HW1

2022020428 Jihun Lim

```r
knitr::opts_chunk$set(echo = TRUE)
library(fds)
```

```
## Warning:  'fds' R  4.1.3
```

```
##   : rainbow
```

```
## Warning:  'rainbow' R  4.1.3
```

```
##   : MASS
```

```
##   : pcaPP
```

```
## Warning:  'pcaPP' R  4.1.3
```

```
##   : RCurl
```

```
## Warning:  'RCurl' R  4.1.3
```

```r
library(fda)
```

```
## Warning:  'fda' R  4.1.3
```

```
##   : splines
```

```
##   : deSolve
```

```
## Warning:  'deSolve' R  4.1.3
```

```
##
##   : 'fda'
```

```
## The following object is masked from 'package:graphics':
##
##     matplot
```

```r
library(ggplot2)
```

```
## Warning:  'ggplot2' R  4.1.2
```

```r
library(fields)
```

```
## Warning:  'fields' R  4.1.3
```

```
##   : spam
```

```
## Warning:  'spam' R  4.1.3
```

```
## Spam version 2.9-1 (2022-08-07) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g. 'help( chol.spam)'.
```

```
##
##   : 'spam'
## The following objects are masked from 'package:base':
##
##     backsolve, forwardsolve

##   : viridis

## Warning:  'viridis' R  4.1.2

##   : viridisLite

## Warning:  'viridisLite' R  4.1.2

##
## Try help(fields) to get started.
```

```
library(expm)
```

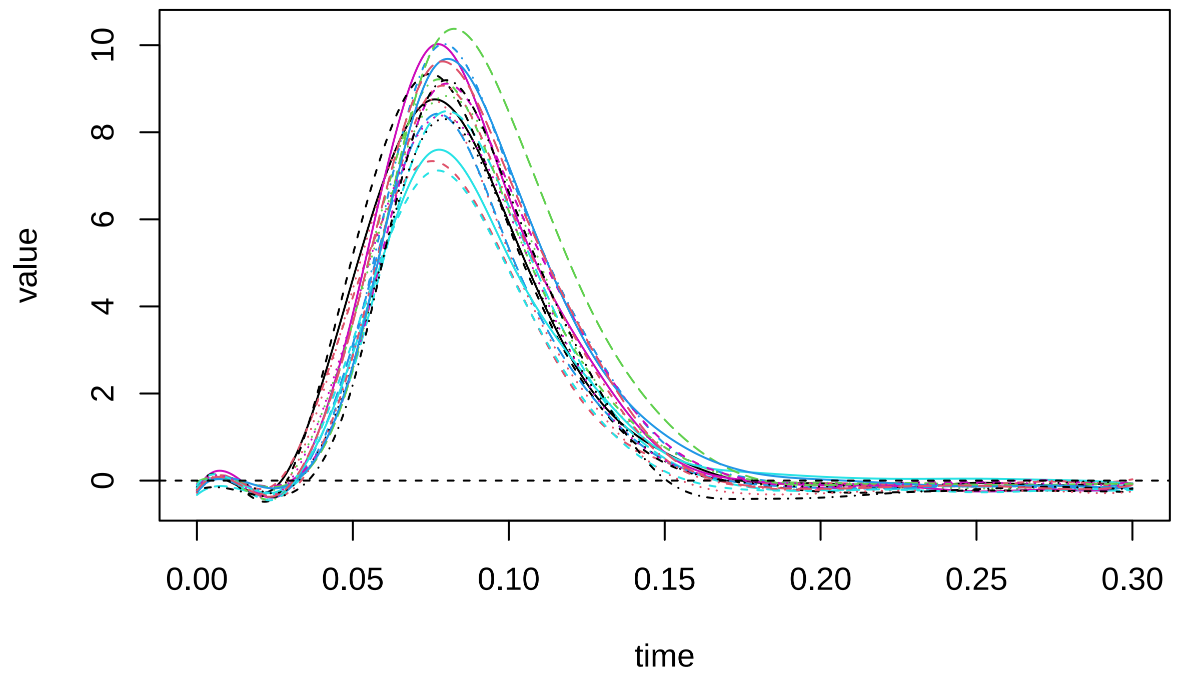
```
## Warning:  'expm' R  4.1.3

##   : Matrix

##
##   : 'Matrix'

## The following object is masked from 'package:spam':
##
##     det

##
##   : 'expm'

## The following object is masked from 'package:Matrix':
##
##     expm
```

Chapter 1

Problem 1

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.

```
time = pinchtime
basis <- create.bspline.basis(c(0,0.3),nbasis=15, norder = 4)
pinch.F<-Data2fd(time, pinch, basis)
plot(pinch.F)
```



```
## [1] "done"
```

