Chapter 5

Longitudinal Time Series Models

5.1 Functional Data Analysis

5.1.1 Elements of Functional Data Analysis

- Functional data is multivariate data with an ordering on the dimensions $y_{it_1}, y_{it_2}, \ldots, y_{it_n}$, where ordering t is often defined by time.
- In functional data analysis, data consist of (at least) pair of discrete observed values of time T variable and the response variable Y for each sampling unit i:

$$\mathbf{t} = \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_n \end{pmatrix}, \qquad \mathbf{y}_i = \begin{pmatrix} y_{it_1} \\ y_{it_2} \\ \vdots \\ y_{it_n} \end{pmatrix}, \qquad i = 1, 2, \ldots, N.$$

- For each sampling unit i, the observed values $\mathbf{y}_i = (y_{it_1}, y_{it_2}, \ldots, y_{it_n})'$ are assumed to be realizations from the equation

$$y_{it_j} = y_i(t_j) + \varepsilon_{it_j}, \tag{5.1}$$

where $y_i(t_j)$ is smooth function (curve) that fits to the data as well as reasonable, and ε_{it_j} is random noise associated mostly to the measurement error.
- Often in analysis, the functional $y_i(t)$, is further decomposed to parts, but starting point for the analysis is "smoothing", i.e., estimating the functional $y_i(t)$ for each sampling unit i that so that estimated functionals $\hat{y}_i(t)$ are fitting to the data as well as reasonable.

– In functional analysis, the function $y_i(t)$ is considered to have a form

$$y_i(t) = \sum_{k=1}^{K} \phi_k(t)c_{ik} = \boldsymbol{\phi}(t)'\mathbf{c}_i, \tag{5.2}$$

where $\boldsymbol{\phi}(t) = (\phi_1(t), \phi_2(t), \ldots, \phi_K(t))'$ are some basis functions and $\mathbf{c}_i = (c_{i1}, c_{i2}, \ldots, c_{iK})'$ are coefficients for i:th observation in the *basis function expansion*.

– The discrete observed values $\mathbf{y}_i = (y_{it_1}, y_{it_2}, \ldots, y_{it_n})'$ can be written in matrix form as

$$\mathbf{y}_i = \boldsymbol{\Phi}\mathbf{c}_i + \boldsymbol{\varepsilon}_i, \tag{5.3}$$

where

$$\boldsymbol{\Phi} = \begin{pmatrix} \boldsymbol{\phi}(t_1)' \\ \boldsymbol{\phi}(t_2)' \\ \vdots \\ \boldsymbol{\phi}(t_n)' \end{pmatrix}, \qquad \boldsymbol{\varepsilon}_i = \begin{pmatrix} \varepsilon_{it_1} \\ \varepsilon_{it_2} \\ \vdots \\ \varepsilon_{it_n} \end{pmatrix}$$

– The basis functions can be

Constant: $\phi(t) = 1$,

Monomial: $\phi_1(t) = 1, \phi_2(t) = t, \phi_3(t) = t^2, \ldots, \phi_K = t^{(K-1)}$,

Exponential: $\phi_1(t) = e^{\omega_1 t}, \phi_2(t) = e^{\omega_2 t}, \ldots, \phi_K = e^{\omega_K t}$,

