Lecture 2. FDA Basic - First steps

Functional Data Analysis

Jun Song

Department of Statistics
Korea University

Outline

We continue with Chapter 1 - First Steps.

- Mean functions
- Covariance functions
- Functional principal components
- Two examples

We will focus especially on how to use bases for various computations.

Data structure

Assume we have an iid sample of functions

$$X_n(t) \qquad n = 1, \ldots, N \qquad 0 \le t \le 1.$$

What does it mean to be iid? From here on, we will assume that we can construct these functions in `R` as fd objects.

Mean Function

pointwise

Population level:

$$\mu(t) := \mathrm{E}[X_n(t)].$$

Sample level:

$$\widehat{\mu}(t) := \frac{1}{N}\sum_{n=1}^{N} X_n(t).$$

In later chapters, we will discuss asymptotic properties of $\widehat{\mu}(t)$.

Mean Function - Computation

We can use the basis expansion of $X_n(t)$ to do this computation fairly easily. Recall that

$$X_n(t) = \sum_m c_{nm} B_m(t).$$

So one has

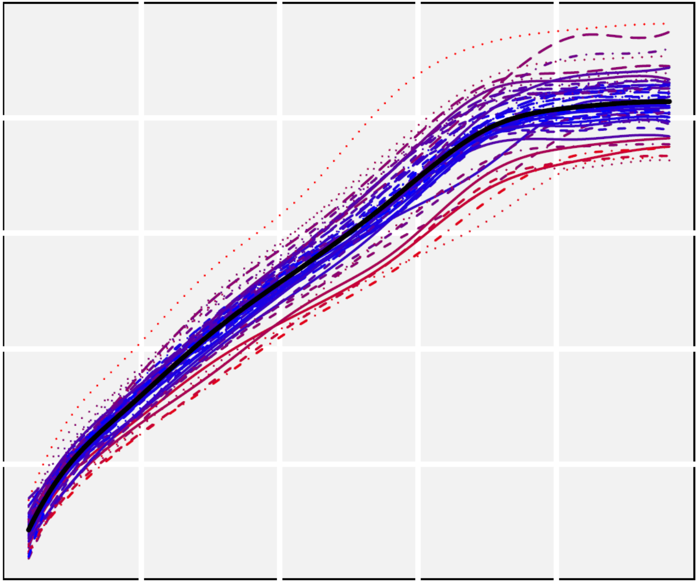
$$\widehat{\mu}(t) = \frac{1}{N} \sum_n \sum_m c_{nm} B_m(t) = \sum_m \bar{c}_m B_m(t),$$

where

$$\bar{c}_m = \frac{1}{N} \sum_n c_{nm}.$$

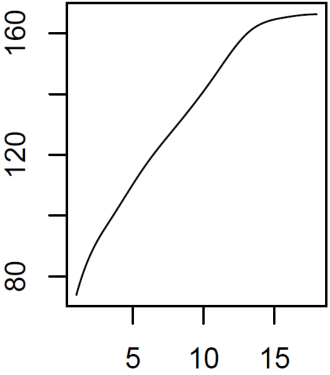
As in univariate and multivariate statistics, we use the mean to summarize the center/average value of the data. We can also visualize how this value changes with time.

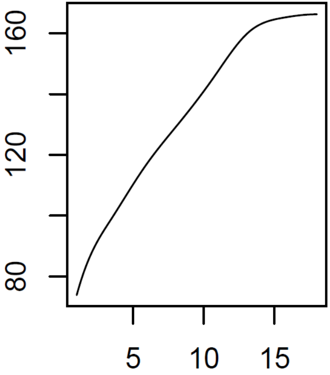
Mean Function - Berkeley Growth



Mean Function - Alt Calc

```r
GHeight_coef<-GHeight.F$coefs
mean_coef<-rowMeans(GHeight_coef)
mu.F2<-fd(coef=mean_coef,basisobj=my_basis)
plot(mu.F,ylab="",xlab="");plot(mu.F2,ylab="",xlab="")
```





Covariance Function

Covariance function can be defined as

- Bivariate function $C(s, t)$ where s, t are in the domain of functions.
- Linear operator from $\mathcal{H}$ to $\mathcal{H}$, where $\mathcal{H}$ is the function space for the data.

We will look at the bivariate-function version first. The linear operator version will be discussed later.