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

```
mean.fd(pinch.F)
```

```
## $coefs
##              mean
##  [1,] -0.19640318
##  [2,]  0.53667546
##  [3,] -1.49889015
##  [4,]  2.68956848
##  [5,] 10.92286613
##  [6,]  6.04721931
##  [7,]  2.28283849
##  [8,]  0.35324569
##  [9,] -0.09900356
## [10,] -0.16429367
## [11,] -0.11493253
## [12,] -0.15313847
## [13,] -0.10891391
## [14,] -0.15580597
## [15,] -0.12331208
##
## $basis
## $call
## basisfd(type = type, rangeval = rangeval, nbasis = nbasis, params = params,
##     dropind = dropind, quadvals = quadvals, values = values,
##     basisvalues = basisvalues)
##
## $type
## [1] "bspline"
##
## $rangeval
## [1] 0.0 0.3
##
## $nbasis
## [1] 15
##
## $params
##  [1] 0.025 0.050 0.075 0.100 0.125 0.150 0.175 0.200 0.225 0.250 0.275
##
## $dropind
## NULL
##
## $quadvals
## NULL
##
## $values
## list()
##
## $basisvalues
## list()
##
## $names
##  [1] "bspl4.1"  "bspl4.2"  "bspl4.3"  "bspl4.4"  "bspl4.5"  "bspl4.6"
##  [7] "bspl4.7"  "bspl4.8"  "bspl4.9"  "bspl4.10" "bspl4.11" "bspl4.12"
## [13] "bspl4.13" "bspl4.14" "bspl4.15"
##
## attr(,"class")
## [1] "basisfd"
##
## $fdnames
## $fdnames$time
##   [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##  [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##  [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##  [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##  [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##  [91]  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108
## [109] 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
## [127] 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144
## [145] 145 146 147 148 149 150 151
##
## $fdnames$reps
## [1] "mean"
##
## $fdnames$values
## [1] "mean value"
##
```

```
##
## attr(,"class")
## [1] "fd"
```

```
std.fd(pinch.F)
```

```
## $coefs
##              [,1]
##  [1,]  0.07975281
##  [2,]  0.09865960
##  [3,] -0.08021130
##  [4,]  0.95808616
##  [5,]  0.64742807
##  [6,]  1.11728335
##  [7,]  0.49878254
##  [8,]  0.32483098
##  [9,]  0.11764947
## [10,]  0.12235584
## [11,]  0.06434126
## [12,]  0.08597867
## [13,]  0.08325722
## [14,]  0.07574510
## [15,]  0.07045293
##
## $basis
## $call
## basisfd(type = type, rangeval = rangeval, nbasis = nbasis, params = params,
##     dropind = dropind, quadvals = quadvals, values = values,
##     basisvalues = basisvalues)
##
## $type
## [1] "bspline"
##
## $rangeval
## [1] 0.0 0.3
##
## $nbasis
## [1] 15
##
## $params
##  [1] 0.025 0.050 0.075 0.100 0.125 0.150 0.175 0.200 0.225 0.250 0.275
##
## $dropind
## NULL
##
## $quadvals
## NULL
##
## $values
## list()
##
## $basisvalues
## list()
##
## $names
##  [1] "bspl4.1"  "bspl4.2"  "bspl4.3"  "bspl4.4"  "bspl4.5"  "bspl4.6"
##  [7] "bspl4.7"  "bspl4.8"  "bspl4.9"  "bspl4.10" "bspl4.11" "bspl4.12"
## [13] "bspl4.13" "bspl4.14" "bspl4.15"
##
## attr(,"class")
## [1] "basisfd"
##
## $fdnames
## $fdnames$time
##   [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##  [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##  [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##  [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##  [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##  [91]  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108
## [109] 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
## [127] 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144
## [145] 145 146 147 148 149 150 151
##
## $fdnames$`Std. Dev.`
##  [1] "rep1"  "rep2"  "rep3"  "rep4"  "rep5"  "rep6"  "rep7"  "rep8"  "rep9"
## [10] "rep10" "rep11" "rep12" "rep13" "rep14" "rep15" "rep16" "rep17" "rep18"
## [19] "rep19" "rep20"
##
## $fdnames$`Std. Dev. values`
## [1] "value"
##
##
## attr(,"class")
## [1] "fd"
```

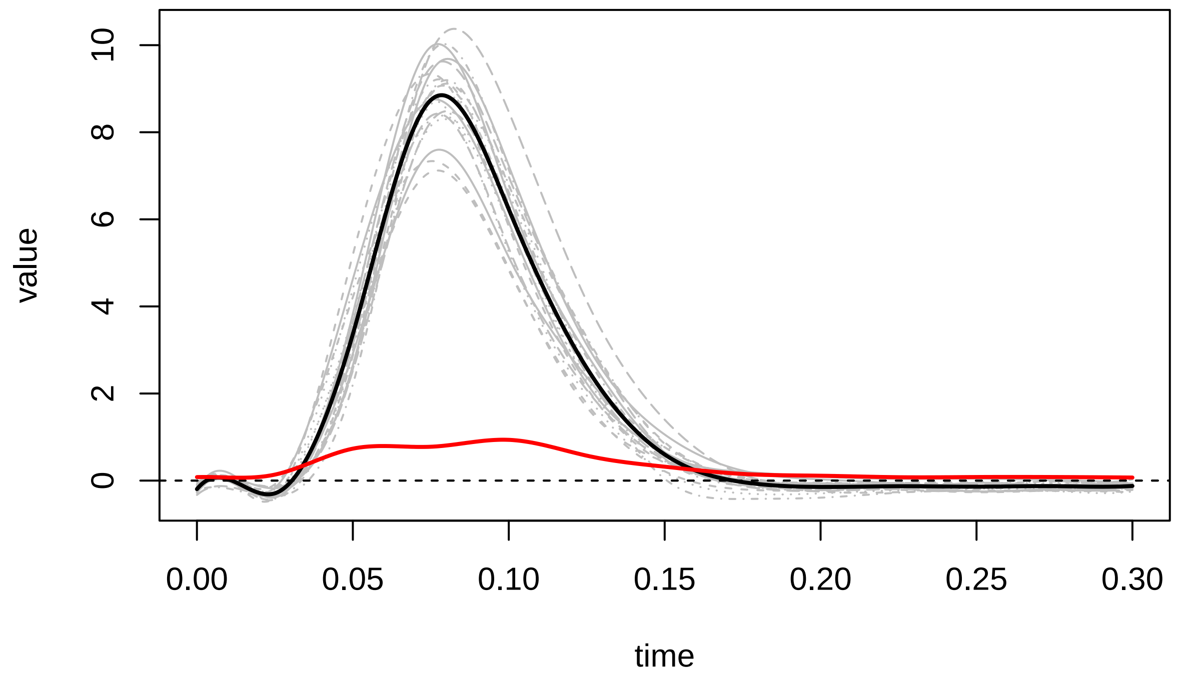
```
plot(pinch.F, col = "grey")
```

```
## [1] "done"
```

```
plot(mean.fd(pinch.F), lwd =2, add = TRUE)
```

```
## [1] "done"
```

