FDA Homework 1

Seokjun Choi

September 30, 2019

1 Chapter 1

1.1 Problem 1

The pinch is a dataset included in the fda package. It consists of 151 measurements of pinch force for 20 replications (curves).

(a) Convert the pinch data to functional objects using 15 B-splines of order four (cubic splines) and plot the 20 smoothed curves on one graph.

To get the result we want, call the fda package and execute the code below.

#R code

```
b_spline_basis<-create.bspline.basis(c(1,151), nbasis=15)
pinch.F<-Data2fd(1:151, pinch, b_spline_basis)
plot(pinch.F)
```

Then we get the result.

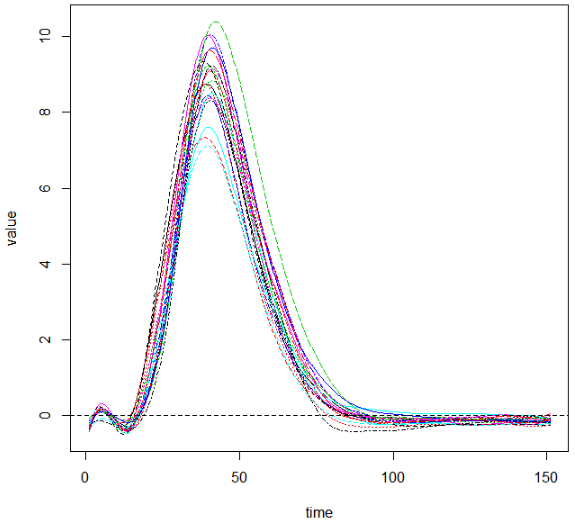

Figure 1: fitting pinch data using 15 b-splines

(b) Calculate the pointwise mean and SD and add them to the plot.

For calculation, run the code below.

```
#R code

pinch.F.mean <- mean(pinch.F)
pinch.F.mean$coefs

pinch.F.std <- std.fd(pinch.F)
pinch.F.std$coefs

par(mar=c(4,4,1,1))
plot(pinch.F, col="grey")
plot(pinch.F.mean,lwd=4,add=TRUE)
plot(pinch.F.std, lwd=4, add=TRUE, col="red")
```

For analytic expression of pointwise mean and point sd, denote b_i as i-th b-spline of order 4 on domain $[0, 120] \subset \Re$. Then

$$Mean(x) = \sum_{i=1}^{16} c_i b_i(x)$$

$$Sd(x) = \sum_{i=1}^{16} s_i b_i(x)$$

where c_i : -0.31109576, 0.73142704, -1.63349675, 2.72475346, 10.93667264, 6.03218354, 2.28791198, 0.34957598, -0.09607592, -0.16637125, -0.11240041, -0.15664938, -0.10293668, -0.16311715, -0.11897440 and s_i : 0.0846463, 0.1182031, -0.0996403, 0.9735739, 0.6517423, 1.1140145, 0.5002106, 0.3255844, 0.1166337, 0.1232983, 0.0639047, 0.0869009, 0.0824064 0.0776430, 0.07207136 respectively.

And here is a plot of the result.

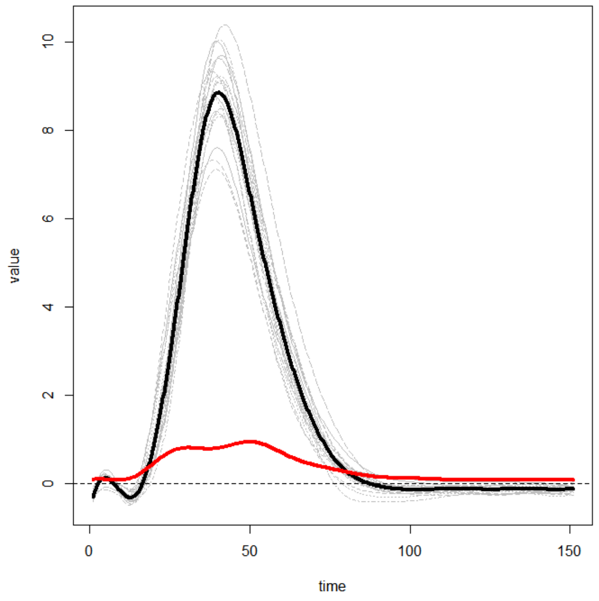

Figure 2: fitting pinch data using 15 b-splines:
Mean is black bold curve, and Sd is red bold curve.

(c) Graph the perspective and contour plots of the sample covariance function $\hat{c}(t,s)$ of the pinch curves.

We can easily do them by using for var.fd function of fda package and persp, contour function of R.

```
#R code

pinch.F.cov <- var.fd(pinch.F)
dim(pinch.F.cov$coef) #15*15
grid <- 1:15
pinch.F.cov.mat = eval.bifd(grid, grid, pinch.F.cov)
par(mfrow=c(1,2), mar=c(4,4,1,1))
persp(grid, grid, pinch.F.cov.mat, xlab="s", ylab="t", zlab="cov(s,t)")
contour(grid, grid, pinch.F.cov.mat)
```

Result plots are here.