Covariance Fucntion

Population level:

$$C(t,s) := \mathrm{E}[(X_n(t) - \mu(t))(X_n(s) - \mu(s))].$$

Sample level:

$$\widehat{C}(t,s) := \frac{1}{N-1}\sum_{n=1}^{N}(X_n(t) - \widehat{\mu}(t))(X_n(s) - \widehat{\mu}(s)).$$

In later chapters, we will discuss asymptotic properties of $\widehat{C}(t,s)$. As in multivariate statistics, C describes how the different time points covary, i.e. how dependent is your height today and 10 years ago?

Covariance Function - Computation

Let $\tilde{c}_{nm} = c_{nm} - \bar{c}_m$ be the centered coefficients and $\tilde{\mathbf{c}}$ be the matrix of centered coefficients. Then

$$\begin{aligned}\widehat{C}(t,s) &= \frac{1}{N-1}\sum_n \sum_{m_1} \sum_{m_2} \tilde{c}_{nm_1}\tilde{c}_{nm_2} B_{m_1}(t) B_{m_2}(s) \\ &= \frac{1}{N-1}\sum_{m_1}\sum_{m_2} (\tilde{\mathbf{c}}^\top \tilde{\mathbf{c}})_{m_1,m_2} B_{m_1}(t) B_{m_2}(s) \\ &= \sum_{m_1}\sum_{m_2} (\boldsymbol{\Sigma}_c)_{m_1 m_2} B_{m_1}(t) B_{m_2}(s).\end{aligned}$$

So $\widehat{C}(t,s)$ can be expressed using a basis expansion. The basis $\{B_{m_1}(t)B_{m_2}(s) : m_1 = 1,\dots,M,\ m_2 = 1,\dots,M\}$ is called a *tensor basis*. The coefficients are given by $(N-1)\mathbf{c}^\top\mathbf{c}$.

Covariance Function - `bifd`

```
GHeight_var<-var.fd(GHeight.F)
class(GHeight_var)

## [1] "bifd"

names(GHeight_var)

## [1] "coefs"      "sbasis"     "tbasis"     "bifdnames"

dim(GHeight_var$coefs)

## [1] 10 10
```

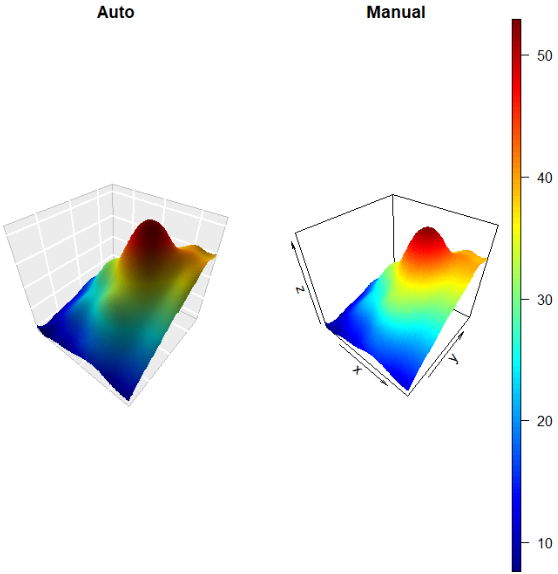
The covariance function is a *bivariate functional object* in R. *coefs* represents the coefficient of basis expansion for the bivariate covariance function.

Covariance Function - Manual

```r
# Manual
coef_mat<-coef(GHeight.F); N<-dim(coef_mat)[2]
coef_mean<-rowMeans(coef_mat)
coef_center<-sweep(coef_mat,1,coef_mean)
C_hat_coef<-(coef_center%*%t(coef_center))/(N-1)
C_hat<-bifd(coef=t(C_hat_coef),sbasisobj = my_basis,
            tbasisobj = my_basis)
```

Warning: Remember that the `fda` package assumes that different curves correspond to different columns in the coefficient matrix (the transpose of how we defined $\mathbf{c}$).

Covariance Function - Manual vs Auto



Functional Principal Components

As was mentioned, Principal Components are the eigenfunctions (vectors) of the covariance function (matrix). These are a cornerstone of FDA and multivariate statistics, so we will give them more attention later on. An eigenvalue/function pair (λ_j, v_j) satisfies

$$\text{Population:} \quad \lambda_j v_j(t) = \int_0^1 C(t,s) v_j(s)\, ds \quad \text{with} \quad \int v_j(t)^2\, dt = 1.$$

$$\text{Sample:} \quad \hat{\lambda}_j \hat{v}_j(t) = \int_0^1 \widehat{C}(t,s) \hat{v}_j(s)\, ds \quad \text{with} \quad \int \hat{v}_j(t)^2\, dt = 1.$$

More details/theory on this later.