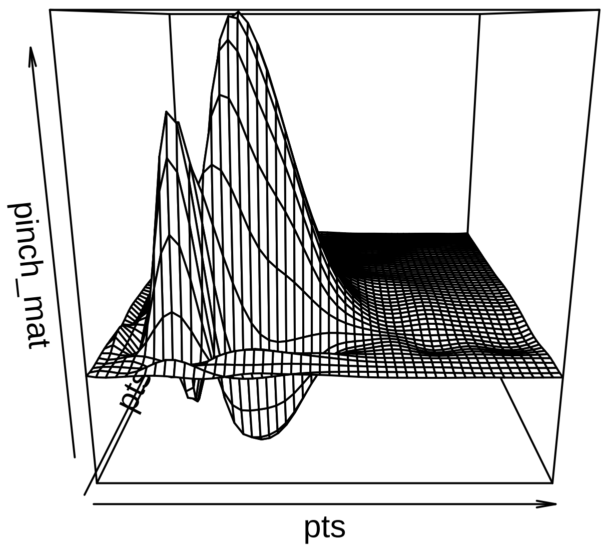
```
plot(std.fd(pinch.F), lwd =2, add = TRUE, col = "red")
```



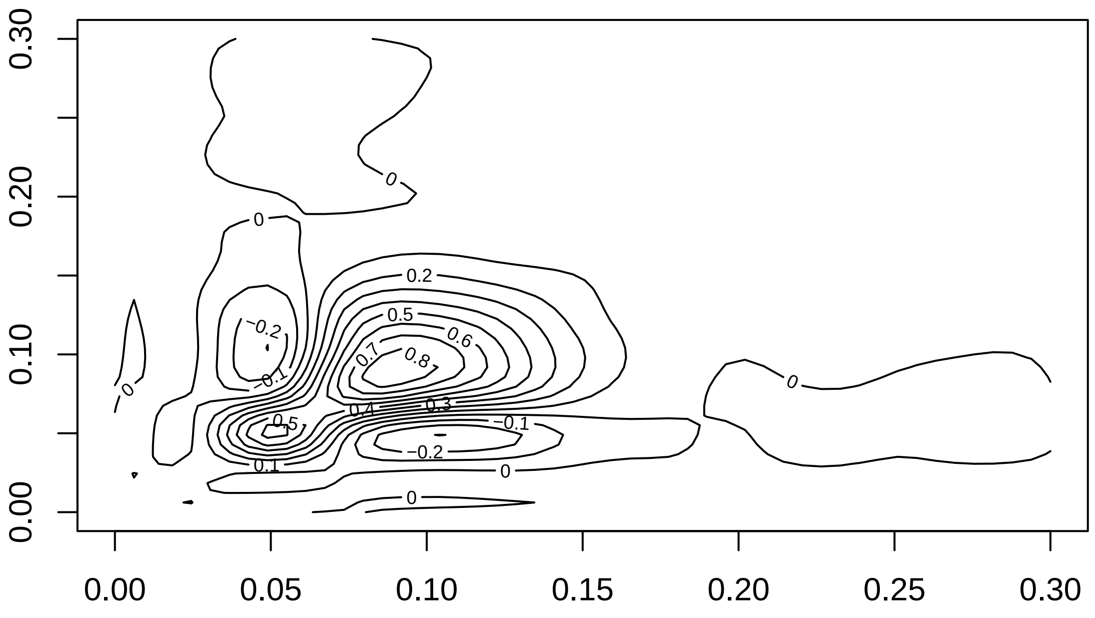
```
## [1] "done"
```

c) Graph the perspective and contour plots of the sample covariance function ^c(t, s) of the pinch curves.

```
pinch_var<-var.fd(pinch.F)
pts<-seq(from=0, to=0.3, length = 50)
pinch_mat = eval.bifd(pts, pts, pinch_var)
persp(pts, pts, pinch_mat)
```



```
contour(pts, pts, pinch_mat)
```



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

```
pinch.pca = pca.fd(pinch.F, nharm=4)
pinch.pca$varprop
```