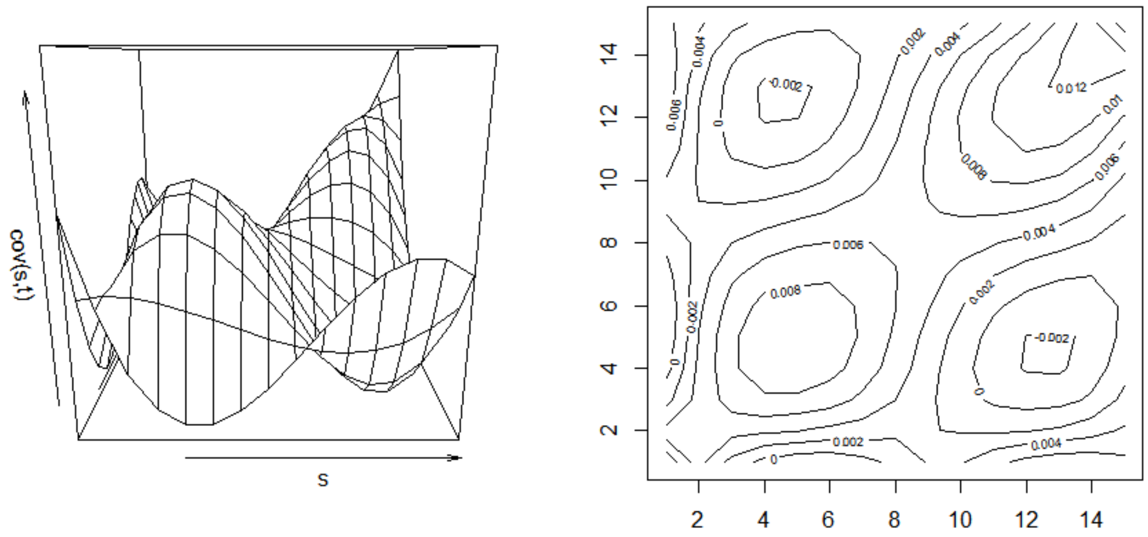

Figure 3: fitting pinch data using 15 b-splines:
covariance perspective plot and contour plot.

(d) Graph the first four EFPC's of the pinch data. How many components do you need to explain 90% of variation?

Let's do PCA with 4 components following the instruction of problem, and see the proportions of variance of each component for determining number of components for 90% explanation.

#R code

```
par(mfrow=c(1,1))
pinch.F.pca = pca.fd(pinch.F, nharm=4)
plot(pinch.F.pca$harmonics, lwd=3)
pinch.F.pca$varprop
```

Firstly, Plot the first 4 eigenfunctions.

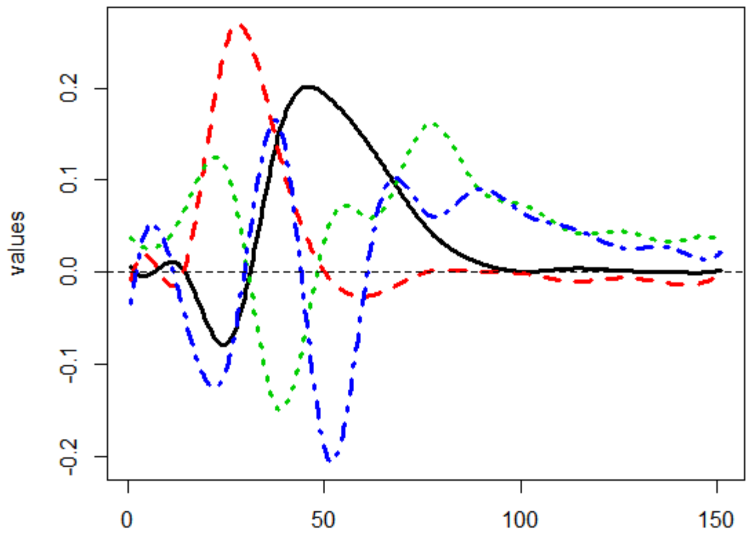

Figure 4: fitting pinch data using 15 b-splines and do PCA:
first 4 components.

And the ouput of last line is

#Console

```
> pinch.F.pca$varprop
[1] 0.67225632 0.24845297 0.04603548 0.01933904
```

Since the sum of first 2 proportion is 0.9207093, we need only 2 components to reach 90% level of explanation of variation.

1.2 Problem 2

For this problem, download the R package fds and use the data set of the United States Federal Reserve interest rates, FedYieldcurve, which contains the monthly interest rates from January 1982 to June 2009. The x-values are the maturity terms of 3, 6, 12, 60, 84 and 120 months which can be identified with the t_j in this chapter. The y-values are the interest rate of the United States Treasury obligations due in x months which can be identified with the $x_n(t_j)$, where n is a month in the range January 1982 to June 2009.

Firstly Note that, in R code, variable names are set by

```
#R code

yield = FedYieldcurve
terms = yield$x
```

FedYieldcurve is in fds package.

(a) On one graph, plot the interest rates $x(t_j)$ for January 1982 and for June 2009 against the maturity terms t_j. How do the interest rates in these two months compare?

Run the code below and get a plot before discuss about comparing yield rates.

```
#R code

plot(terms, yield$y[,1], pch=15, ylab="Yield", ylim=c(0,16))
points(terms, yield$y[,330], pch=16)
```

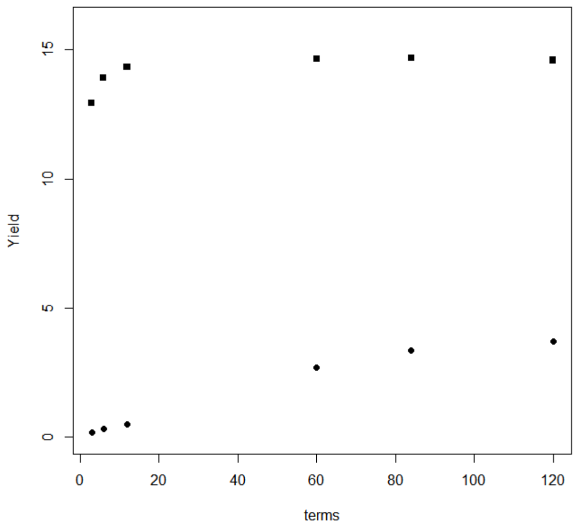

Figure 5: yield points of Jan 1982 and June 2009

At the maturity of the data exist, we can compare yield rate directly by calculating absolute value of difference. But for other maturity value, we should use more than data themself. We may use interpolation technique, or other fitting method like LSE or time series analysis's, but here are FDA class, so use method of FDA's! :)

(b) Convert the yield data to functional objects using bspline basis with four basis functions. Calculate and plot the the mean yield function. What is the average behavior of interest rates as a function of the maturity?