FPCA - Calc

These next steps are a lot easier if $B_m(t)$ satisfy:

$$\langle B_{m_1}, B_{m_2} \rangle_{L_2} = \int B_{m_1}(t) B_{m_2}(t)\ dt = 1_{m_1 = m_2}.$$

So we will assume this for a few slides (called orthonormal). If we expand $v_j(t)$ using the $B_j(t)$:

$$v_j(t) = \sum_m v_{jm} B_m(t),$$

then we can obtain a set of linear equations involving $\mathbf{v}_j = \{v_{jm}\}$ by integrating against $B_m(t)$:

$$\lambda_j \int v_j(t) B_{m_1}(t)\ dt = \int_0^1 \int_0^1 C(t,s) v_j(s) B_{m_1}(t)\ ds\ dt.$$

Notice that

$$\lambda_j \int v_j(t) B_{m_1}(t) = \lambda_j \sum_{m_2} v_{jm_2} \int B_{m_1}(t) B_{m_2}(t)\ dt = \lambda_j v_{jm_1}.$$

FPCA - Calc (cont)

We also have that

$$\begin{aligned}
&\int_0^1 \int_0^1 C(t,s) v_j(s) B_{m_1}(t) \; ds \; dt \\
&= \sum_{m_2} \sum_{m_3} (\boldsymbol{\Sigma}_c)_{m_2 m_3} \int B_{m_2}(t) B_{m_1}(t) \; dt \int B_{m_3}(s) v_j(s) \; ds \\
&= \sum_{m_3} (\boldsymbol{\Sigma}_c)_{m_1 m_3} v_{j m_3}.
\end{aligned}$$

So the coefficients, $\mathbf{v}_j$, for the eigenfunction v_j, satisfies

$$\lambda_j \mathbf{v}_j = \boldsymbol{\Sigma}_c \mathbf{v}_j \quad \text{and} \quad \mathbf{v}_j^\top \mathbf{v}_j = 1.$$

FPCA - Nonorthonormal basis

If the $B_m(t)$ are not orthonormal, then the equations are a bit uglier. However, any basis can be made orthogonal via some linear (matrix) transformation $\mathbf{W}$, that is

$$\tilde{B}_m(t) = \sum_{m_1} W_{mm_1} B_m(t),$$

are an orthonormal basis. This implies that

$$\begin{aligned} v_j(t) = \sum_m v_{jm} B_m(t) &= \sum_m v_{jm} \sum_{m_1} (\mathbf{W}^{-1})_{mm_1} \tilde{B}_{m_1}(t) \\ &= \sum_{m_1} (\mathbf{W}^{-1}\mathbf{v})_{m_1} \tilde{B}_{m_1}(t). \end{aligned}$$

And so $\mathbf{W}$ provides a change of basis transformation $\tilde{\mathbf{v}}_j = \mathbf{W}^{-1}\mathbf{v}_j$.

FPCA - Nonorthonormal basis - Cont

Similar arguments show that

$$\widetilde{\mathbf{\Sigma}}_c = \mathbf{W}^{-1}\mathbf{\Sigma}_c\mathbf{W}^{-1},$$

and we therefore have a relation

$$\lambda_j \tilde{\mathbf{v}}_j = \widetilde{\mathbf{\Sigma}}_c \tilde{\mathbf{v}}_j \text{ and } \lambda_j \mathbf{W}^{-1}\mathbf{v}_j = \mathbf{W}^{-1}\mathbf{\Sigma}_c\mathbf{W}^{-2}\mathbf{v}_j$$
$$\text{with } \mathbf{v}_j^\top \mathbf{W}^{-2}\mathbf{v}_j = 1.$$

FPCA in R

```
GHeight_pc<-pca.fd(GHeight.F,nharm=3)
class(GHeight_pc)

## [1] "pca.fd"

names(GHeight_pc)[1:3]; names(GHeight_pc)[4:5]

## [1] "harmonics" "values"    "scores"
## [1] "varprop" "meanfd"
```