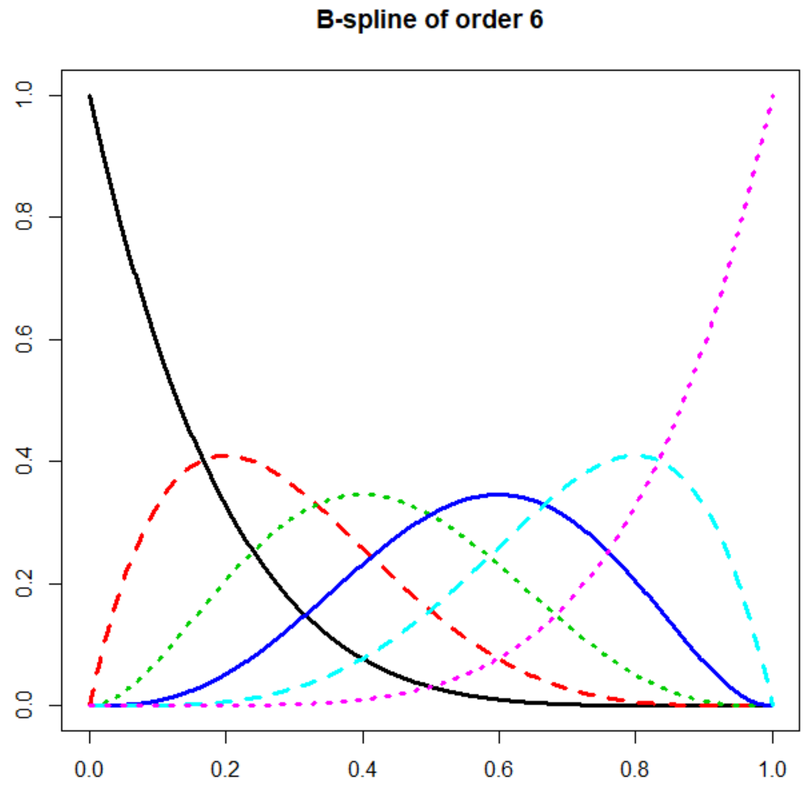
Power: $\phi_1(t) = t^{\omega_1}, \phi_2(t) = t^{\omega_2}, \ldots, \phi_K = t^{\omega_K}$.

– If functions $f_i(t)$ are required to repeat themselves over a certain period T, then Fourier series are suitable basis functions

Fourier: $\phi_1(t) = 1, \phi_2(t) = \sin(\omega t), \phi_3(t) = \cos(\omega t), \phi_4(t) = \sin(2\omega t), \phi_5(t) = \cos(2\omega t), \ldots,$

where the constant ω is related to the period T by $\omega = \frac{2\pi}{T}$.

– Most often used basis functions are B-splines in smoothing. Splines are piecewise polynomials, and are constructed by dividing interval of observation into subintervals with boundaries at points called *break points*. Over any subinterval, the spline function is a polynomial of fixed degree, but nature of the polynomial changes as one passes into the next subinterval.



5.1.2 Estimation in Smoothing of Functional Data

– In functional $y_i(t) = \boldsymbol{\phi}(t)'\mathbf{c}_i$, the vector of coefficients $\mathbf{c}_i$ is unknown and needs to be estimated by the observed discrete values $\mathbf{y}_i = \boldsymbol{\Phi}\mathbf{c}_i + \boldsymbol{\varepsilon}_i$.

– An estimate $\hat{\mathbf{c}}_i$ of $\mathbf{c}_i$ is called the least squares estimate, if it is the solution for the following minimizing problem

$$\min_{\mathbf{c}_i}(\mathbf{y}_i - \boldsymbol{\Phi}\mathbf{c}_i)'(\mathbf{y}_i - \boldsymbol{\Phi}\mathbf{c}_i) = \min_{\mathbf{c}_i}\left[\mathbf{y}_i'\mathbf{y}_i - 2\mathbf{y}_i'\boldsymbol{\Phi}\mathbf{c}_i + \mathbf{c}_i'\boldsymbol{\Phi}'\boldsymbol{\Phi}\mathbf{c}_i\right]. \tag{5.4}$$

– By process of differentiation, we can find least squares estimate, since

$$\frac{\partial(\mathbf{y}_i - \boldsymbol{\Phi}\mathbf{c}_i)'(\mathbf{y}_i - \boldsymbol{\Phi}\mathbf{c}_i)}{\partial\mathbf{c}_i} = -2\boldsymbol{\Phi}'\mathbf{y}_i + 2\boldsymbol{\Phi}'\boldsymbol{\Phi}\mathbf{c}_i. \tag{5.5}$$

– Setting now above derivative to equal zero, and solving with respect $\mathbf{c}_i$, we get

$$\begin{aligned} -2\boldsymbol{\Phi}'\mathbf{y}_i + 2\boldsymbol{\Phi}'\boldsymbol{\Phi}\hat{\mathbf{c}}_i &= \mathbf{0} \\ \boldsymbol{\Phi}'\boldsymbol{\Phi}\hat{\mathbf{c}}_i &= \boldsymbol{\Phi}'\mathbf{y}_i \\ \hat{\mathbf{c}}_i &= (\boldsymbol{\Phi}'\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}'\mathbf{y}_i. \end{aligned} \tag{5.6}$$