Execute the code below for following the first and second directions of problem.

```
#R code

b_spline_basis = create.bspline.basis(c(1,120), nbasis=4)
yield.F = Data2fd(terms, yield$y, b_spline_basis)
plot(yield.F, col='gray')

yield.F.mean = mean(yield.F)
plot(yield.F.mean,lwd=4,add=TRUE)
```

The result plot is, The mean curve has increasing shape generally, but the slope goes flatter with maturity

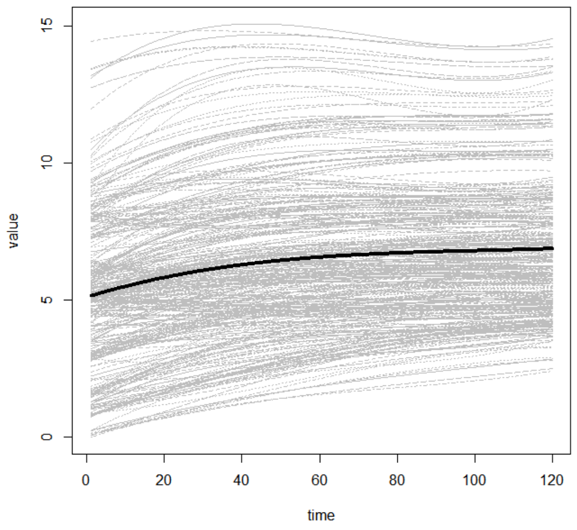

Figure 6: Fitting yield data using b-spline 4 basis:
Bold curve indicates mean.

becoming longer.

(c) Plot the first principal component of the interest rate curves. What percentage of variance does this component explain? Interpret the plot and the percentage of variance.

Run the below code to perform PCA with one component.

```
#R code

yield.F.pca = pca.fd(yield.F, nharm=1)
plot(yield.F.pca$harmonics, lwd=3, , ylim=c(-1,1))
yield.F.pca$varprop
```

The second line gives the plot of the component function.

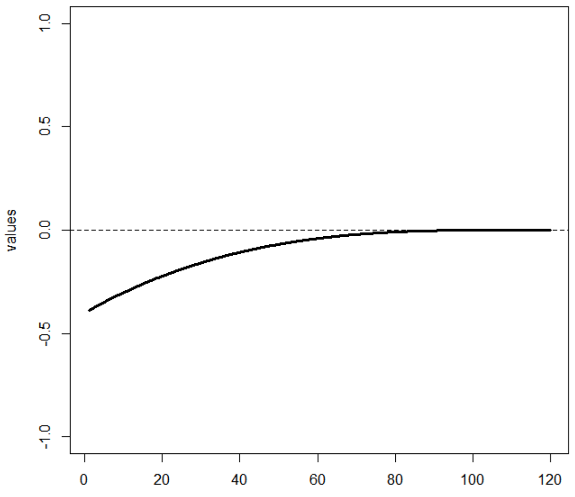

Figure 7: Fitting yield data using b-spline 4 basis:
First component of pca.

Since overall functions has similar form as we view at Figure 6, the shape of the component is also similar to them, especially mean.

```
#Console

> yield.F.pca$varprop
[1] 0.999982
```

Likewise, the proportion of explained variation has high value consisting the above context.

1.3 Problem 6

In this section we expressed $x_n(t) = \sum_{m=1}^{M} c_{nm} B_m(t)$. One can show that

$$\bar{x}_N(t) = \sum_{m=1}^{M} a_m B_m(t)$$

and

$$\hat{c}(t,s) = \sum_{m=1}^{M} \sum_{k=1}^{M} b_{mk} B_m(t) B_k(s)$$

Find expressions for the coefficients a_m and b_{mk} assuming that the $B_m(t)$ are an orthonormal basis:

$$\int B_m(t) B_k(t) = 1_{m=k}$$

Happily, the solution of this problem is in our lecture note, so I repeat the contents. Firstly for sample mean, observe that

$$\bar{x}_N(t) = \frac{1}{N} \sum_{n=1}^{N} x_n = \frac{1}{N} \sum_{n=1}^{N} \sum_{m=1}^{M} c_{nm} B_m(t) = \sum_{m=1}^{M} (\frac{1}{N} \sum_{n=1}^{N} c_{nm}) B_m(t)$$

so if we set $a_m = \frac{1}{N} \sum_{n=1}^{N} c_{nm}$, we get the expression that we want. And for sample covariance,

$$\hat{c}(t,s) = \frac{1}{N-1} \sum_{n=1}^{N} (x_n(t) - \bar{x}(t))(x_n(s) - \bar{x}(s))$$

then where $\tilde{c}_{nm} = c_{nm} - a_m$,

$$= \frac{1}{N-1} \sum_{n=1}^{N} \sum_{m=1}^{M} \sum_{k=1}^{M} \tilde{c}_{nm} \tilde{c}_{nk} B_m(t) B_k(s)$$

$$= \sum_{m=1}^{M} \sum_{k=1}^{M} (\frac{1}{N-1} \sum_{n=1}^{N} \tilde{c}_{nm} \tilde{c}_{nk}) B_m(t) B_k(s)$$

so if we set $b_{mk} = \frac{1}{N-1} \sum_{n=1}^{N} \tilde{c}_{nm} \tilde{c}_{nk}$, or equivalently using matrix notation $b_{mk} = (N-1)^{-1} \mathbf{c}^T \mathbf{c}$, we get what we want.