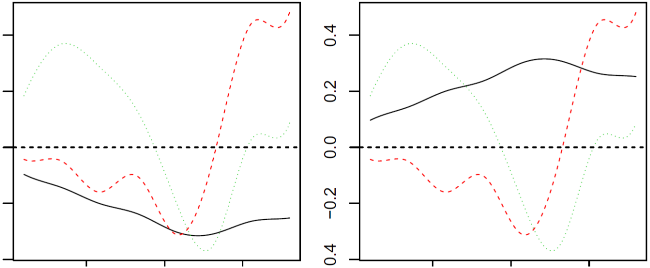
- *nharm*: the number of desired pcs
- *harmonics*: these are the principal components
- *values*: these are the eigenvalues
- *scores*: these are the coefficients in the basis expansion
- *varprop*: these are the explained variances for each pc
- *meanfd*: mean function of the data

FPCA in R - manual

```
library(expm);
W2inv<-inprod(my_basis,my_basis);
Sig_c<-cov(t(coef_mat)); A<-Sig_c%*%W2inv
e_A<-eigen(A)
names(e_A)

## [1] "values"  "vectors"

e_A$values[1:3]

## [1] 493.16109  35.23920  14.03662

GHeight_pc$values[1:3]

## [1] 484.02773  34.58686  13.77659
```

FPCA in R - manual (cont)

```
v_coef<-e_A$vectors[,1:3]
norm_v<-diag(t(v_coef)%*%W2inv%*%v_coef)
v_coef<-v_coef%*%diag(1/sqrt(norm_v))
e_fun<-fd(v_coef,my_basis)
plot(e_fun); plot(GHeight_pc$harmonics)

## [1] "done"
```



Note the scales are off for reasons we will discuss later on.

FPCA - Scores

A primary use of FPCA is dimension reduction. High/infinite dimensional objects can be represented using a fairly small number of FPCs:

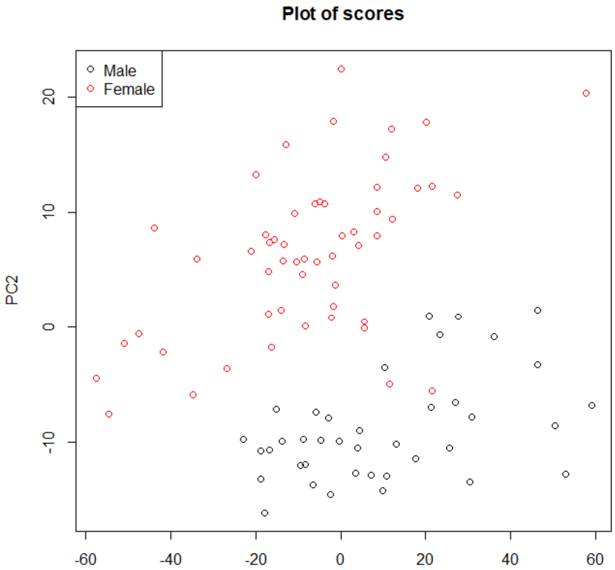
$$X_n(t) - \bar{X}(t) \approx \sum_{j=1}^{p} \hat{\xi}_{nj} \hat{v}_j(t),$$

where

$$\hat{\xi}_{nj} = \int (X_n(t) - \bar{X}(t)) \hat{v}_j(t).$$

These scores can be used for a variety of purposes. Sometimes we just can't work with infinite-dimensional objects. At other times, the scores may prove useful to examine.

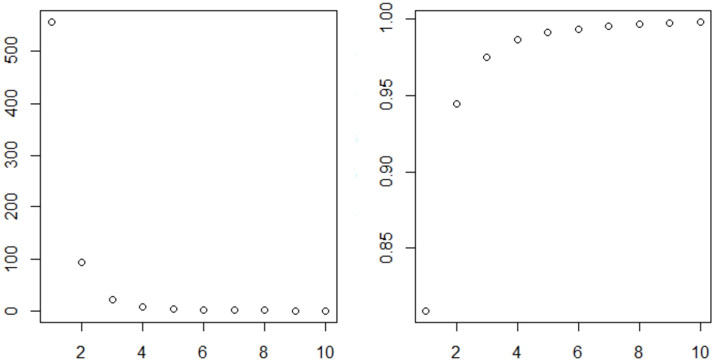
FPCA - Scores - Berkeley, both genders



FPCA - Explained Variance - Berkeley

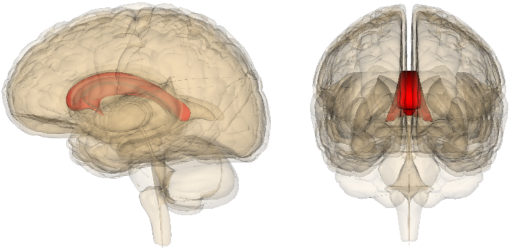
The first p eigenfunctions explain the following proportion of variance

$$\sum_{j=1}^{p} \lambda_j / \sum_{j=1}^{\infty} \hat{\lambda}_j.$$



Diffusion Tensor Imaging

DTI data is part of the `refund` package in `R`. This package is composed of tools for functional regression. The data consists of scans of the *corpus callosum*, a white matter tract in the brain connecting the two hemispheres of the brain. The data was collected as part of a study on Multiple Sclerosis and its effect on the brain. Note:"The MRI/DTI data were collected at Johns Hopkins University and the Kennedy-Krieger Institute", an image from Wikipedia.