– Estimated functional has the form $\hat{y}_i(t) = \boldsymbol{\phi}(t)'\hat{\mathbf{c}}_i$, and the discrete fitted values are

$$\hat{\mathbf{y}}_i = \boldsymbol{\Phi}\hat{\mathbf{c}}_i = \boldsymbol{\Phi}(\boldsymbol{\Phi}'\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}'\mathbf{y}_i.$$

– Another approach for estimation of $\mathbf{c}_i$ is smoothing with roughness penalties. An estimate $\hat{\mathbf{c}}_i$ of $\mathbf{c}_i$ is called the penalized least square estimate, if it is the solution for the following minimizing problem

$$\min_{\mathbf{c}_i}(\mathbf{y}_i - \mathbf{\Phi}\mathbf{c}_i)'(\mathbf{y}_i - \mathbf{\Phi}\mathbf{c}_i) + \lambda\mathbf{c}_i'\mathbf{R}\mathbf{c}_i, \tag{5.7}$$

where roughness penalty matrix $\mathbf{R} = \int \left(\frac{d^2\boldsymbol{\phi}(t)}{dt}\right)\left(\frac{d^2\boldsymbol{\phi}(t)}{dtdt}\right)' dt$ measures curvature of the basis functions $\boldsymbol{\phi}(t)$.

– It can be shown that the penalized least square estimate is

$$\hat{\mathbf{c}}_i = (\mathbf{\Phi}'\mathbf{\Phi} + \lambda\mathbf{R})^{-1}\mathbf{\Phi}'\mathbf{y}_i, \tag{5.8}$$

where the smoothing parameter λ can be chosen by minimizing the generalized cross-validation measure

$$GCV(\lambda) = \frac{n \cdot \sum_{i=1}^{N}(\mathbf{y}_i - \hat{\mathbf{y}}_i)'(\mathbf{y}_i - \hat{\mathbf{y}}_i)}{(n - df(\lambda))^2}, \quad df(\lambda) = \mathrm{trace}(\mathbf{\Phi}(\mathbf{\Phi}'\mathbf{\Phi} + \lambda\mathbf{R})^{-1}\mathbf{\Phi}'). \tag{5.9}$$

– Distinctive feature of the functional data analysis is the use of derivates $\hat{y}_i'(t), \hat{y}_i''(t), \ldots, \hat{y}_i^{(m)}(t)$ in analysis.

– Derivative of the estimated functional $\hat{y}_i(t) = \boldsymbol{\phi}(t)'\hat{\mathbf{c}}_i$ is

$$\hat{y}_i'(t) = \left(\frac{d\,\boldsymbol{\phi}(t)}{d\,t}\right)'\hat{\mathbf{c}}_i = \boldsymbol{\phi}'(t)'\hat{\mathbf{c}}_i. \tag{5.10}$$

Example 5.1.

Diffusion tensor imaging, DTI, is a magnetic resonance imaging methodology which is used to measure the diffusion of water in the brain. Water diffuses isotropically (i.e. the same in all directions) in the brain except in white mater where it diffuses anisotropically (i.e. differently in different directions). This allows researchers to utilize DTI to generate images of white matter in the brain. We consider fractional anisotropy tract profiles of the corpus callosum, a portion of the data DTI data set in the refund R package. The corpus collosum, the largest white matter structure in the brain, is a bundle of nerve fibers connecting the two hemispheres of the brain. Fractional anisotropy is a value between 0 and 1 which measures the level of anisotropy, and therefore the quantity of white matter, at a particular location. A total of 376 patients are considered, with each tract measured at 93 equally spaced locations. The arguments t_{ij} thus denote a spatial rather than temporal location. The resulting observed curves are shown in figure below.

```
> library(refund)
> data(DTI)
> Y<-t(DTI$cca[-c(125,126,130,131,319,321),])  # removal of missing values and transpose of the data
> t<-seq(0,1,length=dim(Y)[1])
> Y[1:5,1:10]
        1001_1    1002_1    1003_1    1004_1    1005_1    1006_1    1007_1    1008_1    1009_1    1010_1
cca_1  0.4909345 0.4721627 0.5023738 0.4021894 0.4018747 0.4507296 0.5537167 0.4477326 0.4953400 0.4580893
```

```
cca_2  0.5168018 0.4868219 0.5136516 0.4225127 0.4055805 0.4535052 0.5577723 0.4807621 0.5057313 0.4705768
cca_3  0.5356539 0.5022577 0.5392542 0.4398983 0.3985487 0.4575306 0.5604173 0.5024963 0.5176728 0.4796741
cca_4  0.5553587 0.5233635 0.5742101 0.4600235 0.3860009 0.4655026 0.5710832 0.5181574 0.5409608 0.4886211
cca_5  0.5927610 0.5524401 0.6031339 0.4751297 0.4088248 0.4804154 0.5848086 0.5304736 0.5834464 0.5056029
>matplot(t,Y, type="n")
>matlines(t,Y, col="red")
```

