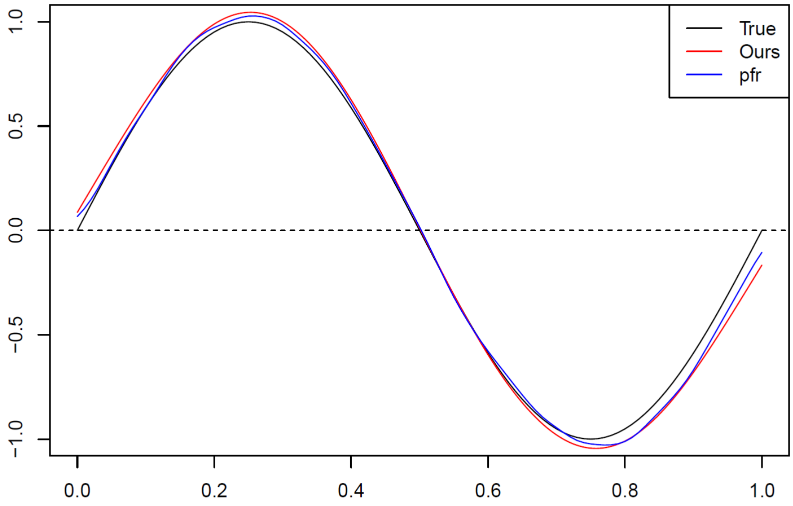
Simulation - FPCA

```
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)          9.5      0.035     274  0.0e+00
## X.pc$scores1         3.0      0.024     124  0.0e+00
## X.pc$scores2         2.8      0.049      57 7.9e-315
## X.pc$scores3        -4.6      0.051     -91  0.0e+00
## Analysis of Variance Table
##
## Response: Y2
##               Df Sum Sq Mean Sq F value    Pr(>F)
## X.pc$scores    3  32111 10703.8  8921.7 < 2.2e-16
## Residuals    996   1195     1.2
##
## X.pc$scores ***
## Residuals
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

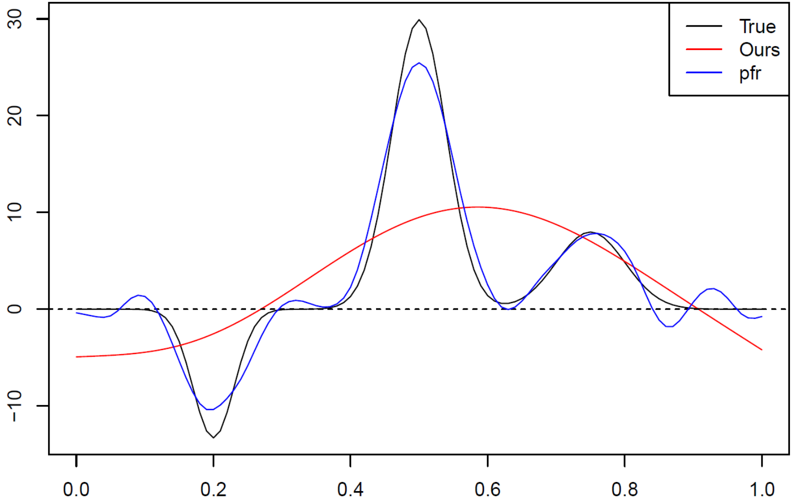
Simulation - FPCA