2 Chapter 2

2.1 Problem 1

Verify the equality,

$$\int_0^T [L(x)(t)]^2 dt = \pi\omega^5 \sum_{j=2}^{J} j^2(j^2-1)^2(a_j^2+b_j^2)$$

with harmonic acceleration operator $L(X)(t) = \omega^2 x^{(1)}(t) + x^{(3)}(t), \omega = 2\pi/T$
and Fourier basis $x_J(t) = c_0 + \sum_{j=1}^{J}\{a_j \sin\omega jt + b_j\cos\omega jt\}$

For a start, I'll calculate $x^{(i)}(t), i = 1, 3$. For notational simplicity, I omit subscript J. The results are

$$x^{(1)}(t) = \sum_j \{a_j\omega j\cos\omega jt - b_j\omega j\sin\omega jt\}$$

$$x^{(3)}(t) = \sum_j \{-a_j\omega^3 j^3\cos\omega jt + b_j\omega^3 j^3\sin\omega jt\}$$

I notice that when j=1, L become 0 operator. So we can exclude to j=1 case in summation.
Then, our operator are actually

$$L(x)(t) = \sum_j \{a_j\omega^3(j-j^3)\cos wjt + i\frac{1}{i}b_j\omega^3(j^3-j)\sin(wjt)\}$$

Then, using definition of complex sine and cosine function, we can rewrite L as

$$L(x)(t) = \sum_j \sqrt{\frac{(a_j+b_j)^2\omega^6 j^2(j^2-1)^2}{2}}e^{i\omega jt}$$

Then, above form can be seen as formal Fourier expansion form (on a circle $[0,T]$). So, by Parseval's identity,

$$\frac{\omega}{2\pi}\int_0^T |L(x)(t)|^2 dt = \frac{1}{2}\sum_j (a_j+b_j)^2\omega^6 j^2(j^2-1)^2$$

Multiplying $\frac{2\pi}{\omega}$ to both term, we get the equation which we want.

2.2 Problem 2

Proceed with FedYieldcurve data, which contains the monthly Federal Reserve interest rates, cf. Problem 1.6.2.

Likewise at problem 2 of chapter 1, firstly I set the variables like this of R code.

```
#R code

yield.J1982 = FedYieldcurve$y[,1]
terms = FedYieldcurve$x
```

(a) Smooth the interest rates (yields) in January 1982 using a B–spline basis with four basis functions. Plot the raw and smoothed interest rates on one graph.

FedYieldcurve data are monthly, so we proceed with only one curve. To get the plot of raw and fitted curves with 4 b-splines, run this code.

```
#R code

b_spline_basis4 = create.bspline.basis(c(1,120), nbasis=4)
yield.F4 = Data2fd(terms, yield.J1982, b_spline_basis4)
plot(yield.F4, col='black', xlim=c(0,120), ylim=c(12.5,15.5),
    xlab="maturity", ylab="yield rate")
points(terms, yield.J1982, type='l', col='gray')
```

The result is here.

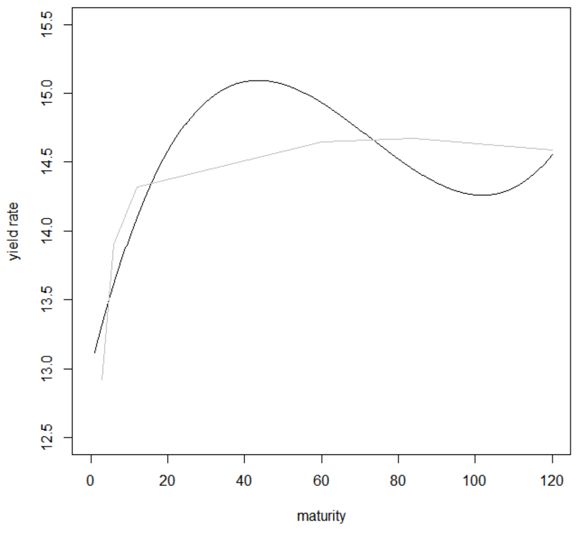

Figure 8: January 1982 yield:
gray line is raw data, and black curve is fitted curve using 4 b-splines.

(b) Re-fit the January 1982 yields using a penalized smoothing based on six basis functions (as many as data points) with the smoothing parameter $\lambda = 1$, and the second derivative as the penalty operator. Add the smooth in red to the graph you obtained in part (a) and comment on the result.

By modifying just parameter of above code and adding smoothing code, we can get the results which we want.

```
#R code

b_spline_basis6 = create.bspline.basis(c(1,120), nbasis=6)
par.F6.S1 = fdPar(b_spline_basis6, Lfdobj=2, lambda=1)
yield.F6.S1 = smooth.basis(terms, yield.J1982, par.F6.S1)

par(new=T)
plot(yield.F6.S1, xlim=c(0,120), ylim=c(12.5,15.5), ylab="", xlab="", col='red')
yield.F6.S1$gcv
```

Then we get below plot.

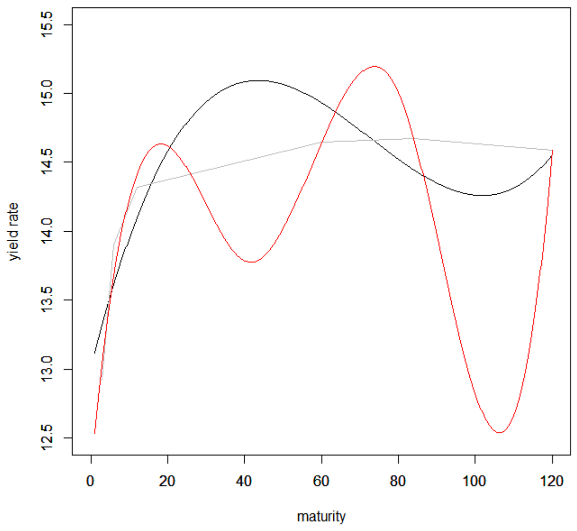

Figure 9: January 1982 yield:
gray: raw data, and black : fitted curve using 4 b-splines, red : fitted curve using 6 b-splines.

Since the number of basis increases and we are using polynomial basis, the fitted curve with 6-basis has more variation, and consquently it seems to get worse comparing to (a)'s 4 basis cases to explain original (raw) line. This phenomenon is similar to the outcome of ordinary overfitting cases raised by too high order polynomial model for fitting to small (point) dataset, and at our functional data case where the order consists the number of basis, corresponds to our intuition. (Note that although here 'is' the term "order" which means other things that i.e. the (polynomial) order of each b-spline, so statement like above may make some confusing, but I think an increasing of 'the order' of each b-splines also gives similar conclusion, thus 'intuitively' the statement still has some consistence of too-high-basis-number case like above.)

(c) Repeat part (b) with several other smoothing parameters λ. Which λ gives the most informative smooth curve?