(a) Convert the Y dataset to functional objects by using $K = 10$ B-splines of order four (cubic splines) and knots with equal intervals, and plot the 376 smoothed curves on one graph. Use t as the "time" variable, and do the smoothing by using the least squares method to estimate coefficient vectors $\mathbf{c}_i$.

(b) Convert the Y dataset to functional objects by using $K = 100$ B-splines of order four (cubic splines) and knots with equal intervals, and plot the 376 smoothed curves on one graph. Use t as the "time" variable, and do the smoothing by using the penalized least squares method with roughness penalty being the square of the second derivative $[D^2 y_i(t)]^2$ to estimate coefficient vectors $\mathbf{c}_i$. You may use the value $\lambda = 0.01$ as the weight to the roughness penalty.

(c) Find the "best" smoothing parameter value λ when everything else are kept same as in (b). Use the generalized cross validation as the criteria for the "best" λ value, and start your search with λ set as $\lambda = 0.00000001$.

Example 5.2.

Consider the dataset gait in library fda. Particularly, consider the Knee Angle part of the dataset.

```
> library(fda)
> gait[,,"Knee Angle"]
      boy1 boy2 boy3 boy4 boy5 boy6 boy7 boy8 boy9 boy10 boy11 boy12 boy13   ...
0.025   10   16   18    5    2   15   13   14   15     9    13     7     9
0.075   15   25   27   14    6   17   16   17   20    22    24     8    14
0.125   18   28   32   16    6   23   22   18   23    25    27    11    16
0.175   18   25   32   17    6   23   17   19   26    21    23    12    15
0.225   15   18   28   10    5   20   12   19   25    10    18     8    15
.
```

Convert the Knee Angle values to estimated functional objects $\hat{y}_i(t)$ by using Fourier basis functions. Do the smoothing by using the penalized least squares method. Use appropriate roughness penalty in smoothing, and choose smoothing parameter λ some sensible way. Also, investigate how the estimates of first derivatives $y_i'(t)$ are behaving and how the fitted values $\hat{\mathbf{y}}_i$ are behaving compare to observed ones $\mathbf{y}_i$.

5.1.3 Prediction in Smoothing of Functional Data

– Let $\hat{y}_1(t), \hat{y}_2(t), \ldots, \hat{y}_N(t)$ be smoothed estimated functionals. Then the sample mean functional is defined as

$$\bar{y}(t) = \frac{\hat{y}_1(t) + \hat{y}_2(t) + \cdots + \hat{y}_N(t)}{N}. \tag{5.11}$$

– Consider the prediction of the value $y_{if} = y_i(t_f)$ when the time point t_f is outside the interval $t_f \notin [t_1, t_n]$. Then one may consider extending the domain of the basis functions $\boldsymbol{\phi}(t)$ to the interval $[t_1, t_f]$ and the calculating prediction $\hat{y}_{if} = \hat{y}_i(t_f) = \boldsymbol{\phi}(t_f)'\hat{\mathbf{c}}_i$.

– Let us assumed that functionals $y_i(t)$ have the following decomposition

$$y_i(t) = \mu(t) + u_i(t), \tag{5.12}$$

where the expectation functional $\mu(t)$ can be estimated by the sample mean $\hat{\mu}(t) = \bar{y}(t)$, the random error functional $u_i(t)$ has an expectation as $\mathrm{E}(u_i(t)) = 0$.