```
beta_coef1 = coef(X.pc$harmonics)%*%lm_fit1$coef[-1]
beta_hat1<-fd(beta_coef1,mybasis)
beta_coef2 = coef(X.pc$harmonics)%*%lm_fit2$coef[-1]
beta_hat2<-fd(beta_coef2,mybasis)
pfr_fit1<-pfr(Y1~fpc(X,ncomp=3))
pfr_fit2<-pfr(Y2~fpc(X,ncomp=15))
```

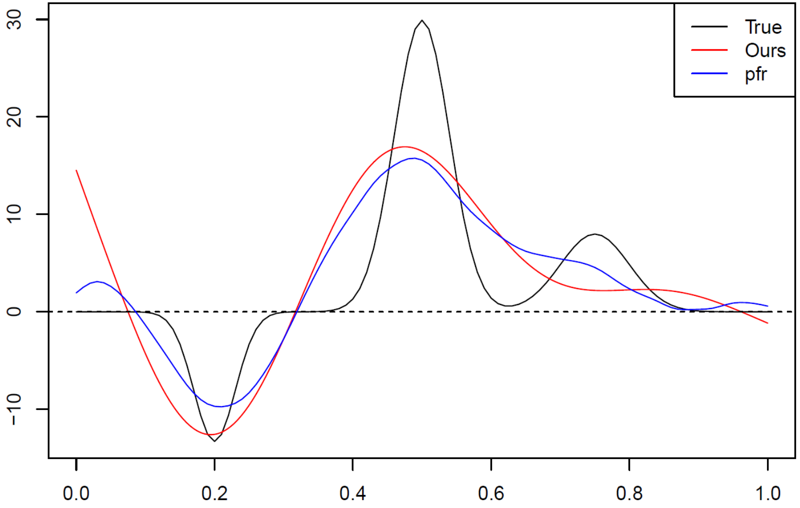
Simulation - FPCA



Simulation - FPCA



Simulation - FPCA - 6pcs



DTI Example

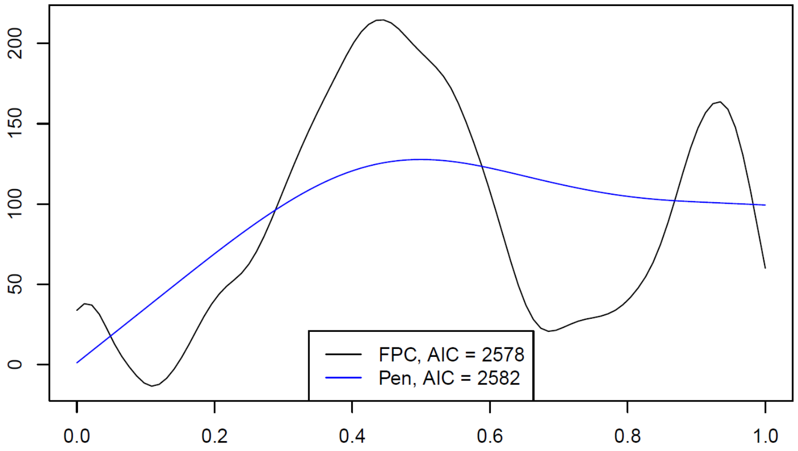
```r
library(refund)
X = DTI$cca[DTI$case==1,]
Y = DTI$pasat[DTI$case==1]
grid = seq(0,1,length=dim(X)[2])
drop<-unique(which(is.na(X),arr.ind=TRUE)[,1])
X = X[-drop,]; Y = Y[-drop]
```

DTI Example

Paced Auditory Serial Addition Test (PASAT) is a diagnostic test

```
k_all = 5:20
aic.fpc = numeric(0)
for(k in k_all){
  pfr.fpc<-pfr(Y~fpc(X,ncomp=k))
  aic.fpc<-c(aic.fpc,pfr.fpc$aic)
}
k = k_all[which.min(aic.fpc)]
pfr.fpc<-pfr(Y~fpc(X,ncomp=k))
pfr.pen<-pfr(Y~lf(X,k=50))
beta_fpc = coef(pfr.fpc)$value
beta_pen = coef(pfr.pen)$value
k

## [1] 7
```

DTI Example



DTI Example

```
rbind(summary(pfr.pen)[[22]],summary(pfr.pen)[[24]])