For finding enoughly best λ value to satisfiablely most(?) informative smooth curve, I'll consider candidates of λ value which are integers from 1 to 200. (For 'the most', we should explore more deliberate floting number, but I'll skip this for simplicity.) And as a criterion, I'll calculate GCVs - a kind of validation measure for fitting - for each λ cases, and choose the λ value which gives the lowest GCV value.
run the below code.

```
#R code

candid_lambda = seq(1, 200, 1)
gcvs = rep(0, length(candid_lambda))
for(i in 1:length(candid_lambda)){
    par.F6.Si = fdPar(b_spline_basis6, Lfdobj=2, lambda=candid_lambda[i])
    yield.F6.Si = smooth.basis(terms, yield.J1982, par.F6.Si)
    gcvs[i]=mean(yield.F6.Si$gcv)
}
plot(candid_lambda, gcvs)
which.min(gcvs)
```

The second line from the end gives plot of GCV values for each λ value, and the output of last line give best λ value to us.

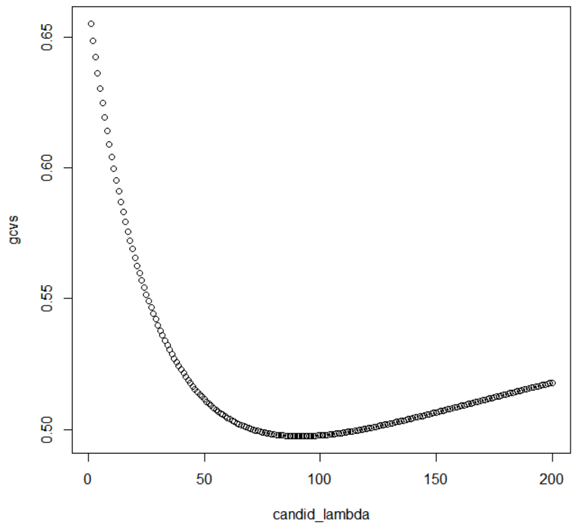

Figure 10: January 1982 yield: GCVs for each λ:
smoothed by penalized smoothing method with 2nd derivative operator, fitted using 6 b-splines.

And minimum GCV value is at $\lambda = 91$. So, I'll repeat smoothing and fitting procedure with the λ value, and get a fitted curve. And additionally, I'll plot second derivative curves for each.
Run this code.

```
#R code

#take lambda=91
par.F6.S91 = fdPar(b_spline_basis6, Lfdobj=2, lambda=91)
yield.F6.S91 = smooth.basis(terms, yield.J1982, par.F6.S91)
```

```
plot(yield.F4, col='black', xlim=c(0,120), ylim=c(12.5,15.5))
points(terms, yield.J1982, type='l', col='gray')
par(new=T)
plot(yield.F6.S1, xlim=c(0,120), ylim=c(12.5,15.5), ylab="", xlab="", col='red')
par(new=T)
plot(yield.F6.S91, xlim=c(0,120), ylim=c(12.5,15.5), ylab="", xlab="", col='blue')

#for 2nd derivative
plot(deriv.fd(yield.F4), col='black', ylim=c(-0.1,0.4))
par(new=T)
plot(deriv(yield.F6.S1$fd), col='red', ylim=c(-0.1,0.4))
par(new=T)
plot(deriv(yield.F6.S91$fd), col='blue', ylim=c(-0.1,0.4))
```

Then, we get below plots.

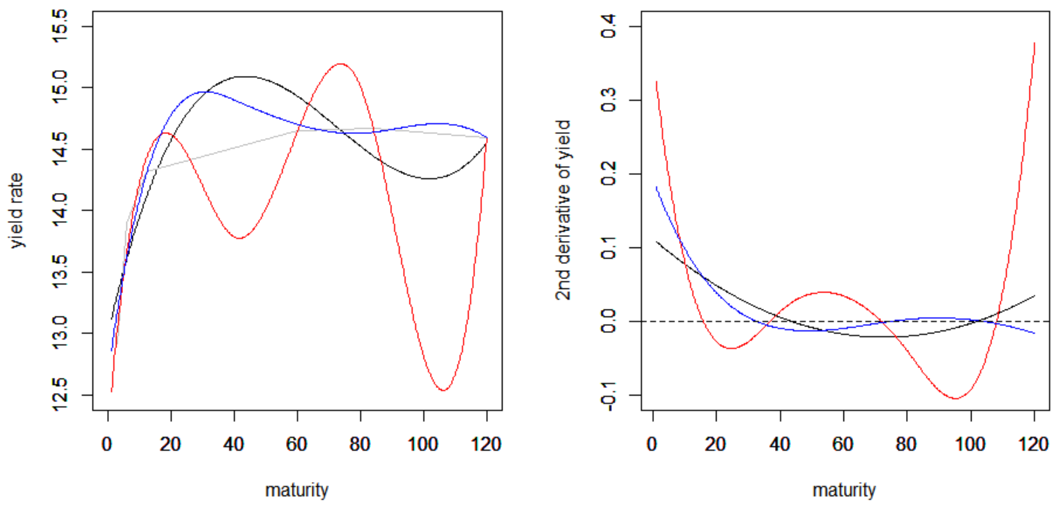

Figure 11: January 1982 yield:
left: yield rate, right: 2nd derivative of yield rate
gray: original data (caution: no derivative curve)
black: fitted using 4 b-spline basis
red: fitted using 6 b-spline basis and smoothed with $\lambda = 1$
blue: fitted using 6 b-spline basis and smoothed with $\lambda = 91$

On the plots, we can verify the effect of smoothing with chosen $\lambda = 91$ value by watching blue curve. On whole of maturity interval $[0, 120]$, the blue curve of right plot has 2nd derivative value closer to 0 comparing to red curve, and also blue curve of left plot is better to explain original rate comparing to red one. And Generally, on most part of domain interval, blue curve shows good performance as black curve which is fitted with 3 b-spline basis.

So, consquently, it is important to choose proper number of basis and smoothing parameter for explaining a given data well.