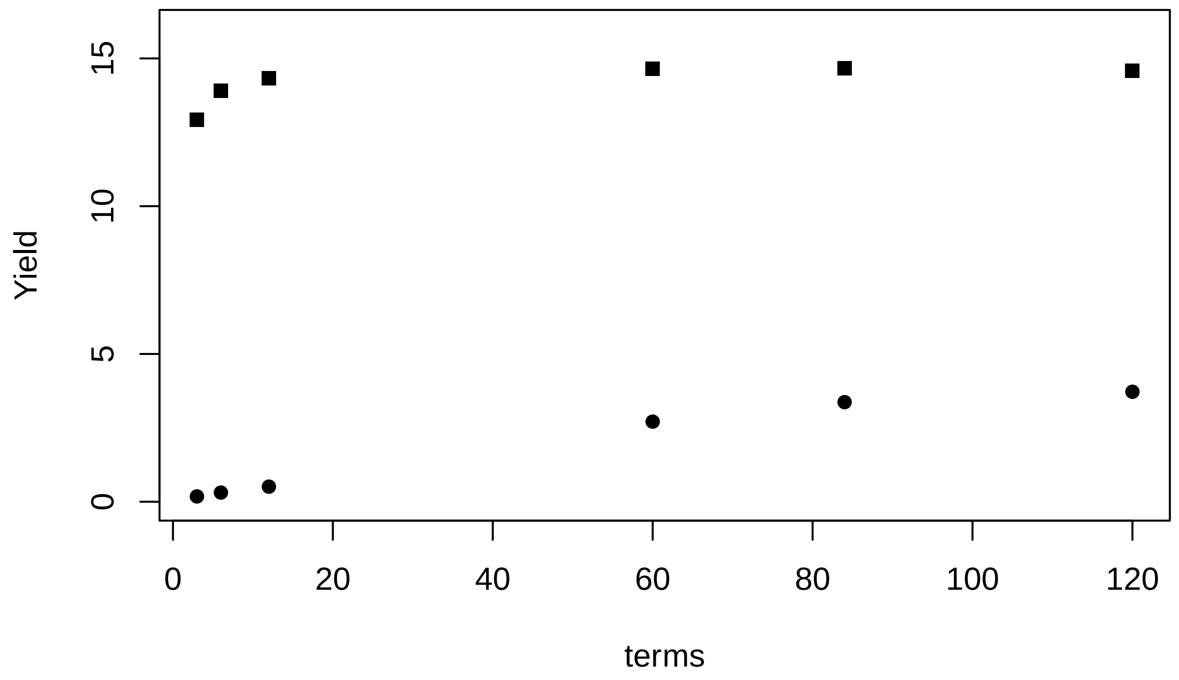
```
## [1] 0.67408784 0.24791744 0.04585583 0.01878534
```

We need 2 components to explain 90% of variation.

Problem 2

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

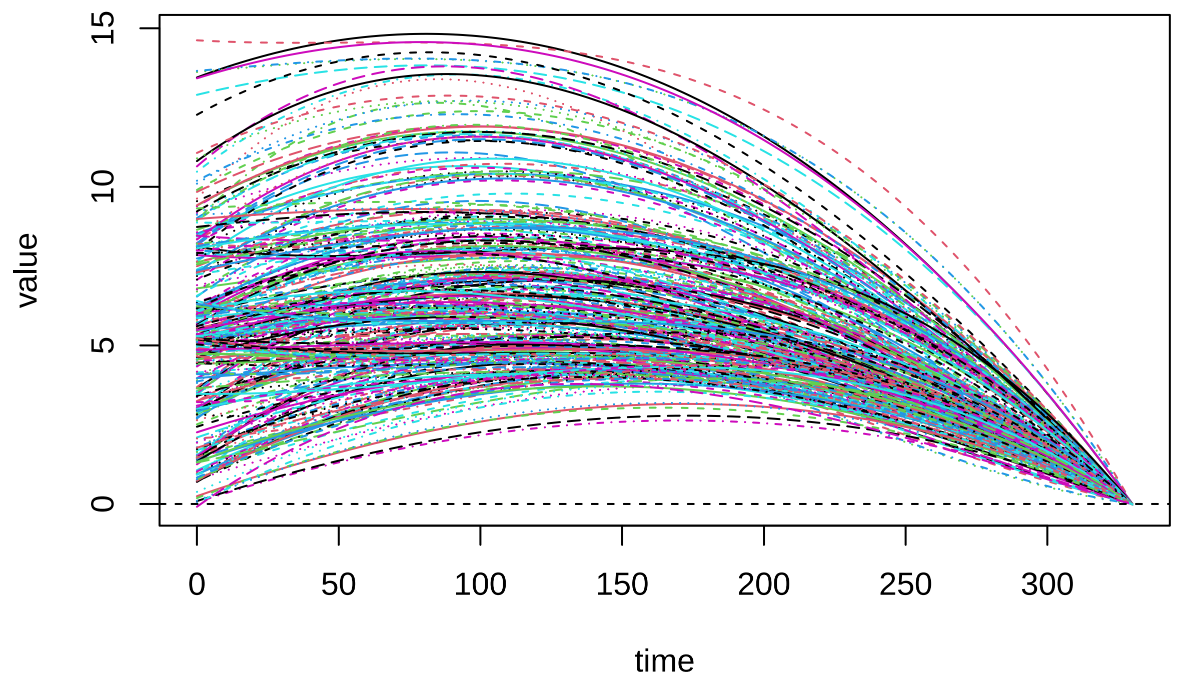
```
yield = FedYieldcurve
terms = yield$x
plot(terms, yield$y[,1], pch=15, ylab="Yield", ylim=c(0,16))
points(terms, yield$y[,330], pch=16)
```



The interest rates for January 1982 is much bigger than the interest rates for June 2009.