– Then discrete observed values has the decomposition

$$\mathbf{y}_i = \boldsymbol{\mu} + \mathbf{v}_i = \boldsymbol{\Phi}\mathbf{c}_\mu + \mathbf{v}_i, \tag{5.13}$$

where $\mathbf{v}_i$ contains the random error term and random noise term $\mathbf{v}_i = \mathbf{u}_i + \boldsymbol{\varepsilon}_i$ with

$$\mathrm{Cov}(\mathbf{v}_i) = \boldsymbol{\Sigma}. \tag{5.14}$$

– The covariance matrix $\mathbf{\Sigma}$ is estimated as

$$\widehat{\mathbf{V}} = \mathbf{Y} - \hat{\boldsymbol{\mu}}\mathbf{1}' = \mathbf{Y} - \mathbf{\Phi}\hat{\mathbf{c}}_\mu\mathbf{1}' \tag{5.15}$$

$$\widehat{\mathbf{\Sigma}} = \frac{\widehat{\mathbf{V}}\widehat{\mathbf{V}}'}{n-1}. \tag{5.16}$$

– Consider the prediction in a partitioned random vector $\mathbf{y}_f$ situation

$$\mathbf{y}_f = \begin{pmatrix} \mathbf{y}_{f1} \\ \mathbf{y}_{f2} \end{pmatrix} = \begin{pmatrix} \mathbf{\Phi}_1 \\ \mathbf{\Phi}_2 \end{pmatrix} \mathbf{c}_\mu + \begin{pmatrix} \mathbf{u}_{f1} \\ \mathbf{u}_{f2} \end{pmatrix}. \tag{5.17}$$

– The best empirical predictor for the vector $\mathbf{y}_{f2}$ is

$$\hat{\mathbf{y}}_{f2} = \widehat{\mathrm{BP}}(\mathbf{y}_{f2}|\mathbf{Y}, \mathbf{y}_{f1}) = \mathbf{\Phi}_2\hat{\mathbf{c}}_\mu + \widehat{\mathbf{\Sigma}}_{21}\widehat{\mathbf{\Sigma}}_{11}^{-1}(\mathbf{y}_{f1} - \mathbf{\Phi}_1\hat{\mathbf{c}}_\mu). \tag{5.18}$$

– The predictive functional $\hat{y}_f(t)$ is then

$$\hat{y}_f(t) = \boldsymbol{\phi}(t)'\hat{\mathbf{c}}_f = \boldsymbol{\phi}(t)'(\mathbf{\Phi}'\mathbf{\Phi})^{-1}\mathbf{\Phi}' \begin{pmatrix} \mathbf{y}_{f1} \\ \hat{\mathbf{y}}_{f2} \end{pmatrix}. \tag{5.19}$$

Example 5.3.

Consider the following dataset files stageheight.txt and stagediameter.txt which are including height and diameter values for 18 trees measured for each tree in ten year interval.

```
> X<-as.matrix(read.table("stagediameter.txt", sep="\t", header=TRUE))
> Y<-as.matrix(read.table("stageheight.txt", sep="\t", header=TRUE))
> t<-seq(10,110,10)
> head(X)
   tree3 tree11 tree17 tree28 tree30 tree31 tree37 tree44 tree52 tree57 tree59 tree61 tree62 tree65 tree72 tree77 tree84 tree85
1 25.146  6.604 18.034 18.034 17.780 21.844  7.112 11.176  5.842 18.542 14.732 20.320 17.018 13.462  2.540  4.318 14.478  6.604
2 36.830 10.414 24.130 24.892 25.654 28.956 11.430 14.986 11.430 24.130 20.574 28.194 23.368 22.860  5.588  8.128 17.526 11.684
3 43.942 14.986 29.210 30.988 33.528 36.068 19.304 21.336 16.256 30.988 26.670 35.560 28.702 30.226 10.414 11.938 22.098 17.526
> head(Y)
     tree3   tree11   tree17   tree28   tree30   tree31   tree37   tree44   tree52   tree57   tree59   tree61   tree62   tree65   tree72   tr
1 17.37360  5.51688 14.38656 16.67256 16.00200 27.70632  6.24840  8.22960  4.20624 16.30680 16.03248 13.68552 15.48384  7.95528  2.07264  4.0
2 22.76856  9.08304 19.23288 23.71344 22.06752 34.01568 10.18032 10.39368  7.25424 21.54936 22.92096 19.87296 21.70176 12.49680  5.12064  8.5
3 26.67000 13.83792 21.91512 27.67584 26.88336 38.98392 17.46504 13.59408 11.06424 26.06040 27.76728 25.66416 24.56688 21.54936  8.93064 14.1
> matplot(t,X,type="n")
> matlines(t,X,col="blue")
> matplot(t,Y,type="n")
> matlines(t,Y,col="red")

The data are internal stem measures from 18 trees. The trees were selected as having been dominant throughout their
lives with no visible evidence of damage or forks. The trees came from stands throughout the inland range of the species.

X=dbhib - Diameter (cm.) at 1.37 m (4'6?) inside bark.
Y=height - Height of tree (m)
```

(a) Convert the Y dataset to functional objects $y_i(t) = \boldsymbol{\phi}'\mathbf{c}_{i_Y}$.