##              Estimate Std. Error    t value
## (Intercept)  1.245090   6.376514  0.1952619
## s(X.tmat):L.X 2.620015   2.985742 20.5994547
##                  Pr(>|t|)
## (Intercept)  8.453079e-01
## s(X.tmat):L.X 2.085683e-12
```

Nonilnear scalar-on-function

Nonlinear Scalar-on-Function

With the `refund` package we can actually fit a wide range of models, including nonlinear ones. Suppose what we want to build a nonlinear relationship between a scalar outcome, Y, and a functional outcome X. Unfortunately, this is a fairly challenging task. If X were $\mathbb{R}^p$ dimensional, and we wanted to fit the model

$$Y_i = f(X_i) + \varepsilon_i,$$

then the minimax rate for estimating $f^{(m)}$ is

$$N^{-\frac{p-m}{2p+d}},$$

where d is the dimension and p is the number of derivatives f posseses.

Nonlinear Scalar-on-Function

Minimax rate for estimating $f^{(m)}$ is

$$N^{-\frac{p-m}{2p+d}},$$

where d is the dimension and p is the number of derivatives f posseses.

- As $p \to \infty$, then this value converges to $1/2$ (the parametric rate).
- For $m = 0$, $d = 1$, and $p = 2$ then the rate is $2/5$ (the nonparametric rate).
- However, the key point here is that you can't let $d \to \infty$ as the rate then becomes 0.

Additive Models

To get around this problem, one can impose a bit of structure. Allowing a completely general form for f is too difficult a problem. So, we make the simplifying assumption that

$$Y_i = \int f(X_i(t), t)\ dt + \varepsilon_i.$$

This is called a *functionally additive model*, and it does not have the same “curse of dimensionality” problem.

Additive Models

One can estimate f using a bivariate basis. A common approach is to use a tensor basis:

$$f(x,t) \approx \sum_{j=1}^{J} \sum_{l=1}^{L} f_{jl} B_j^{\star}(x) B_l(t),$$

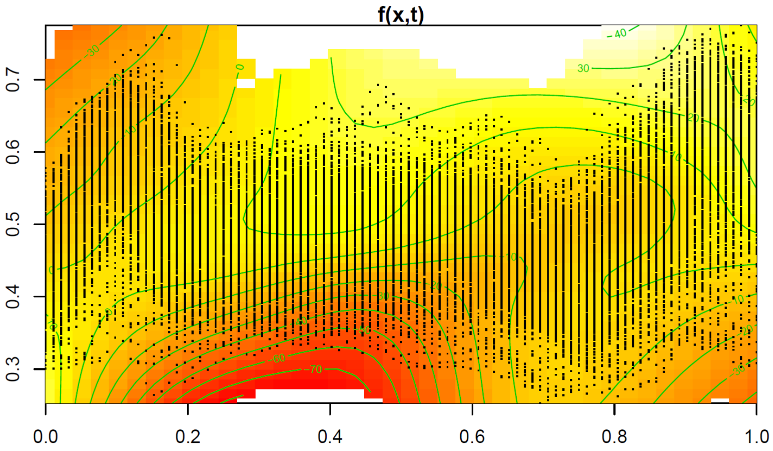
where $B_j^{\star}$ and B_l are two different sets of basis functions. One thing to keep in mind is that while $t \in [0,1]$, x could theoretically be any real value. One approach to is therefore to normalize the $X_i(t)$ to be between 0 and 1.

Additive Models

```
pfr.pen<-pfr(Y~af(X,argvals=grid,k=c(10,10)))
pfr.pen$aic

## [1] 2564.45

rbind(summary(pfr.pen)[[22]], summary(pfr.pen)[[24]])

##                        Estimate Std. Error
## (Intercept)            47.33275  0.7518583
## te(X.tmat,X.omat):L.X 10.11743 13.8607157
##                         t value      Pr(>|t|)
## (Intercept)            62.95435 2.30158e-183
## te(X.tmat,X.omat):L.X  6.50856  1.04344e-11
```

Additive Models

```
par(mar=c(2,2,1,1))
plot(pfr.pen,scheme=2,xlab="t",ylab="x",main="f(x,t)")
```



Some warnings

It is worth noting that these models can be notoriously difficult to interpret. If $f(x,t)$ is all positive/negative, then we can understand the values as the relative contributions to Y. However, what if there are both positive and negative values?