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?

```
yield_data = yield$y
basis <- create.bspline.basis(c(0,330), nbasis=4)
yield.F <- Data2fd(terms, yield_data, basis)
plot(yield.F)
```

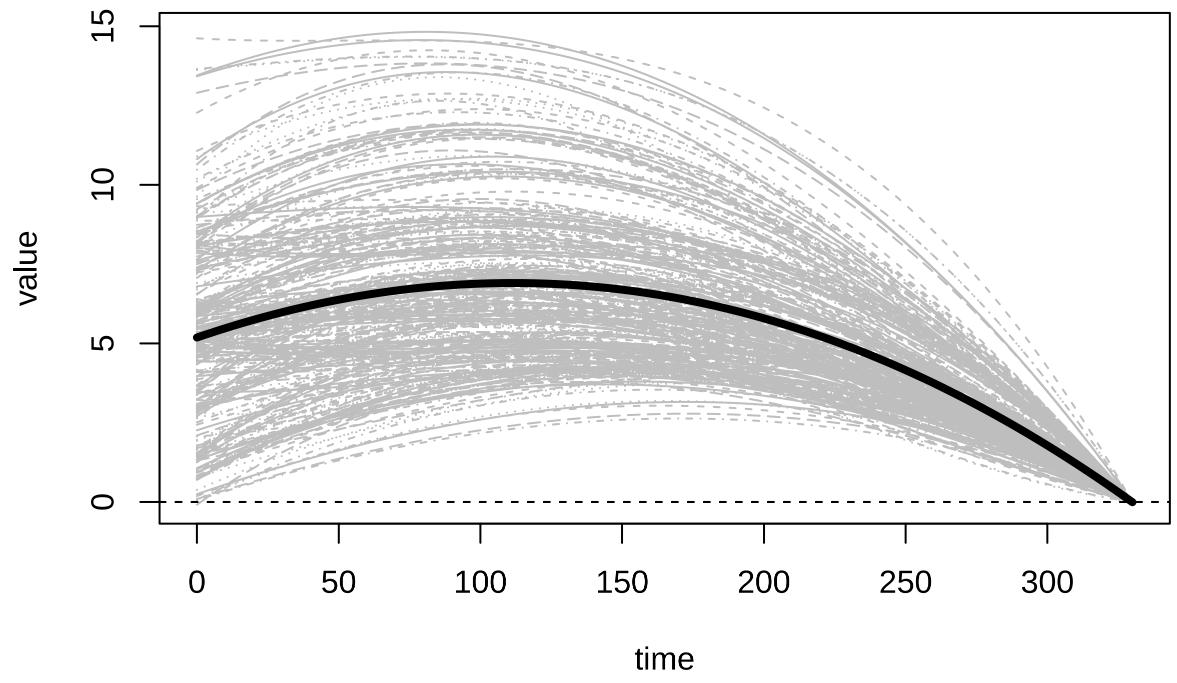


```
## [1] "done"
```

```r
plot(yield.F, col = "grey")
```

```
## [1] "done"
```

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



```
## [1] "done"
```

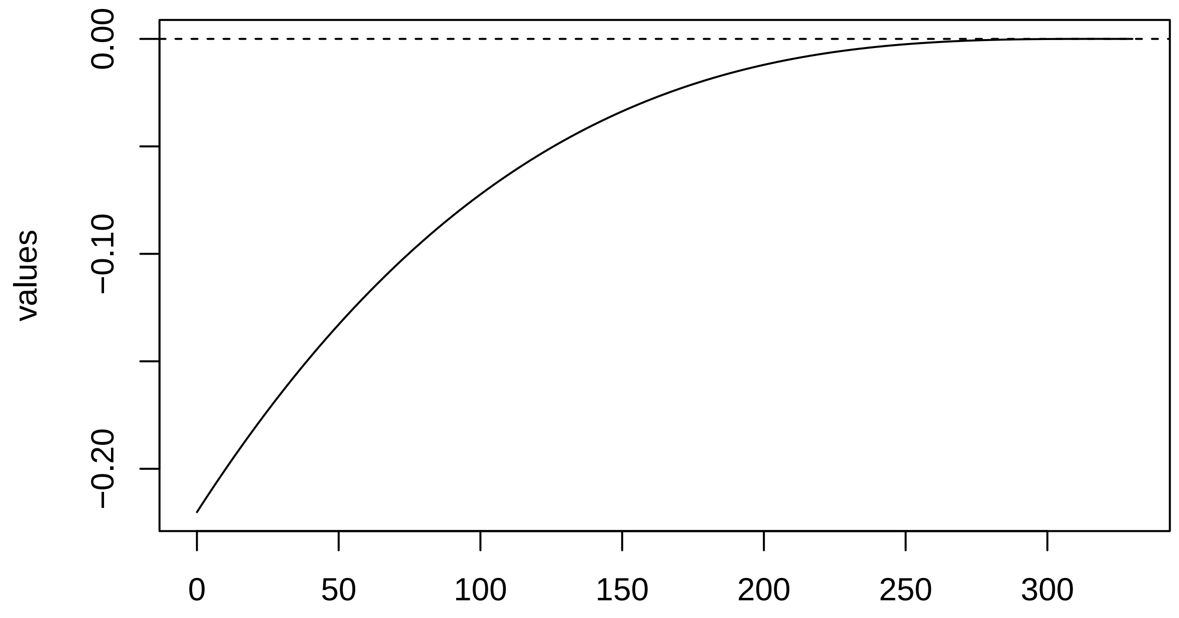
```
mean.fd(yield.F)
```

```
## $coefs
##              mean
## [1,]  5.185258259
## [2,]  8.549438361
## [3,]  7.053882562
## [4,] -0.007704886
##
## $basis
## $call
## basisfd(type = type, rangeval = rangeval, nbasis = nbasis, params = params,
##     dropind = dropind, quadvals = quadvals, values = values,
##     basisvalues = basisvalues)
##
## $type
## [1] "bspline"
##
## $rangeval
## [1]   0 330
##
## $nbasis
## [1] 4
##
## $params
## NULL
##
## $dropind
## NULL
##
## $quadvals
## NULL
##
## $values
## list()
##
## $basisvalues
## list()
##
## $names
## [1] "bspl4.1" "bspl4.2" "bspl4.3" "bspl4.4"
##
## attr(,"class")
## [1] "basisfd"
##
## $fdnames
## $fdnames$time
## [1] "3"   "6"   "12"  "60"  "84"  "120"
##
## $fdnames$reps
## [1] "mean"
##
## $fdnames$values
## [1] "mean value"
##
##
## attr(,"class")
## [1] "fd"
```