(b) Calculate the functional sample mean

$$\bar{y}(t) = \frac{\hat{y}_1(t) + \hat{y}_2(t) + \cdots + \hat{y}_N(t)}{N}$$

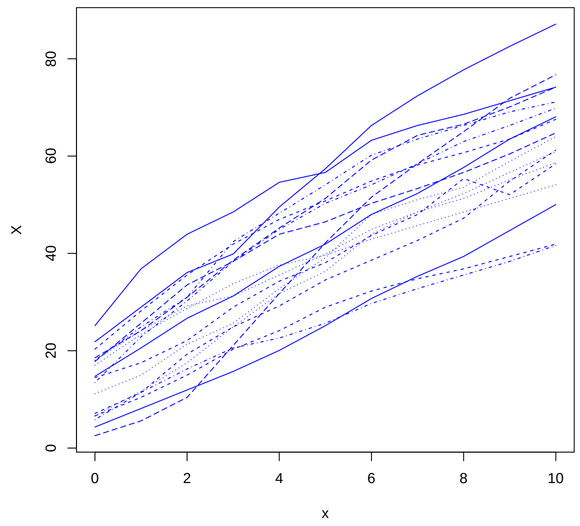
for the $Y = \text{height}$ variable.

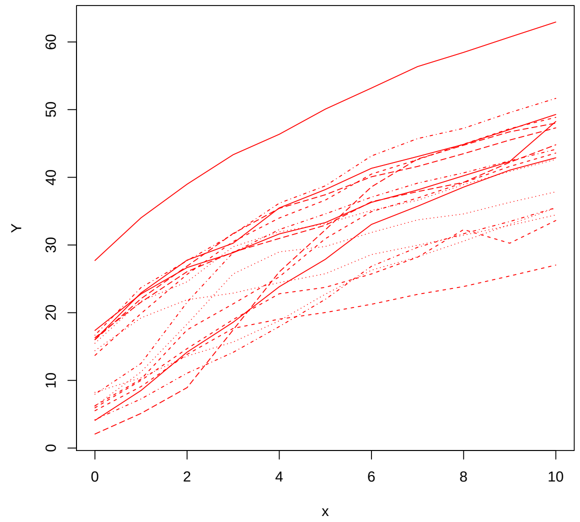
(c) Predict the values of $y_i(t)$, when $t \in (110, 120]$.

(d) Let a tree with observed measured values of height up to age of 40 being

```
[1,]   10 17.37360
[2,]   20 22.76856
[3,]   30 26.67000
[4,]   40 28.86456
```

Predict the height from the age of 40 to 110.





5.1.4 Linear Models for Functional Data

– Let $x_{i1}, x_{i2}, \ldots, x_{ip}$ be values of explanatory variables for the sampling unit i. Then the linear model for the functional $y_i(t)$ is

$$y_i(t) = \beta_0(t) + \beta_1(t)x_{i1} + \beta_2(t)x_{i2} + \cdots + \beta_p(t)x_{ip} + u_i(t), \tag{5.20}$$

where $\beta_0(t) = \boldsymbol{\phi}_{\beta_0}(t)'\mathbf{c}_{\beta_0}, \beta_1(t) = \boldsymbol{\phi}_{\beta_1}(t)'\mathbf{c}_{\beta_1}, \beta_2(t) = \boldsymbol{\phi}_{\beta_2}(t)'\mathbf{c}_{\beta_2}, \ldots, \beta_p(t) = \boldsymbol{\phi}_{\beta_p}(t)'\mathbf{c}_{\beta_p}$ are parameter functionals, and $u_i(t)$ is random error functional.