2.3 Problem 5

In this problem, we will explore some potential dangers of curve alignment. A bump function centered at a value c_0, with radius r_0, and amplitude a_0 is defined as

$$f(x) = \begin{cases} a_0 \exp\{-(1 - (\frac{x-c_0}{r_0}^2)^{-1})\} & \text{if } |x - c_0| < r_0 \\ 0 & \text{otherwise} \end{cases}$$

(a) Simulate a functional sample over the unit interval each with a sample size of 50 from the Matérn process. For the first half of the sample, set the mean function equal to the bump function with parameters $(c_0, r_0, a_0) = (3/8, 1/4, 5)$. For the second half use $(c_0, r_0, a_0) = (5/8, 1/4, 5)$. You may choose the values for the Matérn covariance function as well as the number of points sampled per curve. Plot all of the curves and include a curve for the overall mean function.

Following direction of problem, run this code to generate Matérn process and plot results. For parameters of covariance of Matérn process, I choose $\sigma^2 = 1, \rho = 0.1, \nu = 0.5$ respectively, and for number of points, I take 100 on $[0, 1)$ (see 'times' variable on the code).

#R code

```
library(fields)
library(expm)
library(fda)

num.sample = 50

#define bump function
bump = function(x, c0, r0, a0){
    result = rep(0, length(x))
    for(i in 1:length(x)){
        if(abs(x[i] - c0) < r0){
            result[i] = a0*exp(-(1-((x[i]-c0)/r0)^2)^-1)
        }
    }
    return(result)
}

times = seq(0,0.99,0.01)
bumpval1 = bump(times, 3/8, 1/4, 5)
bumpval2 = bump(times, 5/8, 1/4, 5)
# plot(times, bumpval2)

#generate Matern process
mt.sig2 = 1 #pointwise variance
mt.rho = 0.1 #range parameter (how quickly falls off)
mt.nu = 1/2 #smoothness parameter

d_mat = abs(outer(times,times,"-"))
C_1 = apply(d_mat, c(1,2), FUN=Matern, range=mt.rho, nu=mt.nu)
C_1 = C_1*mt.sig2
C_1_sq = sqrtm(C_1)


MaternSamples = matrix(0, length(times), num.sample)
for(i in 1:num.sample){
    Z<-rnorm(length(times))
    MaternSamples[,i] = C_1_sq%*%Z
}

MtSample.group1 = MaternSamples[, 1:(num.sample/2)]
MtSample.group2 = MaternSamples[, (num.sample/2+1):num.sample]


# mean adjust
for(i in 1:(num.sample/2)){
    MtSample.group1[,i] = MtSample.group1[,i] + bumpval1
    MtSample.group2[,i] = MtSample.group2[,i] + bumpval2
}
adjMtSamples = cbind(MtSample.group1, MtSample.group2)


#plot after adjusted by bump function
par(mfrow=c(1,3))
plot(times,MtSample.group1[,1],type="n", ylim=c(-3,4))
for(i in 1:(num.sample/2)){
    points(times, MtSample.group1[,i],type="l", col="red")
    points(times, MtSample.group2[,i], type="l", col="blue")
}
points(times, bumpval1, type="l", col="red", lwd=3)
points(times, bumpval2, type="l", col="blue", lwd=3)
points(times, rowMeans(adjMtSamples[,1:(num.sample/2)]), type="l", col="yellow", lwd=3)
points(times, rowMeans(adjMtSamples[,(num.sample/2+1):num.sample]), type="l", col="green", lwd=3)
#
plot(times, bumpval1, type="l", col="red", lwd=3, ylim=c(-3,4))
points(times, bumpval2, type="l", col="blue", lwd=3)
points(times, rowMeans(adjMtSamples[,1:(num.sample/2)]), type="l", col="yellow", lwd=3)
points(times, rowMeans(adjMtSamples[,(num.sample/2+1):num.sample]), type="l", col="green", lwd=3)
#
plot(times,MtSample.group1[,1],type="n", ylim=c(-3,4))
for(i in 1:(num.sample)){
    points(times, adjMtSamples[,i],type="l", col="grey")
}
points(times, 0.5*(bumpval1+bumpval2), type="l", col="black", lwd=3)
points(times, rowMeans(adjMtSamples), type="l", col="yellow", lwd=3)
```

Last code block gives plots for simulated process and mean of them.
Each curve on plots indicates,

- **left:** plot of whole processes with group infomation.
  red & blue color indicates the group (first 25s & last 25s) respectively.
  bold red & blue curve are bump functions of groups. (group's theorical-population mean)
  and bold yellow and green curve are realized pointwise mean of each group. (group's sample mean)
- **middle:** draw only group mean curves from left one.
- **right:** plot of whole process without group information.
  grey curves are 50 generated processes.
  bold black curve is mean of 2 bump functions. (theorical mean of total process)
  and bold yellow curve is mean of realized pointwise process. (total sample mean)

By comparing sample mean curves to theorical one, we can verify that simulated processes are properly generated to our purpose.

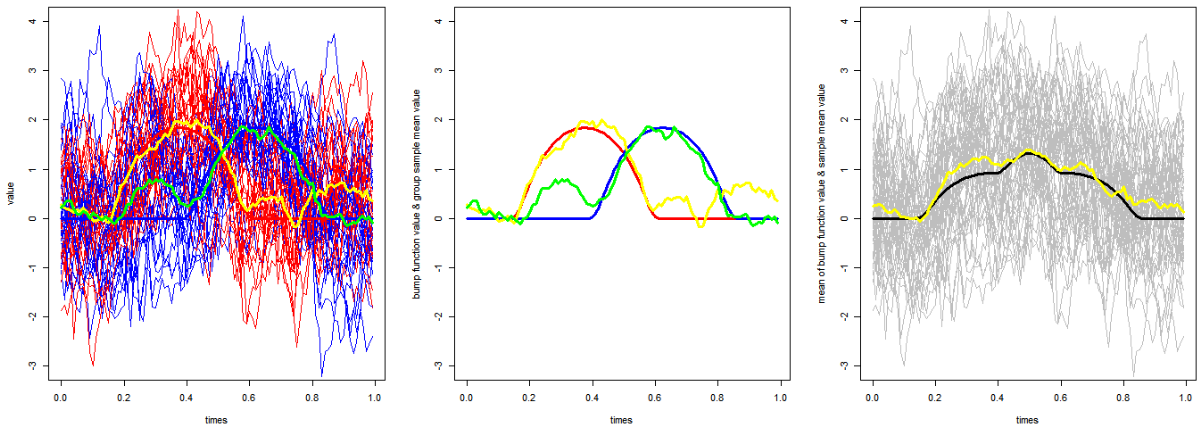

Figure 12: Matérn process

(b) Align the curves using continuous registration. Plot the resulting curves and include a mean function. Comment on any differences with (a) and if the registered curves exhibit any odd patterns.