It increases as time increases and then decreases at a certain point. This curve is well known as a yield curve.

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.

```
yield.pca = pca.fd(yield.F, nharm=1)
plot(yield.pca$harmonics)
```



```
## [1] "done"
```

```
yield.pca$varprop
```

```
## [1] 0.9999981
```

The the component explains 99.99981% of the varaince. It means that it can be well explained by bspline basis.

Problem 6

Since

$$\hat{c}(t,s) = \frac{1}{M-1}\sum_n\sum_m\sum_k \bar{c}_{nm}\bar{c}_{nk}B_m(t)B_k(s)$$

$$= \frac{1}{M-1}\sum_m\sum_k(\mathbf{a}^\top\mathbf{a})_{m,k}B_m(t)B_k(s) = \sum_m\sum_k(\Sigma_{\mathbf{a}})_{m,k}B_m(t)B_k(s),$$

we can say that $\mathbf{b} = (M-1)\mathbf{a}^\top\mathbf{a}$, so $b_{mk} = (M-1)a_m a_k$.
In fact, you only need to worry about the b_{mm} case.

Chapter 2

Problem 1

Since

$$L(x)(t) = \omega^2 \sum_{j=1}^{J}(\omega j a_j \cos(\omega jt) - \omega j b_j \sin(\omega jt)) + \sum_{j=1}^{J}(-\omega^3 j^3 a_j \cos(\omega jt) + \omega^3 j^3 b_j \sin(\omega jt))$$

$$= \omega^3(j^3 - j)\sum_{j=1}^{J}(-a_j \cos(\omega jt) + b_j \sin(\omega jt)),$$

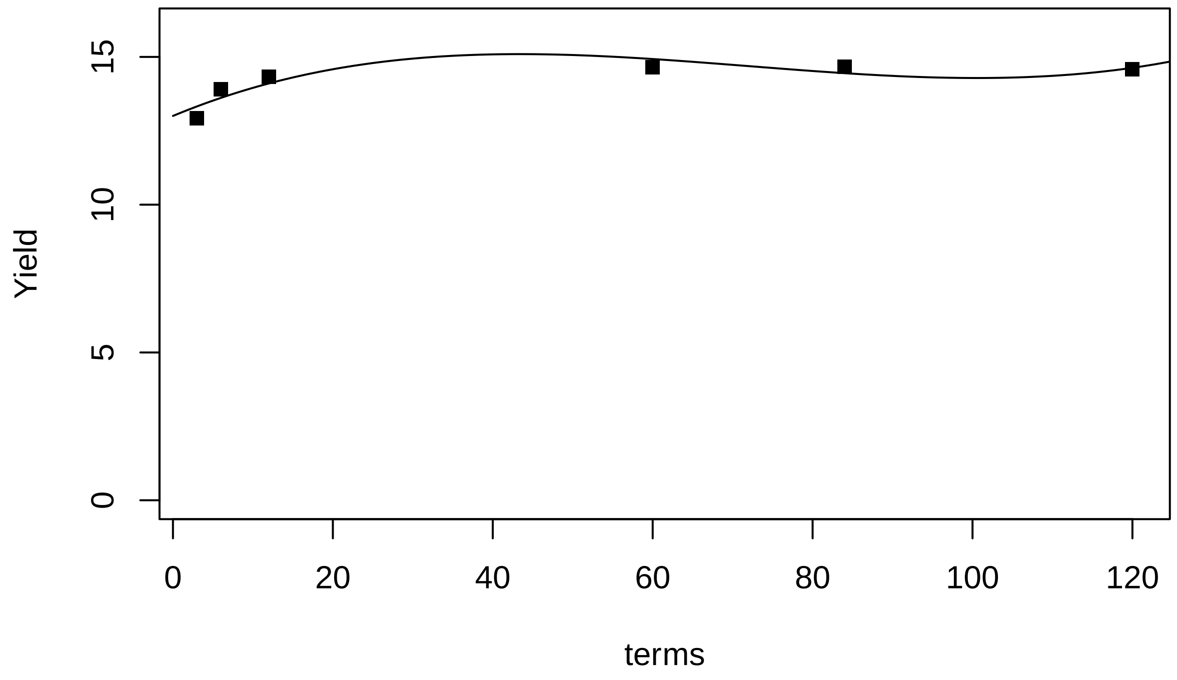
$$\int_0^T [L(x)(t)]^2 dt = \int_0^T (\omega^3(j^3-j))^2(\sum_{j=1}^{J}(-a_j \cos(\omega jt) + b_j \sin(\omega jt)))^2 dt$$

$$= \omega^6 j^2(j^2-1)^2 \int_0^T \sum_{j=1}^{J}(-a_j \cos(\omega jt) + b_j \sin(\omega jt))^2 dt = \pi\omega^5 j^2(j^2-1)^2 \sum_{j=2}^{J}(a_j^2 + b_j^2)$$