– If, for example, x_{i1} is a functional $x_{i1}(t) = \boldsymbol{\phi}_{x_1}(t)'\mathbf{c}_{x_1}$, the functional linear model is

$$y_i(t) = \beta_0(t) + \beta_1(t)x_{i1}(t) + \beta_2(t)x_{i2} + \cdots + \beta_p(t)x_{ip} + u_i(t). \tag{5.21}$$

– Estimates for unknown vectors of coefficients $\mathbf{c}_{\beta_0}, \mathbf{c}_{\beta_1}, \mathbf{c}_{\beta_2}, \ldots, \mathbf{c}_{\beta_p}$ can be found by minimizing integral

$$\min_{\mathbf{c}_{\beta_0},\mathbf{c}_{\beta_1},\mathbf{c}_{\beta_2},\ldots,\mathbf{c}_{\beta_p}} \int ||\mathbf{y}(t) - (\mathbf{1}\boldsymbol{\phi}_{\beta_0}(t)'\mathbf{c}_{\beta_0} + \mathbf{x}_1(t)\boldsymbol{\phi}_{\beta_1}(t)'\mathbf{c}_{\beta_1} + \cdots + \mathbf{x}_p(t)\boldsymbol{\phi}_{\beta_p}(t)'\mathbf{c}_{\beta_p})||^2 \, dt, \tag{5.22}$$

where, e.g.,

$$\mathbf{y}(t) = \begin{pmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_N(t) \end{pmatrix}.$$

Example 5.4.

Diffusion tensor imaging, DTI, is a magnetic resonance imaging methodology which is used to measure the diffusion of water in the brain. Water diffuses isotropically (i.e. the same in all directions) in the brain except in white mater where it diffuses anisotropically (i.e. differently in different directions). This allows researchers to utilize DTI to generate images of white matter in the brain. We consider fractional anisotropy tract profiles of the corpus callosum, a portion of the data DTI data set in the refund R package. The corpus collosum, the largest white matter structure in the brain, is a bundle of nerve fibers connecting the two hemispheres of the brain. Fractional anisotropy is a value between 0 and 1 which measures the level of anisotropy, and therefore the quantity of white matter, at a particular location. A total of 376 patients are considered, with each tract measured at 93 equally spaced locations. The arguments t_{ij} thus denote a spatial rather than temporal location. The resulting observed curves are shown in figure below.

```
> library(refund)
> data(DTI)
> Y<-t(DTI$cca[-c(125,126,130,131,319,321),])  # removal of missing values and transpose of the data
> t<-seq(0,1,length=dim(Y)[1])
> Y[1:5,1:10]
         1001_1    1002_1    1003_1    1004_1    1005_1    1006_1    1007_1    1008_1    1009_1    1010_1
cca_1  0.4909345 0.4721627 0.5023738 0.4021894 0.4018747 0.4507296 0.5537167 0.4477326 0.4953400 0.4580893
cca_2  0.5168018 0.4868219 0.5136516 0.4225127 0.4055805 0.4535052 0.5577723 0.4807621 0.5057313 0.4705768
cca_3  0.5356539 0.5022577 0.5392542 0.4398983 0.3985487 0.4575306 0.5604173 0.5024963 0.5176728 0.4796741
cca_4  0.5553587 0.5233635 0.5742101 0.4600235 0.3860009 0.4655026 0.5710832 0.5181574 0.5409608 0.4886211
cca_5  0.5927610 0.5524401 0.6031339 0.4751297 0.4088248 0.4804154 0.5848086 0.5304736 0.5834464 0.5056029
>matplot(t,Y, type="n")
>matlines(t,Y, col="red")
```

(a) Convert the Y dataset to functional objects $y_i(t)$.

(b) Consider the explanatory variables $X_1 = \mathsf{case}$ and $X_2 = \mathsf{sex}$ defined as following

```
> X1<-DTI$case[-c(125,126,130,131,319,321)]
> X2<-as.numeric(DTI$sex[-c(125,126,130,131,319,321)]=="female")  # denoting females
```

Consider then the following functional linear models

$$\mathcal{M}: \; y_i(t) = \beta_0(t) + \beta_1(t)x_{i1} + \beta_2(t)x_{i2} + \varepsilon_i(t).$$

Calculate estimate for the functional $y_{i_*}(t)$, when the explanatory variables are set to the levels

$$x_{i_*1} = 1, \qquad x_{i_*2} = 1.$$

Example 5.5.