Firstly, I do fitting process to get the functional object using 6 b-spline basis. And next, I carry out the alignment procedure using continuous registration method. Then finally, I plot the curves fitted of before after the alignment.

Run the code below.

```
#R code

# basic fitting to b-spline
b_spline_basis = create.bspline.basis(c(0,1), nbasis=6)
adjMtSample.f = smooth.basis(times, adjMtSamples, b_spline_basis)$fd

#plot: curves before alignment
par(mfrow=c(1,3))
plot(adjMtSample.f, ylim=c(-3,4), col='grey'); plot(mean(adjMtSample.f), add=T, lwd=3)

#apply continuous registration
adjMtSample.f.reg = register.fd(adjMtSample.f)

#plot: aligned curves
plot(adjMtSample.f.reg$regfd, col='grey', ylim=c(-3,4))
plot(mean(adjMtSample.f.reg$regfd), add=T, lwd=3, col='red')

#plot: compare only mean
plot(mean(adjMtSample.f), lwd=3, col='black', ylim=c(-3,4))
plot(mean(adjMtSample.f.reg$regfd), add=T, lwd=3, col='red')
```

Then, we get 3 plots,

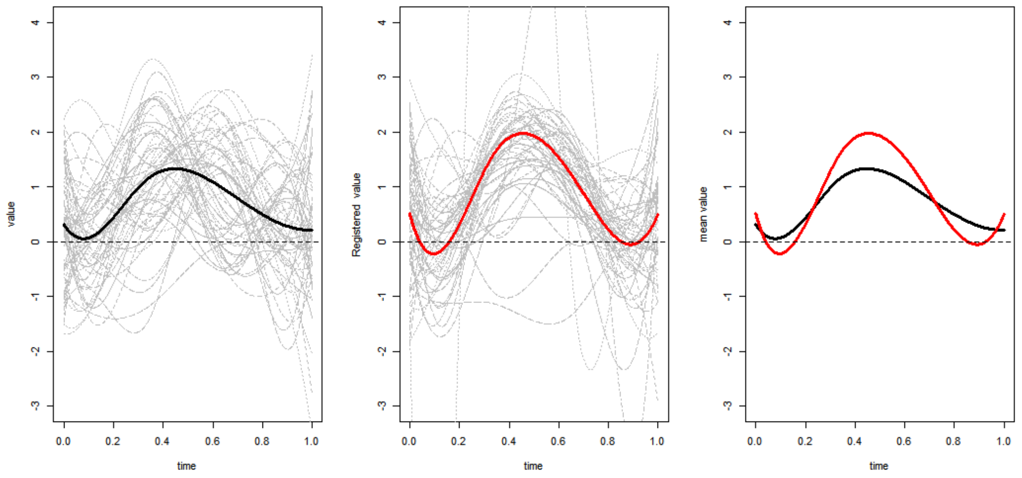

Figure 13: Matérn process: before & after alignment

Each curve on plots indicates,

- **left:** Before alignment.
  gray: Using 6 b-spline basis, get each curve corresponds to each process generated at above.
  bold black: the mean function of fitted curves before alignment.

- **middle:** After alignment.
  gray: aligned curves for fitted ones using 6 b-spline basis to each process.
  bold red: the mean function of aligned curves.

- **right:** plot only mean curves.
  bold black: the mean function of unaligned curves.
  bold red: the mean function of aligned curves.

Because we generated processes through transforming Matérn processes of same covariance structure by 2 bump function whose shapes are same except only location, the effect of the alignment is enormous. The gray curves of aligned plot are adjust to go up and down nearly same points (respectively) for most observations comparing to unaligned's gray curves, and the mean curve of aligned one is more explanatory than unaligned one, especially showing higher amplitude(function value) which replects each curve more exactly.

(c) Carry out an FPCA with one PC on the unaligned and aligned curves separately. For each, do a simple linear regression of the score onto a dummy variable (coded 0/1) indicating which type of mean the function had (i.e. is it from the first or second half of the sample). Calculate a p-value to determine if the estimated slope parameters you get are significant. Compare with the aligned and unaligned curves. What did aligning do to the p-value? You may want to rerun your simulations a few times to see how the p-values change.

Execute below code for doing pca with a 1 component and calculating p-value after least-square estimation of score onto group category variable.

```
#R code

#pca
adjMtSample.f.pca = pca.fd(adjMtSample.f, nharm=1)
adjMtSample.f.reg.pca = pca.fd(adjMtSample.f.reg$regfd, nharm=1)

#pca component plot
par(mfrow=c(1,1))
plot(adjMtSample.f.pca$harmonics, ylim=c(-3,4))
plot(adjMtSample.f.reg.pca$harmonics, col='red', add=T)
```

Firstly, we examine the results of PCA for each case. Below is plot of first component eigenfunction of each.

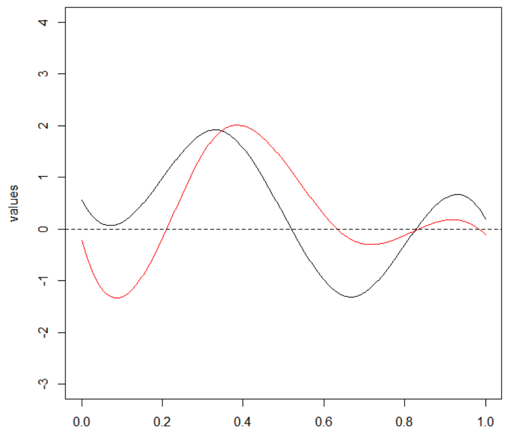

Figure 14: Matérn process: PCA before & after alignment
black : 1st component of pca of before alignment case
red : 1st component of pca of after alignment case

Observe that, by alignment, the component moves to right, direction to mean. It is effect of continuous registration, since the method aligns curves with the mean's location.