Problem 2

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.

```
yield = FedYieldcurve
terms = yield$x
yield_data = yield$y[,1]
basis <- create.bspline.basis(c(0,330), nbasis=4)
yield_smooth <- smooth.basis(terms, yield_data, basis)
plot(terms, yield$y[,1], pch=15, ylab="Yield", ylim=c(0,16))
plot(yield_smooth, add = TRUE)
```



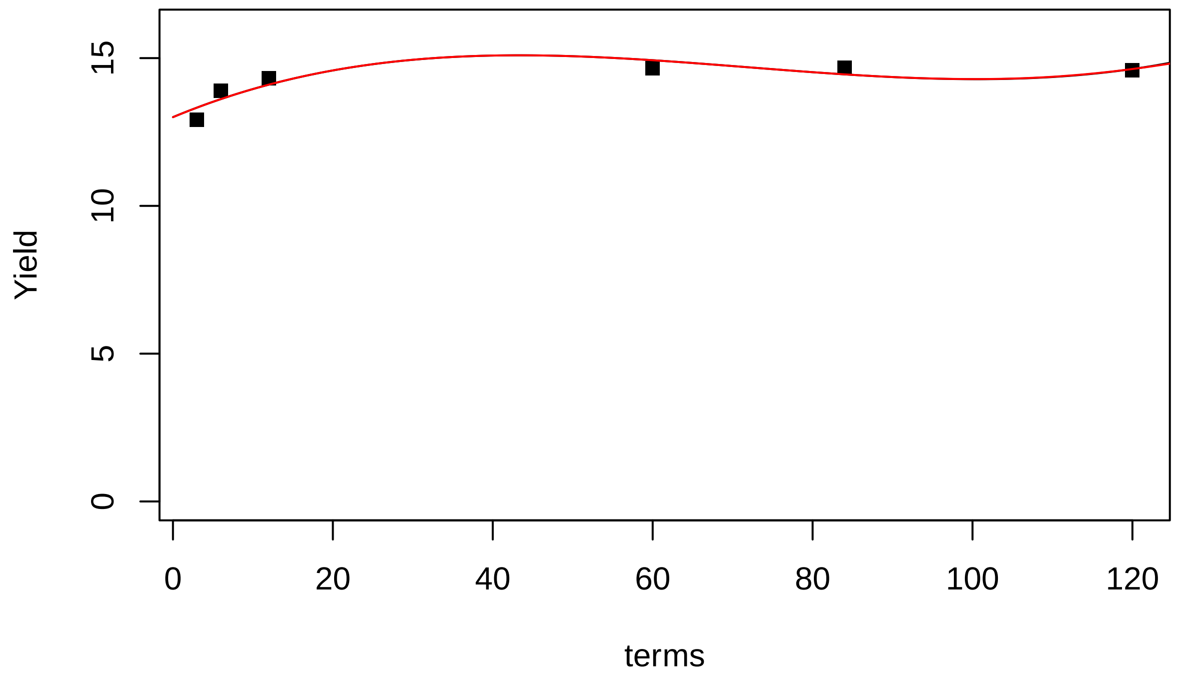
```
## [1] "done"
```

b) Re–fit the January 1982 yields using a penalized smoothing based on six basis functions (as many as data points) with with the smoothing parameter = 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.

```
yield = FedYieldcurve
terms = yield$x
yield_data = yield$y[,1]
basis <- create.bspline.basis(c(0,330), nbasis=4)
yield_smooth <- smooth.basis(terms, yield_data, basis)
plot(terms, yield$y[,1], pch=15, ylab="Yield", ylim=c(0,16))
plot(yield_smooth, add = TRUE)
```

```
## [1] "done"
```

```
basis <- create.bspline.basis(c(0,330), nbasis=6)
yield_par <- fdPar(basis, Lfdobj =2, lambda = 1)
yield_smooth <- smooth.basis(terms, yield_data, yield_par)
plot(yield_smooth, col = "red", add = TRUE)
```



```
## [1] "done"
```

It shows very similar output from a). It is because we are using bspline method.