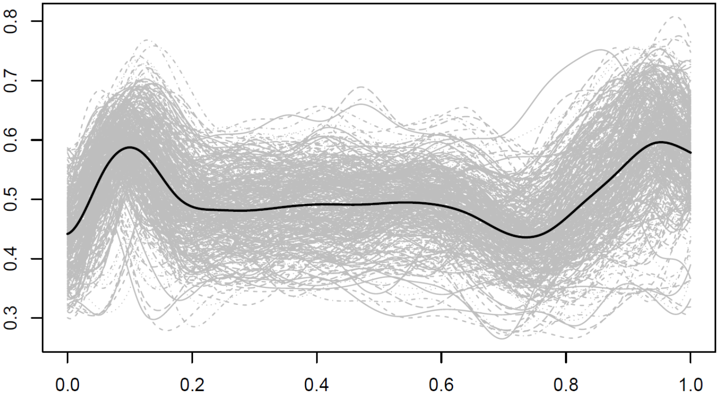


Diffusion Tensor Imaging

```
Corp<-DTI$cca
drop<-unique(which(is.na(Corp),arr.ind=TRUE)[,1])
Corp<-Corp[-drop,] # Missing value
pts<-seq(0,1,length=93)
my_basis<-create.bspline.basis(c(0,1),
              nbasis=100,norder=6)
lambda_all<-10^(-(10:20)/2)
gcv_all<-numeric(0)
for(lambda in lambda_all){
myPar<-fdPar(my_basis,2,lambda)
Corp.F<-smooth.basis(pts,t(Corp),myPar)
gcv_all<-c(gcv_all,mean(Corp.F$gcv))}
```

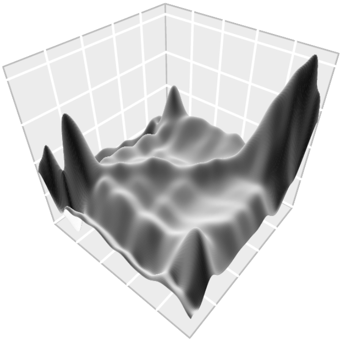
DTI-Mean

```
plot(Corp.F,col="gray")
Corp.F.mean<-mean(Corp.F)
plot(Corp.F.mean,add=TRUE,lwd=2)
```



DTI-Covariance

```
Cov_DTI<-var.fd(Corp.F)
pts<-seq(0,1,length=200)
Cov_DTI_mat<-eval.bifd(pts,pts,Cov_DTI)
persp3D(pts,pts,Cov_DTI_mat,axes=FALSE,colkey=FALSE,bt
```

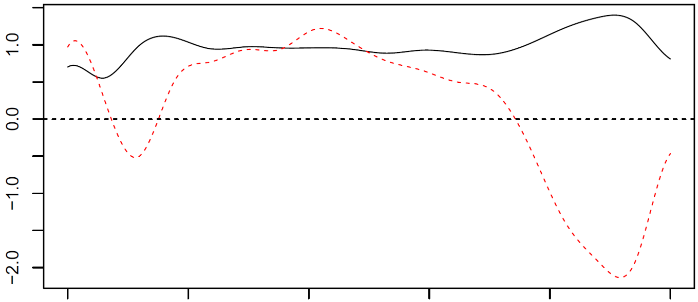


DTI-FPCA

```
DTI_pc<-pca.fd(Corp.F,nharm=2)
DTI_pc$varprop
```

```
## [1] 0.64858233 0.08276098
```

```
plot(DTI_pc$harmonics)
```



DTI-FPCA as an approximation

```
DTI_e_fun<-DTI_pc$harmonics
DTI_scores<-DTI_pc$scores
dim(DTI_scores)
approx_fun<-Corp.F.mean+DTI_e_fun[1]*DTI_scores[1,1]+
  DTI_e_fun[2]*DTI_scores[1,2]
plot(Corp.F[1]); plot(approx_fun,add=TRUE,lty=2)
```