We can also get the proportion of explained variation by the first component, for each case.

```
#Console

> adjMtSample.f.pca$varprop
[1] 0.3593796
```

```
> adjMtSample.f.reg.pca$varprop
[1] 0.7315676
```

The proportion are doubled! It consists the big effect of alignment as we see when examine mean curves. (In fact, the increased variance proportion is NOT observed all the simulation cases. It depends on simulated process's general shapes (when each process path are more similar, more improvement!), so in general, if we simulate with other random-seed value or simply repeat simulation, the size of improvement does NOT reach to double through alignment, even there can be cases for decrease after alignment.)

In succession, I will do least square fit of score, so make the dummy variable, call simple regression code to each case, and plot the result and get p-value of slope following the direction of problem. Run below code.

```
#R code

dummy_var = c(rep(0,num.sample/2),rep(1,num.sample/2))
un_align_score = adjMtSample.f.pca$score
align_score = adjMtSample.f.reg.pca$score

par(mfrow=c(1,2))
lm1_unaligned= lm(un_align_score ~ dummy_var)
plot(dummy_var, un_align_score, ylim=c(-1.5, 1))
abline(lm1_unaligned$coefficient, col='red')
summary(lm1_unaligned)

lm2_aligned= lm(align_score ~ dummy_var)
plot(dummy_var, align_score, ylim=c(-1.5, 1))
abline(lm2_aligned$coefficient, col='red')
summary(lm2_aligned)
```

The results are,

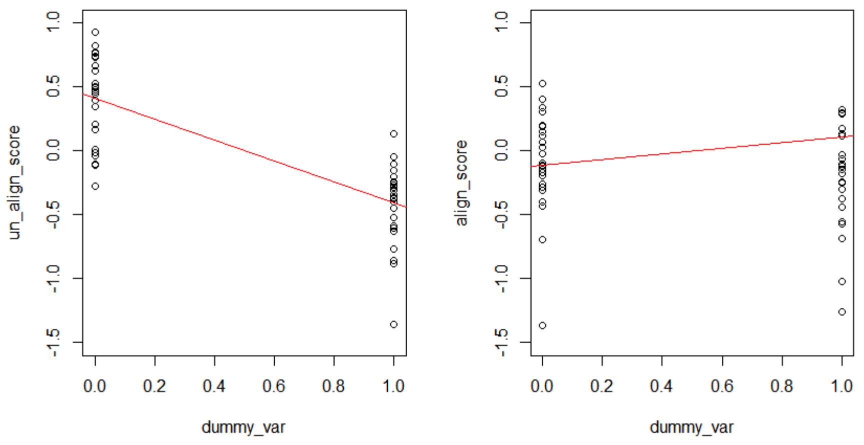

Figure 15: Matérn process: regression of score onto group dummy variable
left: before alignment, right: after alignment 0: first 25 group, 1: last 25 group

```
#Console

> summary(lm1_unaligned)
```

```
Call:
lm(formula = un_align_score ~ dummy_var)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.94443 -0.20165  0.06072  0.24753  0.54723 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  0.40993    0.06706   6.113 1.68e-07 ***
dummy_var   -0.81987    0.09483  -8.646 2.37e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3353 on 48 degrees of freedom
Multiple R-squared:  0.6089,    Adjusted R-squared:  0.6008 
F-statistic: 74.75 on 1 and 48 DF,  p-value: 2.371e-11

> summary(lm2_aligned)

Call:
lm(formula = align_score ~ dummy_var)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.3657 -0.3425 -0.1131  0.1634  7.6486 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.1077     0.2387  -0.451    0.654
dummy_var     0.2154     0.3375   0.638    0.526

Residual standard error: 1.193 on 48 degrees of freedom
Multiple R-squared:  0.008415,  Adjusted R-squared:  -0.01224 
F-statistic: 0.4074 on 1 and 48 DF,  p-value: 0.5263
```

We can observe that,

- Before alignment, the slope value of the regression result is significantly NOT 0 since the p-value of coefficient are 2.37e-11, very small value.
- After alignment, the p-value corresponds to slope coefficient is 0.526, so we can make a conclusion that the slope is statistically 0.

So, in unaligned case, the scores have different level depending on that the sample belongs to which group, but in aligned case, this phenomenon are not represented. It may be able to be explained by below intuition.

- For case of before alignment, although there were 2 types of shape of processes, we use only an 1 component. So in fitting process using the eigenfunction to each for original process (or similarly curve fitted by b-spline basis earlier), naturally the score - the scalar value multiplied to the component to make nearest fit- are 2-grouped. And the result is presented by significant slope value with low p-value in above regression.
- While, for the case of after alignment, all (or just, most) fitted curves by b-spline basis are have similar shape. So in fitting process using an eigenfunction, all multipliers of the function to make nearest fit become to have also similar values. And it makes the result that the non-significant coefficient of dummy variable with high p-value in above regression.

(d) Come up with one potential setting/application where you might lose something if you align. Make up whatever scenario you like, but think it through.

As we see above example simulation, the alignment procesure delete some information about location which curves moves specially on a step in respectively tuned clock but the shape that moves are similar for each observation. Thus, for perpose of inference, if the triggered time at which value starts to move specifically is important information, We SHOULD NOT do alignment procedure.

A toy example situration is here:
We want to know the distribution of age that boys start to grow faster in adolescence period by begining a secondary sexual characteristics. So we get some samples that forms of functional-data.

In this case, if we align the curves, we lose information about 'starting age' of most boys, and unintentionally emphasize some unnecessary information like the length of growth. Consquently, we will fail to infer properly from the data, cannot achieve the purpose - getting distribution of starting age.