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

```
yield = FedYieldcurve
terms = yield$x
yield_data = yield$y[,1]
basis <- create.bspline.basis(c(0,330), nbasis=4)
yield_smooth <- smooth.basis(terms, yield_data, basis)
plot(terms, yield$y[,1], pch=15, ylab="Yield", ylim=c(0,16))
plot(yield_smooth, add = TRUE)
```

```
## [1] "done"
```

```
basis <- create.bspline.basis(c(0,330), nbasis=6)
yield_par <- fdPar(basis, Lfdobj =2, lambda = 0.1)
yield_smooth <- smooth.basis(terms, yield_data, yield_par)
print("lambda = 0.1")
```

```
## [1] "lambda = 0.1"
```

```
yield_smooth$gcv
```

```
##      rep1
## 0.6282152
```

```
plot(yield_smooth, col = "red", add = TRUE)
```

```
## [1] "done"
```

```
basis <- create.bspline.basis(c(0,330), nbasis=6)
yield_par <- fdPar(basis, Lfdobj =2, lambda = 0.2)
yield_smooth <- smooth.basis(terms, yield_data, yield_par)
print("lambda = 0.2")
```

```
## [1] "lambda = 0.2"
```

```
yield_smooth$gcv
```

```
##      rep1
## 0.6283764
```

```
plot(yield_smooth, col = "blue", add = TRUE)
```

```
## [1] "done"
```

```
basis <- create.bspline.basis(c(0,330), nbasis=6)
yield_par <- fdPar(basis, Lfdobj =2, lambda = 0.3)
yield_smooth <- smooth.basis(terms, yield_data, yield_par)
print("lambda = 0.3")
```

```
## [1] "lambda = 0.3"
```

```
yield_smooth$gcv
```

```
##      rep1
## 0.6283621
```

```
plot(yield_smooth, col = "green", add = TRUE)
```

```
## [1] "done"
```

```r
basis <- create.bspline.basis(c(0,330), nbasis=6)
yield_par <- fdPar(basis, Lfdobj =2, lambda = 0.5)
yield_smooth <- smooth.basis(terms, yield_data, yield_par)
print("lambda = 0.5")
```

```
## [1] "lambda = 0.5"
```

```r
yield_smooth$gcv
```

```
##      rep1
## 0.6282281
```

```r
plot(yield_smooth, col = "purple", add = TRUE)
```

```
## [1] "done"
```

```r
basis <- create.bspline.basis(c(0,330), nbasis=6)
yield_par <- fdPar(basis, Lfdobj =2, lambda = 1)
yield_smooth <- smooth.basis(terms, yield_data, yield_par)
print("lambda = 1.0")
```

```
## [1] "lambda = 1.0"
```

```r
yield_smooth$gcv
```

```
##      rep1
## 0.6277712
```

```r
plot(yield_smooth, col = "orange", add = TRUE)
```

```
## [1] "done"
```

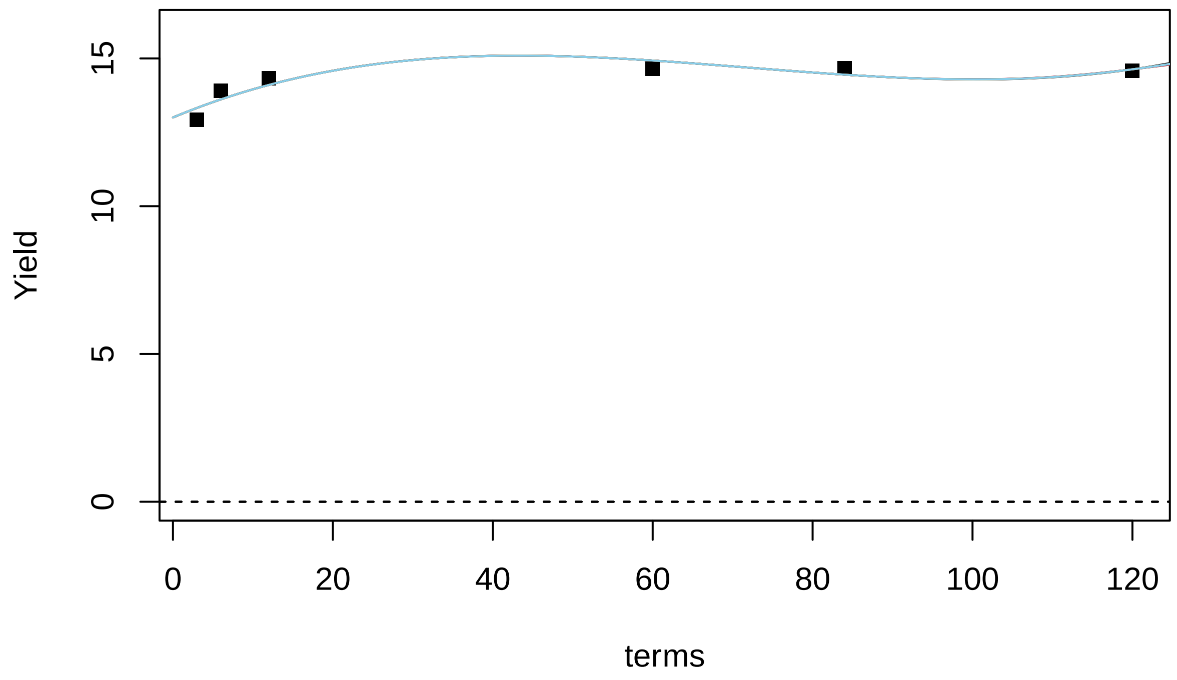
```r
basis <- create.bspline.basis(c(0,330), nbasis=6)
yield_par <- fdPar(basis, Lfdobj =2, lambda = 2)
yield_smooth <- smooth.basis(terms, yield_data, yield_par)
print("lambda = 2.0")
```

```
## [1] "lambda = 2.0"
```

```r
yield_smooth$gcv
```

```
##     rep1
## 0.626784
```

```r
plot(yield_smooth, col = "skyblue", add = TRUE)
```



```
## [1] "done"
```

Comparing to the value of GCV, it seems like the case of $\lambda = 0.2$ gives us a best result in this case.

Problem 5

(a) Simulate a functional sample over the unit interval each with a sample size of 50 from the Mat´ern process. For the first half of the sample, set the mean function equal to the the bump function with parameters (c0, r0, a0) = (3/8, 1/4, 5). For the second half use (c0, r0, a0) = (5/8, 1/4, 5). You may choose the values for the Mat´ern 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.

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.

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.

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.