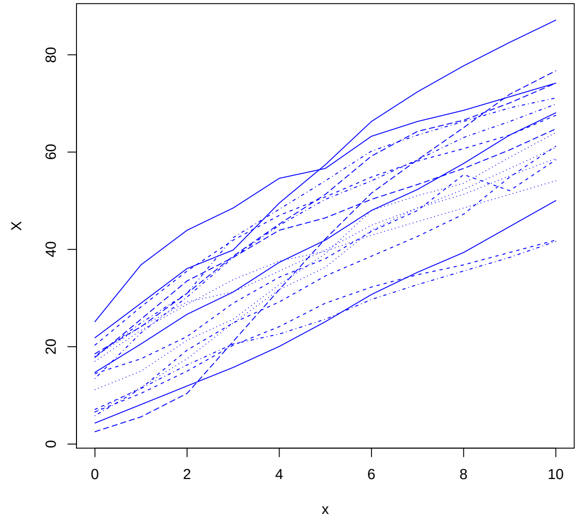
Consider the following dataset files stageheight.txt and stagediameter.txt which are including height and diameter values for 18 trees measured for each tree in ten year interval.

```
> X<-as.matrix(read.table("stagediameter.txt", sep="\t", header=TRUE))
> Y<-as.matrix(read.table("stageheight.txt", sep="\t", header=TRUE))
> t<-0:10
> head(X)
   tree3 tree11 tree17 tree28 tree30 tree31 tree37 tree44 tree52 tree57 tree59 tree61 tree62 tree65 tree72 tree77 tree84 tree85
1 25.146  6.604 18.034 18.034 17.780 21.844  7.112 11.176  5.842 18.542 14.732 20.320 17.018 13.462  2.540  4.318 14.478  6.604
2 36.830 10.414 24.130 24.892 25.654 28.956 11.430 14.986 11.430 24.130 20.574 28.194 23.368 22.860  5.588  8.128 17.526 11.684
3 43.942 14.986 29.210 30.988 33.528 36.068 19.304 21.336 16.256 30.988 26.670 35.560 28.702 30.226 10.414 11.938 22.098 17.526
> head(Y)
     tree3   tree11   tree17   tree28   tree30   tree31   tree37   tree44   tree52   tree57   tree59   tree61   tree62   tree65   tree72   tr
1 17.37360  5.51688 14.38656 16.67256 16.00200 27.70632  6.24840  8.22960  4.20624 16.30680 16.03248 13.68552 15.48384  7.95528  2.07264  4.0
2 22.76856  9.08304 19.23288 23.71344 22.06752 34.01568 10.18032 10.39368  7.25424 21.54936 22.92096 19.87296 21.70176 12.49680  5.12064  8.5
3 26.67000 13.83792 21.91512 27.67584 26.88336 38.98392 17.46504 13.59408 11.06424 26.06040 27.76728 25.66416 24.56688 21.54936  8.93064 14.1
> matplot(t,X,type="n")
> matlines(t,X,col="blue")
> matplot(t,Y,type="n")
> matlines(t,Y,col="red")

The data are internal stem measures from 18 trees. The trees were selected as having been dominant throughout their
lives with no visible evidence of damage or forks. The trees came from stands throughout the inland range of the species.

X=dbhib - Diameter (cm.) at 1.37 m (4'6?) inside bark.
Y=height - Height of tree (m)
```

(a) Convert the X and Y datasets to functional objects $x_i(t) = \boldsymbol{\phi}'\mathbf{c}_{i_X}$ and $y_i(t) = \boldsymbol{\phi}'\mathbf{c}_{i_Y}$. Decide yourself the appropriate basis system $\boldsymbol{\phi}$, and estimate coefficient vectors $\mathbf{c}_{i_X}$ and $\mathbf{c}_{i_Y}$ either by the ordinary least squares method or by the penalized least squares method.



(b) Consider the linear model

$$y_i(t) = \beta_0(t) + \beta_1(t)x_i(t) + u_i(t).$$

Calculate estimate for the functional $y_{i_*}(t)$, when $\mathbf{x}_{i*}$ is

```
        t
 [1,]  10 25.146
 [2,]  20 36.830
 [3,]  30 43.942
 [4,]  40 48.514
 [5,]  50 54.610
 [6,]  60 56.642
 [7,]  70 63.246
 [8,]  80 66.294
 [9,]  90 68.580
[10,] 100 71.374
[11,] 110 74.168
```